Supporting user selection of digital libraries.

Dodd, Helen Margaret

Supporting User Selection of Digital Libraries

Helen Margaret Dodd

Submitted to Swansea University in fulfilment
of the requirements for the Degree of Doctor of Philosophy

Swansea University
Prifysgol Abertawe

Department of Computer Science
Swansea University

2013

Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date ..... 1/2/2013 .....

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed ..... (candidate)

Date ..... 1/2/2013 .....

Statement 2

I hereby give my consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)

Date ..... 1/2/2013 .....

Abstract

Subject specialists and researchers often face the problem of identifying authoritative collections: those directly about their topic of interest, to which they regularly return to satisfy related information needs or monitor for new material. Discovery of such collections is often incidental or relies on suggestions from domain experts. Services such as general purpose search engines and repository directories offer limited support for this search task. As such, there is a clear need for a search service specifically to assist users in finding collections that can serve both their current and future information needs; we refer to this task herein as collection suggestion.

However, developing an effective search service of this kind requires fundamental research. There are several preconditions that should be addressed; it is these that form the focus of this thesis. We summarise these areas as follows.

An effective search service calls for an appropriate algorithm; in this instance, an algorithm for ranking collections with respect to the user's query. To this end, we investigate the applicability of existing algorithms, from relevant domains (collection selection and query performance prediction), to collection suggestion. In addition, towards identifying an optimal algorithm for a collection suggestion search service, we specify and test a new algorithm (and several alternative variants), designed specifically for this task.

The requirement of an appropriate algorithm presents the question of how we evaluate the effectiveness of an algorithm. We have formulated a methodology (comprising evaluation strategies and performance measures) and developed apparatus for evaluating algorithms, with respect to collection suggestion. As far as possible, we have drawn on and extended established algorithm evaluation techniques, to ensure our work follows the expectations of information retrieval research.

Our empirical work is conducted over several synthetic and realistic test data sets: we use established data sets built from the TREC document corpus, in addition to data sets of our own compilation, comprising data from real repositories. This combination of test data types ensures a rigorous test environment for algorithms.

Over our test environment, we have found three algorithms to be potentially suitable for application in a collection suggestion search service. One collection selection algorithm (CORI), and two variants of our own algorithm were shown to have strong and consistent performance, across the range of test data sets and performance measures used.

Acknowledgements

I would first like to express my gratitude towards my supervisors, Matt Jones and George Buchanan, for their support and encouragement over the course of my PhD studies. In addition, I owe thanks to them for reading and commenting on the copious quantity of material I have written, and for challenging and questioning me in discussions; my thesis is much stronger as a result.

Many thanks also to my examiners Stefan Rüger and Mark Jones for their valuable feedback, and for making the viva examination an intellectually stimulating (and genuinely enjoyable) experience.

The support of family and friends has been invaluable to me. In particular, I would like to thank my fiancé, Mark, for going through the experience first, enabling him to empathise and give me useful advice, as well as the usual encouragement and proof reading. Thanks also to my parents, for encouragement and assistance throughout my education.

I am grateful for the friendship of my fellow Computer Science PhD students at Swansea University. In particular, special thanks to Liam O'Reilly for assistance with LaTeX, proof reading, and providing me with frequent distractions from my work!

Finally, I would like to acknowledge that my PhD studies have been funded by a Doctoral Training Grant, courtesy of the Engineering and Physical Sciences Research Council.

Table of Contents

Chapter 1

Introduction

Subject specialists and researchers often utilise several domain-specific resources, such as journals, digital libraries and institutional repositories. They periodically return to these sources to satisfy related information needs, and keep abreast of new developments: browsing and monitoring for, and extracting new material [14].

Recent studies on information seeking in digital resources, such as Buchanan et al. [4] and Xie and Cool [60], suggest that people have difficulty identifying relevant resources and collections. Indeed, familiarity with authoritative collections develops through perseverance and time: traditionally, by consulting colleagues for recommendations, and by examining and following citations in order to identify sources containing material of interest [14].

Alternatively, some users may begin their pursuit of authoritative collections by using a general purpose search engine (such as Google or Bing). However, while the user will be given links to many relevant documents and web pages, there is little indication of whether a given document is representative of its collection [2, p. 267]. That is, without browsing the website, the user will not know if the collection is relevant as a whole, or if they have only found an isolated relevant document.

The task of supporting users in finding relevant resources has received some recent attention. For example, Song et al. [53] and Seo and Croft [46] have focused on relevant website and blog recommendation, respectively.

Song et al. also discuss the potential advantages of a user visiting a domain-specific resource, over performing traditional document search on a general purpose search engine. For instance, as the resource contains documents on a specific topic, the user is less likely to have to filter through completely unrelated search results (consider, for example, the polysemy problem). In addition, the user may benefit from additional material not indexed by a general purpose search engine, such as proprietary or pay-for content. Finally, a domain-specific resource may present results in a more appropriate format or interface for the content, with greater flexibility in how to rank the results [53].

In view of the difficulties faced by users in locating relevant domain-specific collections, and the potential benefits of using such resources, there is a clear need for a search service to assist users with this search task. For example, a search service for collections could take

a user's query (formed from keywords related to their topic of interest), and return a ranked list of resource collections. The collections ranked highest are those that are most relevant, as a whole, to the user's query. The user can then search and browse these resources themselves, and return to them for related information needs.

While such a search service may be intended to operate over digital collections, it could also have value for identifying authoritative *physical* collections. For example, there are many museums and archives scattered around the country, each containing collections of various types of historical and cultural artefacts. A person who wishes to study a particular type of artefact may like to visit the few places with the most extensive and complete collection of such artefacts. As physically travelling to a location can be expensive, in terms of time and money, it is imperative that the person choose the most relevant and useful collection. Visiting a location containing only one or two interesting artefacts is costly, and may cause frustration. Indeed, this scenario further emphasises the importance and advantage of successfully identifying authoritative collections, whether digital or physical.

Developing an effective search service for collections requires fundamental research. As such, the remainder of this chapter provides an overview of recent tools and research relevant to the task of identifying relevant collections. In addition, we highlight the contributions, method and scope of our work. Finally, we give a detailed chapter outline for this thesis.

1.1 State of the Art

In this section we examine the current state of support for users wishing to find domain-specific collections, and introduce relevant research domains.

1.1.1 Collection Finding Tools

Currently, potential starting points for users wishing to find domain-specific collections are repository and resource directories such as OpenDOAR[1], Intute[2] and IMLS DCC[3] (Institute of Library Services Digital Collections and Content). However, these services are of limited value.

OpenDOAR provides a list of around 2000 collections, such as digital libraries and university repositories, which may be filtered by their general subject areas. Alternatively, the user can search for a collection, if they already know its name.

Intute (which as of July 2011 is no longer maintained or updated) comprises a database of research and education web resources that have been manually selected and evaluated by subject specialists. Again, the user can browse resources by subject area, or they can submit a query to search for relevant resources. However, searches on Intute are executed over human-compiled descriptions and keywords, which may not be sufficient to represent the full scope and coverage of the resource.

[1] `http://www.opendoar.org/`
[2] `http://www.intute.ac.uk/`
[3] `http://imlsdcc.grainger.uiuc.edu/`

The IMLS DCC service offers similar functionality to OpenDOAR and Intute: browse for collections by subject, and search based on collection titles and descriptions.

These services are reminiscent of the original website directory services, such as Yahoo!. However, such services were superseded by search engines such as Google. As such, there is scope for a similar transition within collection finding tools.

1.1.2 Collection Selection and Federated Search

The problem of ranking collections and resources according to their relevance to a query is not a new one: often referred to as *collection selection*, it is traditionally thought of as a sub-problem of federated search.

A federated search service operates in the domain of traditional document retrieval, aiming to extend the coverage of a search by dispatching a user's query to multiple search engines and resources.[4] Search results are returned to the federated search service, which aggregates them into a single result list, ordered by relevance [38].

In federated search, collection selection is the process of identifying a subset of resources most likely to contain or index relevant material. The motivation for this is straightforward: submitting a query to only potentially useful resources reduces network traffic and consumption of system resources [38].

There is a large body of research investigating collection ranking algorithms for collection selection, and evaluating the effectiveness of those algorithms. Collection selection is highly relevant to our work; as such, we provide a detailed discussion of the domain and its practices in Chapter 2 of this thesis.

However, we note here that there are differences between collection selection and the task of recommending domain-specific collections. In the federated search paradigm, collection selection supports document retrieval. The goal is to maximise the number of highly relevant documents in the final results list, while minimising the number of resources the query is sent to. Therefore, collection selection algorithms aim to order collections by the number of relevant documents they are likely to contain.

In contrast, our goal, as described at the start of this chapter, is not to find collections merely containing some number of relevant documents, but to identify authoritative collections: those that are directly about the user's query. We suggest that such collections should have a high *quantity* of documents relevant to the query, which amount to a significant *proportion* of the collection. As such, in this thesis we essentially reformulate collection selection as an independent search task.

Given the context of current support for users seeking domain-specific collections, and existing relevant research, in the following section we highlight the contributions and scope of our work.

[4] Gulli and Signorini [24] found there is only a small overlap in the pages a search engine indexes, and thus a user could benefit from utilising several search engines.

1.2 Contributions

Through the evaluation of existing literature, we have identified a potential user need that is not currently addressed by existing search services. That is, support for identifying collections that are authorities on a particular topic, that the user can visit to satisfy both current and future information needs.

As such, the motivating aim of this work is to develop a search service for identifying collections that are relevant to a user's query. Such a service would rank resource collections according to a user's query. In the remainder of this thesis we will refer to this task as *collection suggestion.*

Developing an effective search service of this kind demands fundamental research. For example, the first question that comes to mind is: how do we rank collections according to a user's query? We require an effective algorithm for this. Consequently, how do we determine whether an algorithm is optimal for the collection suggestion task? For this, we require an evaluation methodology, and appropriate test data. It is these elements that we deal with in this thesis. In addressing these areas, a logical starting point is to draw from existing work from relevant domains (such as collection selection), and make reasoned alterations and additions where required.

We specify our fundamental contributions towards collection suggestion, as follows:

Evaluate the effectiveness of collection selection and query performance prediction algorithms, for collection suggestion: To implement a collection suggestion search service, we require a suitable algorithm for ranking collections with respect to a user's query.

In identifying such an algorithm, a logical starting point is to investigate whether any existing algorithms may effectively be applied to the task. To this end, we empirically evaluate the suitability of algorithms from two relevant domains: collection selection (discussed previously in Section 1.1.2), and query performance prediction (where algorithms are designed to estimate the quality of retrieval results for a query executed on a single collection). We discuss the motivation behind these choices in Chapter 2.

Specify and evaluate a new algorithm for collection suggestion: We develop a new collection ranking algorithm, specifically for the collection suggestion task. We evaluate its performance against that of the algorithms from other domains; our objective is to develop an algorithm that exceeds the performance achieved by existing algorithms.

Develop a robust methodology, apparatus and test data for evaluating algorithms for collection suggestion: To evaluate the effectiveness of algorithms for collection suggestion, we require a methodology and apparatus. As the related collection selection domain has an established methodology, we will use this as a basis for our experiments, adapting and extending it to reflect the specific objectives of collection suggestion. In addition, we develop an extensive testing apparatus to support the execution of our experiments.

We utilise multiple test data sets (made up of document collections and test queries) for our algorithm evaluations. In addition to using established TREC-based data sets

from the collection selection domain, we also develop and use our own test data sets, comprising real data from open access repositories. In this way, we can test algorithms on both controlled and realistic test data, providing a rigorous test environment.

We note here that our empirical work is focussed solely on the effectiveness of potential collection ranking algorithms for collection suggestion. We do not undertake subjective user-based evaluation of the value of a collection suggestion service, and its effectiveness from a user's standpoint, at this time. Our rationale for this is straightforward: such an evaluation can only occur on a mature prototype. As we have discussed, there are several prerequisites for achieving such a prototype.

The above contributions are addressed in this thesis, according to the chapter structure given in the next section.

1.3 Chapter Outline

The remainder of the thesis is organised as follows:

Chapter 2: Here we provide a comprehensive discussion of literature relevant to this research; in particular we consider the current evaluation techniques used in collection selection, and traditional information retrieval. We look to draw on these established techniques for our evaluation of algorithms for collection suggestion. As such, we identify aspects of the methodologies that we must adapt to better reflect the objectives of the collection suggestion task.

In addition, this chapter also summarises the existing collection selection algorithms (and some algorithms from another similar domain: query performance predictors), which we later test for applicability to collection suggestion.

Chapter 3: Given our review of current evaluation techniques (in Chapter 2), we introduce our methodology for evaluating the effectiveness of algorithms for ranking collections, with respect to collection suggestion.

Our primary strategy follows that used to evaluate collection selection algorithms: utilising several performance measures to assess how closely an algorithm-produced collection ranking matches an optimal ranking of collections. We use statistical methods to reflect on the significance of the performance scores; something often overlooked in collection selection algorithm evaluations.

In addition, we develop and use a scenario-based evaluation technique. This acts as a preliminary stage to algorithm evaluation, in order to identify and filter out any algorithms that are clearly unsuitable for collection suggestion.

Chapter 4: In this chapter we discuss the test data sets used in our algorithm evaluations. We utilise several test data sets: some comprise TREC data (following previous convention), while others are built from real repositories.

Chapter 5: A key contribution of this thesis is to develop a new collection ranking algorithm, designed with the specific goals of collection suggestion in mind. Thus, this chapter provides the initial specification of our algorithm. To ensure we have an effective algorithm, we use our test data sets and evaluation methodology to examine various formulations of the algorithm. In this way, we can continue our experiments (comparing its performance to existing algorithms) with the most effective algorithm formulation.

Chapter 6: In this chapter we undertake the first part of our comprehensive evaluation of algorithms, with respect to the collection suggestion task. Here, we conduct experiments using controlled TREC-based test data sets.

Chapter 7: Continuing our evaluation of the effectiveness of algorithms for collection suggestion, here we use test data sets formulated from real repository data. Thus, we test algorithms in conditions that are representative of the intended operational environment.

Chapter 8: Finally, in Chapter 8 we conclude: we summarise our findings and contributions, and reflect on possible future directions of work in this domain.

Chapter 2

Literature Review

In the previous chapter, we identified the need for a search service to provide support for users wishing to identify collections, such as digital libraries, that are relevant as a whole to their topic of interest. We have dubbed this task as *collection suggestion.*

The focus of this work is to address the prerequisites associated with developing an effective collection suggestion search service. First and foremost, we require an algorithm that can effectively rank collections according to the user's query. Consequently, we require a methodology and test data to evaluate the effectiveness of potential algorithms for this task.

To this end, there are several existing areas of research that are relevant to us, that we can utilise, experiment with, and draw inspiration from. This chapter provides a detailed discussion of these topics. Specifically, in Section 2.1 we begin by introducing the domain of collection selection, which is of primary interest to us. Collection selection is a sub-problem of federated search; federated search services improve coverage of document retrieval by simultaneously submitting a user's query to multiple other search services. Thus, collection selection is a precursor to document retrieval, used to identify resources that are likely to contain many relevant documents, to which the user's query can be sent.

Following this, in Section 2.2 we provide an overview and critique of the established techniques for retrieval system evaluation; we will follow accepted evaluation procedures where appropriate, to ensure our work meets the expectations of information retrieval research.

Given the wider context of information retrieval evaluation practices, in Section 2.3 we discuss techniques commonly used to evaluate collection selection algorithms: we draw on and amend these for our evaluation of algorithms for collection suggestion. In Section 2.4 we give a detailed discussion of the state of the art collection selection algorithms: we will later evaluate a selection of these, to investigate their suitability for the collection suggestion task.

In Section 2.5 we discuss an additional class of algorithm that we will evaluate for applicability to collection suggestion: namely, query performance predictors. These algorithms are normally used to evaluate the quality of results likely to be achieved at a search engine (so that methods to boost retrieval performance, such as query term expansion, can be ap-

plied to queries predicted to return poor results). We speculate that a collection ranking based on the predicted quality of retrieval results within each collection may be suitable for the collection suggestion task.

Finally, in Section 2.6 we summarise our discussion of relevant literature, and reflect on its application in our own work.

2.1 An Overview of Collection Selection

The focus of this work is to conduct fundamental research towards providing support for a user wishing to find collections, such as digital libraries, that will have long term value: collections that can satisfy their current and future information needs. One body of research that is highly relevant to this task is that of *collection selection* (sometimes referred to as database selection [23, 38], server selection [56], or resource selection [50]): one of the sub-problems associated with *federated search.* There are three sub-problems associated with federated search: collection representation, collection selection, and results merging [48]. We touch on each of these in this section.

Federated search systems, such as metasearch engines, support a user performing traditional document retrieval. The aim of such systems is to maximise the number of relevant documents returned to the user.

The scale of the web is vast [36, 24], with a plethora of search services, such as general purpose search engines and specialist repositories [38, 29, 48], each indexing different content. Federated search extends search coverage by providing unified access to these search engines and collections [38, 48]. Given a user's query, the federated search service (the *broker*) passes it to other search engines and collections. Thus, multiple resources are searched simultaneously. The search results from the various resources are returned to the broker, which aggregates them into a list, ordered by global relevance (results merging).

A federated search service can potentially forward a user's query to hundreds or thousands of resources. However, doing so would incur unnecessary network traffic and waste system resources both at the broker and the resource: inevitably, resources that indexed little or no relevant documents would be queried [38, 48]. Thus, collection selection is the process of deciding which resources index potentially relevant material, by ranking the resources according to the user's query (similar to the way in which documents are ranked in document retrieval).

To choose suitable collections, a summary or representation of each collection is required (collection representation); these are often statistics such as term and document frequencies and collection size. However, their comprehensiveness depends on the level of cooperation offered by the resource. A cooperative resource will allow the broker access to complete lexicon statistics, derived from access to the full collection contents. However, an uncooperative resource will not make these statistics available to the broker, and so they must be estimated through other techniques, such as document sampling [48].

In the domain of federated search, collection selection acts as a precursor to document retrieval; the goal of federated search is to maximise the number of relevant documents retrieved, while minimising the number of resources that are queried. This goal is reflected

in the operation of the many collection selection algorithms documented in the literature: they aim to emulate a collection ranking that favours collections containing a large number of relevant documents.

Our goal differs from that of collection selection: we do not intend to perform document retrieval, but rather identify collections the user may browse themselves. As such, we are exposing the entire collection to the user. In this instance, a collection ranking based solely on the quantity of relevant documents is not appropriate: there may be many documents in the collection that are not relevant to the user's topic of interest. Instead, the collection ranking should take into account both the number and concentration of relevant documents likely to be in the collection.

As discussed above, collection selection is traditionally considered as a sub-problem of federated search, used to identify collections likely to contain relevant documents. In our work however, we treat 'collection selection' as a search task in its own right, where the goal is to identify collections that are relevant as a whole to a user's query. To avoid confusion, we refer to this task as *collection suggestion.*

A variety of algorithms have been developed for traditional collection selection; as such, a primary contribution of this work is to evaluate whether they are suitable for the alternative collection suggestion scenario, where our criteria for choosing highly ranked collections differs from that of collection selection.

To this end, we present and discuss several collection selection algorithms in Section 2.4. Prior to this, in the following section we offer an overview of established information retrieval evaluation techniques. This provides context for our discussion and critique of accepted collection selection algorithm evaluation techniques, given in Section 2.3.

2.2 Overview of Retrieval System Evaluation

The central focus of our work is the evaluation of algorithms with respect to the collection suggestion task. For this, we require an evaluation methodology.

To ensure our work meets the expectations of information retrieval research, we follow and draw from established and accepted evaluation techniques. This section gives a general overview of the standard practices of retrieval system evaluation, and reflects on any associated limitations.

2.2.1 Performance

There are two aspects of performance to consider when evaluating an information retrieval system: *efficiency* and *effectiveness.* When examining the efficiency of a retrieval system, we are concerned with the cost and time associated with executing a search. In contrast, the effectiveness of a system indicates the precision of the result set. For example, in document retrieval, effectiveness measures the ability of the system to retrieve relevant documents, while ensuring non-relevant documents are not returned [58, p. 10,145] [45, p. 158].

A complete evaluation of a retrieval system should examine both its efficiency and effectiveness. However, many evaluations of retrieval systems focus primarily on effectiveness;

while efficiency is important, if a system is not effective, its efficiency is somewhat irrelevant.

Depending on the scenario, a variety of performance measures can be used to determine the effectiveness of a retrieval system. Different metrics investigate different aspects of performance, and therefore selection of evaluation metrics should be based on the feature of retrieval behaviour that is of interest [5, p. 73].

Two of the most fundamental metrics used in document retrieval system evaluation are *recall* (the fraction of all possible relevant documents that have been retrieved) and *precision* (the fraction of retrieved documents that are actually relevant) [2]. The aim is to maximise both of these metrics; however there is often a trade-off between the two. High recall may come at the expense of several non-relevant documents within the result set, while high precision can mean some relevant documents are missed.

2.2.2 Format of Evaluations

In document retrieval, the established methodology for evaluating the effectiveness of retrieval systems follows that used in the Cranfield tests [7]. Here, three components were utilised: a collection of documents, a set of queries to execute against the documents, and a set of relevance judgements indicating which documents are relevant to each query. The relevance judgements are used to calculate performance scores, such as recall and precision.

This format is not just applicable to evaluating traditional document retrieval systems, it can also be applied to evaluate systems in other scenarios, such as collection selection (see Section 2.3).

2.2.3 Analysis of Results

Two categories of analysis may be used to understand and scrutinise the results produced by the various performance metrics used in a retrieval system evaluation: *descriptive* and *deductive* [33].

Descriptive analysis presents results diagrammatically, to show general trends in the data. Deductive analysis is more concrete, and examines the statistical significance of results. For example, statistical tests such as t-test [33] and the sign test [58] help determine the importance of a result, and whether there are any real differences between two systems.

Ideally, to provide a robust evaluation, both descriptive and deductive analysis should be used in combination; however, as we see in Section 2.2.4 (and Section 2.3, where we discuss techniques for the evaluation of collection selection algorithms), deductive analysis is generally neglected [30].

2.2.4 TREC

One of the most important developments in document retrieval evaluation was the introduction of the Text REtrieval Conference (TREC). Prior to TREC, the evaluation of retrieval systems was chaotic, with no agreement on how to evaluate results. Although evaluations

tended to follow the Cranfield method, different test collections and performance measures were used. As such, the findings from different experiments were often not comparable [59]. The purpose of TREC was to provide a consistent and robust testbed for evaluation.

Like the Cranfield tests, TREC utilises three components: a set of documents, query topics, and relevance judgements. The TREC document collections consists of several gigabytes of newspaper and government data: amounting to over a million documents. The query topics, created by domain experts, describe an information need, indicating what makes a document relevant. The relevance judgements are compiled using the *pooling* method, whereby retrieval systems participating in TREC submit their top n relevant documents for each query into a pool. Human assessors then evaluate the documents in this pool, to determine their relevance to the query topic [25].

In TREC, the evaluation of retrieval systems is supported by the `trec_eval` program, which computes scores for a variety of performance measures. These scores are used to produce a performance report for each retrieval system, which includes: precision at document cutoff values, precision at recall points, eleven point average, R-precision, Mean-Average Precision (MAP), and a recall-precision curve [5].

Rather than decide on an outright winning retrieval system, TREC aims to identify techniques that work well for information retrieval [55]. To facilitate this, the participating systems are ranked by MAP scores, and the best performing systems are presented together on a recall-precision graph, to allow comparison of their performances.

With the advent of TREC, many researchers use the TREC document corpus when conducting their own evaluations. Thus, findings from different studies can be compared. In general, evaluations of document retrieval systems tend to use recall-precision graphs and MAP.

However, decisions on which systems perform best are often made based on observations over recall-precision graphs, and by which scores the highest MAP. This a descriptive analysis; the deductive, statistical analysis of scores is often overlooked. As such, there is little evidence as to whether differences in scores are significant; or indeed, if the 'best' system is actually *good enough.*

We continue our discussion of existing retrieval evaluation procedures in the following section, where we consider the practices used to evaluate collection selection algorithms.

2.3 Collection Selection Evaluation

In Section 2.2.2, we described how evaluations of document retrieval systems follow the Cranfield method: involving test documents and queries, and a set of document relevance judgements. Evaluations of collection selection algorithms are also based on this method, but with some differences. Rather than a single collection of documents, multiple document collections are required; instead of relevance judgements, *baseline rankings* are used. These often represent the 'right answer': the *optimal* ordering of collections.

Since the goals of collection selection and document retrieval differ (ranking collections instead of documents), the performance measures used in document retrieval evaluations are not applicable to collection selection evaluations. As such, a combination of new and

existing metrics are employed to measure the key aspects of collection selection algorithm performance.

In the following sections we discuss the elements of collection selection algorithm evaluation: baselines, performance measures, and test data.

2.3.1 Baselines

In collection selection, every collection has a degree of *merit* associated with it, with respect to a query. Merit can be defined in a variety of ways, depending on the goal. For example, the merit of a collection could be given by the number or proportion of relevant documents it contains, or the total number of documents in the collection [16]. A collection selection algorithm should aim to order collections by decreasing merit.

The established technique for evaluating algorithms for collection selection is centred around merit, from which we can derive *baselines* for algorithm performance. A variety of performance measures (discussed in Section 2.3.2) are used to compare how well an estimated collection ranking (produced by a collection selection algorithm) matches a baseline ranking. Several baselines have been used in the literature, most notably:

Count-Based Ranking (CBR): Collections are ranked in decreasing order of the number of documents they contain, that satisfy a boolean predicate. For example, a document is counted if it contains all terms in the query [22].

Ideal: Collections are ranked by a *goodness* score; the goodness of a collection is the similarity between the query and each document in the collection. Documents with a similarity greater than a user-defined threshold contribute to the goodness score. Therefore, goodness is an accumulation of document-query similarity scores [23].

Relevance-Based Ranking (RBR): Collections are ranked in decreasing order of the number of relevant documents they contain [6]. This baseline requires access to relevance judgements for the documents in each collection, for each test query.

Size-Based Ranking (SBR): Collections are ranked in decreasing order of the total number of documents they contain [17]. This is a fixed baseline, constant for all possible queries.

The most frequently used baselines are Relevance-Based Ranking (RBR), and Size-Based Ranking (SBR). The RBR baseline, like the majority of collection selection baselines, represents an *optimal* collection ranking; it is the performance to which algorithms should aspire. In contrast, the SBR baseline is often used to represent a *lower bound* on performance; a collection selection algorithm should aim to do no worse than the SBR baseline. Different baselines may be used in an evaluation, depending on what collection properties we are interested in.

In the following section we discuss the performance measures commonly used to examine the relationship between algorithm-produced collection rankings (for collection selection), and baseline collection rankings.

2.3.2 Performance Measures

In the previous section we saw that the evaluation of the effectiveness of collection selection algorithms is based around a baseline ranking. This often represents an optimal ordering of collections, based on some desirable collection property. However, a baseline can also represent a lower bound on performance.

Given a baseline ranking, and a collection ranking produced by a collection selection algorithm, the question remains: what is an appropriate measure for comparing the two rankings?

A variety of performance measures have been presented in the collection selection literature, some of which have been developed specifically for the task, while others are drawn from other domains. In general, each metric examines different aspects of performance, and thus when used in combination, the strengths and weaknesses of an algorithm can be determined.

We present the most commonly used of these metrics below, and describe how their results are presented and interpreted. Note that, in each metric, a baseline ranking is denoted by B, while E represents the ranking produced by a collection selection algorithm. This follows the convention of French et al. [18].

Mean-Squared Error (MSE)

Mean-Squared Error (MSE) was first used in a collection selection evaluation by Callan et al. [6], and later by French et al. [18, 16]; it measures the dispersion between a baseline collection ranking, and a collection ranking produced by an algorithm. That is, the metric indicates how much the two collection rankings differ. For a single query, MSE is calculated by:

$$MSE(E,B) = \frac{1}{n} \cdot \sum_{i=1}^{n} (B_i - E_i)^2,$$

where:

B_i is the rank of collection i, in the baseline ranking;

E_i is the rank of collection i, according to the collection selection algorithm; and

n is the number of collections being ranked.

An optimal MSE is 0: there is no variation between the two rankings. The MSE scores achieved by a particular algorithm over every query are often presented on a scatter graph. In addition, a summary MSE is given by averaging the scores over all queries.

To help derive meaning from, and quantify a MSE score, French and Powell [16] calculated an upper bound on MSE, by considering the reverse of the baseline ranking: the worst case behaviour. Thus, the maximum MSE, MSE_{max}, may be calculated by:

$$MSE_{max} = \frac{n^2 - 1}{3},$$

where n is the number of collections being ranked [16]. With this, average MSE scores for an algorithm can also be presented as a percentage of the maximum MSE.

Spearman Rank Correlation

The Spearman rank correlation coefficient has primarily been used in collection selection algorithm evaluations by French et al. [17, 16, 44]. It measures the correlation between a baseline collection ranking, and the collection ranking produced by an algorithm. The correlation, ρ, is calculated by:

$$\rho = 1 - \frac{6 \cdot \sum_{i=1}^{n} D_i^2}{n(n^2-1)},$$

where:

D_i is the difference between the ith paired ranks in the baseline and algorithm collection rankings; and

n is the number of collections being ranked [16].

The Spearman rank correlation is similar to Mean-Squared Error, in that both look at the relationship between two rankings. However, Spearman rank correlation may be considered to be more meaningful. The metric produces values between 1 and -1: a positive score indicates a positive relationship between the two rankings, with 1 indicating total agreement. In contrast, a negative score shows a negative relationship, with a score of -1 showing that the two rankings are the exact opposite of each other.

In the literature, Spearman rank correlation is used to compare algorithm performance to the SBR fixed baseline, rather than an optimal ranking such as RBR. As such, those evaluations have investigated whether algorithms show a tendency to favour large collections over smaller, possibly more relevant, collections.

The Spearman rank correlation results are presented similarly to MSE scores: a scatter graph plots the correlation achieved on each query, with a summary score averaging the correlations over every query also being given. Unfortunately, the analysis and interpretation of Spearman scores tends to be informal, with casual statements made. For example, one evaluation states that an algorithm "exhibits a very strong positive correlation with SBR" [44]. Such statements are open to interpretation. To properly qualify correlation scores, statistical tests such as t-test should be applied, to provide a clear indication of the *significance* of the correlations, and to show that the similarity of the two rankings is unlikely to have occurred by coincidence.

$\mathcal{R}_n$

Defined by Gravano et al. [23], $\mathcal{R}_n$ is analogous to the recall metric used in document retrieval. $\mathcal{R}_n$ measures the fraction of the *available merit* from the top n ranked collections of the baseline, that has been accumulated by the top n collections in the algorithm-produced collection ranking [23, 17]. That is, it measures how well an algorithm selects the best collections, according to an optimal baseline ranking. $\mathcal{R}_n$ is calculated by:

$$\mathcal{R}_n(E, B) = \frac{\sum_{i=1}^{n} E_i}{\sum_{i=1}^{n} B_i},$$

where:

E_i is the merit associated with the ith ranked collection in the collection ranking produced by an algorithm;

B_i is the merit associated with the ith ranked collection in the baseline ranking; and

n is the rank position at which we wish to calculate the score [16].

The $\mathcal{R}_n$ values are averaged over all test queries, and plotted on a graph as a curve, for each possible value of n (one, to the number of collections being ordered). To help interpret the results, the optimal performance (the optimal baseline against itself) is also plotted on the graph. Thus, we can observe how far from the optimal performance an algorithm is.

The $\mathcal{R}_n$ metric is perhaps the most established of those used in collection selection algorithm evaluations: recent work, such as that of Si and Callan [49, 50], Shokouhi and Thomas [47, 57], uses the measure exclusively.

$\hat{\mathcal{R}}_n$

An alternative to the $\mathcal{R}_n$ metric described above, $\hat{\mathcal{R}}_n$ was proposed by French et al. [18]; rather than measure the available merit accumulated, $\hat{\mathcal{R}}_n$ measures the accumulation of the *total merit* by the top n collections in a ranking produced by an algorithm. $\hat{\mathcal{R}}_n$ is defined as follows:

$$\hat{\mathcal{R}}_n(E,B) = \frac{\sum_{i=1}^{n} E_i}{\sum_{i=1}^{n^*} B_i},$$

where:

E_i is the merit associated with the ith ranked collection in the collection ranking produced by an algorithm;

B_i is the merit associated with the ith ranked collection in the baseline ranking;

n is the rank position at which we wish to calculate the score; and

$n^* = k$ such that $B_k \neq 0$ and $B_{k+1} = 0$. That is, n^* is the rank position of the last collection with non-zero merit [16].

The $\hat{\mathcal{R}}_n$ scores are presented in the same way as those of $\mathcal{R}_n$: scores for all possible values of n are calculated, and averaged over all queries. These are plotted as a line graph, with the optimal performance also plotted for comparison.

$\mathcal{P}_n$

In addition to their recall-like metric, Gravano et al. [23] also specified a metric analogous to precision. The $\mathcal{P}_n$ metric gives the fraction of the top n collections in an algorithm-produced collection ranking that have non-zero merit, according to the baseline. That is, at each value of n, it shows the proportion of selected collections that have some merit. $\mathcal{P}_n$ is calculated by:

$$\mathcal{P}_n(E,B) = \frac{|\{c \in Top_n(E) | B_c > 0\}|}{|Top_n(E)|},$$

where:

c is a collection;

$Top_n(E)$ is the set of collections in the top n rank positions of a ranking produced by an algorithm;

B_c is the merit associated with collection c, according to the baseline ranking; and

n is the rank position at which we wish to calculate the score [23, 16].

Like the $\mathcal{R}_n$ and $\hat{\mathcal{R}}_n$ metrics, a $\mathcal{P}_n$ score is calculated for all possible values of n, and averaged over all queries. The scores achieved by each algorithm being tested are plotted on a line graph, with the optimal baseline also being shown, to represent the benchmark the algorithms are trying to achieve.

2.3.3 Test Data

An essential component in the evaluation of any type of retrieval system, is the test data; this includes test queries, and a sample of the type of item the retrieval system is intended to operate over.

Whereas a document retrieval system is evaluated using a single collection of documents, collection selection evaluation requires multiple collections of documents. To conduct a robust evaluation, an appropriate number of collections is required, and these collections should be representative of the intended operational environment [10]. For example, the collections should be of varied sizes and subjects, but the content of an individual collection should follow a single theme or topic [10].

There are two strategies for compiling test data for collection selection algorithm evaluations: create artificial collections using the TREC document corpus, or use real collections. Both techniques have advantages and disadvantages; we discuss these in this section.

The most frequently used and favoured strategy for compiling test data for evaluating collection selection algorithms is to use the TREC corpus, in particular discs 1–3 (for example, French and Powell [18, 44], Si [49], Shokouhi [47] and Thomas [57] have all used similar TREC-based corpora). This TREC material consists of over a million documents from several (primarily news and government) sources.

In an attempt to create a realistic test environment, the documents in the corpus are grouped to create synthetic collections. Grouping may be performed in a number of ways, but the most common strategies are to group the documents by original source and then date, or such that collections are of similar size [44] (specific details of the formulation of these test collections are given later in Chapter 4).

One reason for researchers using the TREC corpus to create test collections, is that test queries are also supplied with the data. However, the primary reason is the provision of relevance judgements [56]. Most collection selection evaluations focus on testing how well an algorithm produces the optimal Relevance-Based Ranking baseline, described in Section 2.3.1; this baseline requires document relevance judgements.

However, some [10, 47] consider these synthetic TREC-based test collections to be unsuitable for collection selection evaluations. Real collections are large and diverse, whereas the TREC documents stem from a small set of sources, mostly news articles. Given that these documents are often grouped by source and date, the contents of the individual collections may be fairly heterogeneous.

D'Souza et al. [10] suggest that a grouping by author would help create topic-centric collections (an author tends to write about only a small set of subjects), but Shokouhi [47] disagrees: such a grouping is equally unsuitable. In addition, Shokouhi also comments that the synthetic TREC collections are often much smaller than real collections, and thus do not provide a realistic test environment.

An alternative to using artificial test collections compiled from the TREC corpus, is to use real collection data; the benefit being that the data accurately reflects the intended operational environment. While several collection selection evaluations have been conducted over real data, such as those of Gravano et al. [22] (databases accessible via Stanford University's information retrieval system), Yuwono and Lee [63] (real text collections), and more recently Thomas and Hawking [56] (public mailing list archives, calendar data, personal email, and a crawl of US government websites), the experiments have been modest. For example, each used only a handful of domain-specific collections; and as such producing a collection ranking may have been unchallenging for the algorithms tested.

Using real collections as test data presents several difficulties; the first being that suitable test queries must also be produced. Gravano et al. and Yuwono and Lee were fortunate to have access to the query logs associated with their test collections. However, Thomas and Hawking used programmatic methods to extract queries from key fields in their data. This raises a second complication in using real collections: the production of document relevance judgements.

As discussed above, many collection selection evaluations employ the Relevance-Based Ranking baseline, for which document relevance judgements are a necessity. For their generated test queries, Thomas and Hawking followed the TREC pooling method (see Section 2.2.4), by using a document ranking algorithm to determine the top n relevant documents, before manually assessing them for relevance. Depending on the number of queries and documents used, this manual process can be onerous.

As is evidenced by the myriad of collection selection algorithm evaluations, over numerous different data sets, the performance of a particular algorithm can vary significantly between different data sets [47]. As such, a robust evaluation should test algorithms over a variety of data sets. For example, Thomas and Hawking [56] use real collections to conduct an initial evaluation, and use synthetic collections built from TREC documents, to confirm trends observed on their own data.

Further details of test data sets used in collection selection algorithm evaluations are given in Section 2.4, alongside discussion of the algorithms they were used to test. Prior to this, in the following section we summarise the common techniques for evaluating collection selection algorithms, and their associated merits.

2.3.4 Summary of Collection Selection Evaluation Techniques

To date, many collection selection algorithm evaluations have been disparate, making comparison of different evaluation findings difficult. Most notably, several different performance measures and test data sets have been used. Some agreement may be observed in the evaluations: recent work tends to favour the $\mathcal{R}_n$, recall-like, performance measure; however this metric considers just one aspect of algorithm performance.

Many researchers tend to agree on the origin of their test data, with most preferring to use synthetic test collections built from the TREC document corpus. However, many different configurations of this data have been used, and there are questions over its suitability. The TREC corpus is drawn mostly from news articles, which are not representative of the diverse material present in real digital collections. In addition, the test collections created from these documents are small by comparison to real collections, and tend to be heterogeneous in content. Real collections may be expected to have some general topic or theme.

In contrast, some evaluations have used real collection data to test algorithms. However, these evaluations also have inadequacies. For example, most use only a handful of distinct collections. As such, the test environment may not be particularly challenging.

A robust evaluation should use a variety of appropriate performance measures. This allows an algorithm to be scrutinised from different angles, and therefore we can identify its strengths and weaknesses. Algorithms have been shown to perform differently on different configurations of test data, and so it is beneficial to use a range of test data sets, exhibiting different properties. For example, a mixture of synthetic TREC-based test data, and real collection data.

An additional short-coming of previous collection selection algorithm evaluations is a lack of rigour in analysing the results of the performance measures. Like document retrieval evaluations, the analysis of results is largely descriptive. Conclusions are generally drawn from observations made on graphs. While this descriptive analysis is itself an important and informative part of evaluating data, some deductive analysis would support and strengthen observations. For example, examining the exact degree of difference between algorithm-produced collection rankings and the baseline ranking, and the significance of the difference: quantification of the results. Naturally, we want to know if one algorithm performs better than another, but if both are so far away from the optimal, the fact that one is better is irrelevant: they are both ultimately not good enough for the task.

Given the context of common techniques for conducting evaluations of collection selection algorithms, in the following section we give an overview of the state of the art algorithms in this domain. We will later examine the applicability of a selection of these to the collection suggestion task (Chapters 6 and 7).

2.4 Collection Selection Algorithms

Recall from the previous sections that collection selection is concerned with identifying resources (such as digital libraries or search engines) that are likely to contain or index relevant documents. Traditionally, collection selection algorithms are evaluated based on their ability to rank collections in decreasing order of the number of relevant documents they ac-

tually contain (called the Relevance-Based Ranking), using a variety of performance measures.

A primary contribution of our research is to examine whether traditional collection selection algorithms may be used effectively for the alternative task of collection suggestion: identifying collections likely to contain a large number and concentration of relevant documents. As such, in this section we provide a discussion of the various algorithms designed for collection selection, and consider their performance over the collection selection task, as documented in the literature. In addition, we speculate on their suitability for collection suggestion.

2.4.1 GlOSS

Among the earliest work on collection selection, is that relating to the Glossary of Servers Server (GlOSS), undertaken by Gravano et al. [22, 23]. Here we discuss the two versions of GlOSS: bGlOSS, which is designed to evaluate boolean AND queries, and vGlOSS, which works over the vector space retrieval model.

bGlOSS

The initial implementation of GlOSS, namely bGlOSS [22], differs from subsequent collection selection algorithms in that it uses the boolean retrieval model, rather than the vector space model. As such, a collection must contain all query terms to be considered relevant.

The bGlOSS algorithm has very low storage and computational requirements: for each collection it requires only the document frequencies for each term, and the total number of documents within the collection. With this data, bGlOSS aims to predict the number of documents in each collection that contain all query terms. Thus, for a query q, of the form $t_1, \ldots, t_n$, the number of relevant documents in a collection is estimated by:

$$bGlOSS(q, c) = \frac{\prod_{i=1}^{n} freq(t_i, c)}{ColSize(c)^{n-1}},$$

where:

$freq(t_i, c)$ is the number of documents in collection c that contain term t_i, and

$ColSize(c)$ is the number of documents in collection c [22].

Given an estimate of the number of relevant documents in each collection, bGlOSS chooses only the collection(s) containing the highest number of relevant documents.

The evaluation of bGlOSS was carried out using a set of six real collections, and nearly 7000 user queries, extracted from the associated query logs. This evaluation differs from most later experiments, where the use of the TREC document corpus, split into synthetic collections, is preferred over real data.

One of the difficulties in using real collection data to evaluate algorithms lies in the production of an optimal collection ranking. As discussed in Section 2.3.1, this optimal is traditionally a Relevance-Based Ranking (RBR), where collections are ordered by the number

of relevant documents they contain. As such, RBR normally requires relevance judgements for each document.

Gravano et al. use RBR to evaluate bGlOSS, however as they are working within the boolean retrieval model, generating the RBR does not have the usual complexities. By simply executing each boolean query over all documents from the collections, they find the exact subset of relevant documents: those that contain all query terms. The optimal RBR can then be produced accordingly.

As described above, once the bGlOSS algorithm has estimated the number of relevant documents in each collection, the best collection(s) are chosen. As such, the evaluation by Gravano et al. does not check that the ranking as a whole is accurate (there is not a ranking as such produced), but that the 'best' collection(s) is correct. Results are reported as the percentage of queries where the algorithm produced the correct set of 'best' collections.

Two experiments were conducted: the first used only two collections, while the second used all six collections. In the first experiment, bGlOSS chose the best collection(s) for 91.75% of the queries; for the second the best collection(s) were correctly chosen for 82.06% of the queries.

Gravano et al. found these results encouraging, however we note here that the six collections were fairly discrete in their subject areas: psychology, geology/geophysics, business periodical literature, educational materials, engineering and physics/electrical engineering/computer science. As such, questions arise over the scalability of the algorithm: would similar performances be achieved when there are hundreds of collections to distinguish between, with several collections exhibiting similarity?

The bGlOSS algorithm has received little attention in large-scale evaluations. However, it has featured in additional small-scale evaluations, where the accuracy of the collection ranking as a whole was considered. The results were mixed: in Yuwono and Lee's experiment [63], bGlOSS was found to perform worse than the other algorithms tested. However, in Thomas and Hawking's experiment [56] (in the context of resource selection for personal metasearch), bGlOSS performed moderately well in relation to the other algorithms tested. Clearly there is scope to further test bGlOSS.

In the following section we consider the alternative specification of GlOSS, which operates over the vector space retrieval model.

2.4.2 vGlOSS

The bGlOSS implementation of GlOSS described above operates over the boolean retrieval model. Gravano et al. also developed an implementation of GlOSS to work within the vector space model: vGlOSS [23]. Whereas bGlOSS aims to identify the collections with the most documents that feature all query terms, vGlOSS assumes the user is interested in documents that are *similar* to their query. As such, an optimal collection ranking would put collections in decreasing order of the sum of their documents' similarity to the query.

Gravano et al. [23] specify two algorithms to estimate the total document-query similarity for a collection: *Max(l)* and *Sum(l)*. Here, l is a user defined threshold which states the minimum level of similarity to the query a document should have, to be considered interesting.

Max(l) and *Sum(l)* have low storage requirements, needing only two vectors of information: one containing the number of documents in each collection that feature the query terms, and one containing the sum of the weight of each term (for example, calculated by *tf-idf*) over all documents, within each collection. To estimate the total document-query similarity for a collection, *Max(l)* and *Sum(l)* make assumptions about the co-occurrence of query terms: *Max(l)* assumes query terms occur together in a collection's documents, while *Sum(l)* takes the opposite view. The reader is referred to Gravano et al. [23] for a comprehensive specification of the two estimators.

To facilitate the evaluation of the two estimators, *Max(l)* and *Sum(l)*, Gravano et al. present their notion of an optimal collection ranking, called *Ideal(l)*. This optimal ranking represents the principle of vGlOSS, that collections should be ranked according to the similarity of their documents to the user's query. *Ideal(l)* relies on a *goodness* metric, which calculates a score for the collections. The goodness of a collection c, for the query q, with a similarity threshold l is calculated as follows:

$$\begin{aligned} Goodness(l, q, c) &= \sum_{d \in Rank(l,q,c)} sim(q, d) \\ Rank(l, q, c) &= \{d \in c \mid sim(q, d) > l\} \\ sim(q, d) &= \sum_{i=1}^{n} q_i \cdot w_i \end{aligned}$$

where:

q_i is the weight of a given query term, in the query; and

w_i is the weight of a given query term, in the document (calculated by *tf-idf*) [23].

That is, the goodness of a collection is the sum of the similarities between its documents and the query. Document-query similarity is defined as the inner product of the weight of the terms in the query, and the weight of the terms in the document. A similarity score is used in the calculation only if it is above the threshold l.

In their work, Gravano et al. [23] use the $\mathcal{R}_n$ and $\mathcal{P}_n$ metrics (described in Section 2.3.2) to evaluate how closely *Max(l)* and *Sum(l)* estimate their *Ideal(l)* ranking; their findings suggest that the two algorithms are reasonable estimators of *Ideal(l)*. Indeed, when the threshold $l = 0$ is used (where we consider a document with any degree of similarity to the query to be relevant), *Max(l)* and *Sum(l)* are equivalent to *Ideal(l)*: they produce identical rankings.

Given this equivalence, many subsequent studies of vGlOSS, such as those of French et al. [18, 17, 44], Thomas and Hawking [56] and Yuwono and Lee [63], have used *Ideal(l)* to represent a vGlOSS implementation. Powell and French [44] indicate that from an implementation perspective, using *Ideal(l)* over *Max(l)* or *Sum(l)* is simpler, suggesting it may be easier to acquire term weights, than the vectors needed for the two estimators.

The subsequent evaluations of vGlOSS have primarily been undertaken by French et al. [18, 17, 44], and have investigated how closely *Ideal(l)* estimates the Relevance-Based Ranking (RBR) baseline. In addition, they also examined the performance of vGlOSS in relation to the Size-Based Ranking (SBR) lower bound baseline. The experiments were conducted over three test data sets constructed from discs 1–3 of the TREC document corpus

(see Sections 4.1.1, 4.1.2 and 4.1.3 in Chapter 4 for specific details of the composition of these data sets).

The empirical findings suggest that vGlOSS is not a good predictor of RBR: while it does rank good collections highly, those collections are not necessarily the best [18, 44]. In addition, using Spearman rank correlation, vGlOSS was shown to be highly correlated to SBR: it has a tendency to favour large collections [17, 44]. The reason for this lies in the composition of the algorithm: a collection score is calculated as the sum of its documents' similarity to the query. As such, a very large collection with many documents of marginal similarity may appear overall to be more useful than a collection with a few documents that are highly similar to the query [18].

A significant drawback of vGlOSS, is that it utilises term weights. These are generally calculated using *tf-idf*, but are local to each collection. As such, terms that are common in a collection (suggesting the collection may be *about* that concept), will receive a low weighting. However, another collection with fewer occurrences of that term will associate a larger weight to the term, as they have more discriminatory power. Therefore, a collection with many documents about the query term may be ranked lower than one with only a few documents. French et al. [18] note that for this reason, "similarity scores across heterogeneous document collections are essentially noncomparable."

The two collection ranking algorithms specified by Gravano et al. [23] are amongst the earliest work in the collection selection domain. In the following section we discuss the seminal work of Callan et al. [6].

2.4.3 CORI

One of the earliest algorithms for ranking collections is the Collection Retrieval Inference Network (CORI), attributed to Callan et al. [6]. CORI is based on the document ranking algorithm *tf-idf*: it treats each collection as a large document, and can be summarised as *df-icf*, where the document frequency *df* is the number of documents containing a query term, and the inverse collection frequency *icf* is the inverse of the number of collections containing the query term. The advantage of such an approach is that ranking collections has the same computational complexity as traditional document ranking, and similar storage requirements [6].

More specifically, CORI ranks collections based on their similarity to the query. Thus, the similarity of a collection c to the query q is calculated by:

$$
\begin{aligned}
CORI(q,c) &= \frac{\sum_{t \in q}(d_b + (1-d_b)\cdot T \cdot I)}{|q|} \\
T &= d_t + (1-d_t)\cdot \frac{df}{df+K} \\
I &= \frac{\log(\frac{(|C|+0.5)}{cf})}{\log(|C|+1)} \\
K &= k\cdot((1-b)+b\cdot cw/\overline{cw})
\end{aligned}
$$

where:

d_b is the minimum belief component when term t occurs in collection c, set as 0.4;

d_t is the minimum term frequency component when term t occurs in collection c, set as 0.4;

df is the number of documents in collection c containing term t;

$|C|$ is the number of collections;

cf is the number of collections containing term t;

k controls the magnitude of K. Callan et al. found $k = 200$ to be best for their test data;

b is a constant: varying its value between 0 and 1 increases the sensitivity of K to the size of the collection. Callan et al. found $b = 0.75$ to be best for their test data;

cw is the number of terms in collection c; and

$\overline{cw}$ is the mean cw of the collections being ranked [6].

Callan et al. evaluated the effectiveness of CORI for ranking collections with the TREC Volume 1 document collection. The documents were grouped by source and year, to give seven heterogeneous collections. The evaluation used 50 queries, formed from TREC topics 51 to 100.

The TREC document collection comes with relevance judgements, specifying for each topic, which documents are relevant. Thus, for each query, the collection ranking produced by the CORI algorithm is compared to an optimal ranking generated by ordering the collections by the number of relevant documents they contain (determined by the relevance judgements). The Mean-Squared Error (MSE) metric is used to determine the variation between a collection ranking generated by CORI, and the optimal ranking, and is averaged over the 50 queries to give a single value representing performance.

CORI showed promising performance in this test environment, achieving an average MSE of 1.4586 [6]. However, we note that the number of collections used was very small, and unrealistic of an operational environment. In addition, the heterogeneity of the collections suggests distinguishing between them is not difficult; ordinarily we may expect a few very similar collections to be present, and as such ranking these correctly is more difficult.

Despite the limitations of the test environment used to evaluate CORI, the algorithm has frequently been used as a benchmark for comparison, when testing new collection selection techniques (for example, Yuwono and Lee [63] and Srinivasa et al. [54] have evaluated their own algorithms against CORI). Several other evaluations have shown CORI to have the most accurate and consistent performance (for example those of French and Powell [16, 44]). More recently, evaluations such as those of Si and Callan [49] and Thomas and Shokouhi [57] have still found CORI to perform well.

However, D'Souza et al. [13] suggest that the use of CORI as a benchmark collection selection algorithm is not justified. This stems from the use of the k, b, d_b and d_t parameters. When developing CORI, Callan et al. experimented with different values of k and b, reporting values they found gave the best (MSE) results for their test data set.

D'Souza et al. tested CORI using 11 test data sets, each partitioning the TREC document collection in different ways. They explored the effect of different values for the parameters, in total testing 546 combinations of values for each test data set, and associated queries.

The findings of the experiment can be summarised as follows: "it appears that parameters cannot reliably be chosen for CORI: not only do the optimal choices vary between data sets, but they also vary between query types and indeed, vary wildly within query sets." [13] Most notably, when testing the standard CORI values of $k = 200$ and $b = 0.75$, performance was poor. This is of concern, as these parameters are often used in work where CORI is used as a benchmark.

In the following section we consider a series of algorithms that utilise a variety of lexicon and collection statistics, to rank collections.

2.4.4 Lexicon Inspection

Zobel's work on collection selection techniques [66] examines four algorithms that use various term and collection statistics to produce a collection score, namely: Cosine Measure, Inner Product, Skew and Highest-available Similarity. Like the CORI algorithm discussed in the previous section, these algorithms are inspired by traditional document retrieval algorithms. We present the four algorithms below, before discussing their respective performances.

Cosine Measure: A similarity measure originally found to be effective for document retrieval. The algorithm has been modified for collection ranking, and for this purpose has been defined as follows:

$$\begin{aligned} C(q,c) &= \frac{\sum_{t \in q} w_{q,t} \cdot w_{c,t}}{W_c} \\ w_{q,t} &= w_t \cdot \log(f_{q,t} + 1) \\ w_{c,t} &= \log(f_{c,t} + 1) \\ w_t &= \log(\frac{N}{f_t} + 1) \\ W_c &= \sqrt{\sum_{t \in c} w_{c,t}^2} \end{aligned}$$

where:

$f_{q,t}$ is the number of occurrences of the term t in the query q;

$f_{c,t}$ is the number of documents in collection c containing term t;

N is the number of documents (across all collections); and

f_t is the number of documents (across all collections) containing term t [66].

Inner Product: This algorithm calculates the similarity between a query and a collection by summing the products of query term and collection weights; Inner Product is similar to vGlOSS, discussed in Section 2.4.2, and is defined as follows:

$$\begin{aligned} I(q,c) &= \sum_{t \in q} w_{q,t} \cdot w_{c,t} \\ w_{q,t} &= w_t \cdot \log(f_{q,t} + 1) \\ w_{c,t} &= w_t \cdot \log(f_{c,t} + 1) \\ w_t &= \log(\frac{N}{f_t} + 1) \end{aligned}$$

where:

$f_{q,t}$ is the number of occurrences of the term t in the query q;

$f_{c,t}$ is the number of documents in collection c containing term t;

N is the number of documents (across all collections); and

f_t is the number of documents (across all collections) containing term t [66].

Skew: This algorithm is designed to estimate whether the query terms are uncommonly frequent in the collection, therefore it may indicate whether a collection is *about* the query. For a collection c and a query q, Skew is defined as follows:

$$\begin{aligned} S(q,c) &= \sum_{t \in q} \frac{f_{c,t}}{f_t} \cdot f_{q,t} \cdot w_t \\ w_t &= \log(\frac{N}{f_t} + 1) \end{aligned}$$

where:

$f_{q,t}$ is the number of occurrences of the term t in the query q;

$f_{c,t}$ is the number of documents in collection c containing term t;

N is the number of documents (across all collections); and

f_t is the number of documents (across all collections) containing term t [66].

Highest-available Similarity: Here, the Cosine Measure is used to estimate the highest-available similarity for a typical length document in a collection. The algorithm is defined as follows:

$$
\begin{aligned}
H(q,c) &= \frac{\sum_{t \in q} w_{q,t} \cdot w_{c,t}}{W_c} \\
w_{q,t} &= w_t \cdot \log(f_{q,t} + 1) \\
w_{c,t} &= w_t \cdot \log(F_{c,t} + 1) \\
w_t &= \log(\frac{N}{f_t} + 1) \\
W_c &= \sqrt{\frac{\sum_{t \in c} F_{c,t}}{N_c}}
\end{aligned}
$$

where:

$f_{q,t}$ is the number of occurrences of the term t in the query q;

$f_{c,t}$ is the number of documents in collection c containing term t;

N is the number of documents (across all collections);

f_t is the number of documents (across all collections) containing term t;

$F_{c,t}$ is the number of occurrences of term t in collection c; and

N_c is the number of documents in collection c [66].

In evaluating these four algorithms, Zobel used two test data sets built from the TREC corpus. The first data set used disc 2 of the TREC data, with documents partitioned by source and date to form 43 collections of between 1600 and 7500 documents each. The query topics 51 to 150 were utilised [66]. This first data set was used during the development of the algorithms; as such, a second data set was used to independently confirm findings.

The second data set was built from the TREC disc 3: 91 collections were formed by breaking the data at random points. These collections are more varied in size than the previous: between 14 and almost 23000 documents, with an average of 5000. Query topics 202 to 250 were used on this occasion [66].

In his experiments, Zobel used a Relevance-Based Ranking to represent the optimal collection ranking (collections are ordered by the number of relevant documents they contain, where document relevance is determined by TREC relevance judgements). To evaluate performance, Zobel counts the number of relevant documents occurring in the highest n collections of each algorithm's collection ranking, and plots these values on a graph. This is similar to the $\mathcal{R}_n$ metric discussed in Section 2.3.2, however it specifically measures recall, rather than merit: "it measures how many of the relevant documents the user will have an opportunity to see" [66]. Zobel also used the fixed, Sized-Based Ranking (SBR), as a lower bound on performance.

On the first data set described above, results showed all four algorithms to perform better than the SBR. Highest-available Similarity and Inner Product performed best and similarly, closely followed by Skew. The Cosine Measure was clearly the weakest of the four.

The results on the second data set were slightly different, with Inner Product performing best, followed by Skew. Highest-available Similarity was marginally better than the Cosine Measure, which tracked SBR. Zobel concludes that while some of these simple similarity measures exhibit satisfactory performance, there is still room for improvement.

There have been very few independent evaluations of these four algorithms, or comparisons with other collection selection algorithms. Indeed, other research featuring these algorithms has had an alternative agenda (rather than simply investigating collection selection algorithm performance).

For example, D'Souza et al. [12] compare Inner Product and CVV (see Section 2.4.5) to a variety of algorithms that use only a partial term index to describe collections (referred to as n-term indexing; see Section 2.4.7). Using the same test environment employed by Zobel [66] (data sets, queries, performance measure and baselines), they find Inner Product to be the most effective algorithm of those tested. In addition, the performance of Inner Product is identical to that shown in Zobel's original tests, therefore confirming his findings. However, the primary conclusions of the work is that lexicon based algorithms outperform those using a partial term index [12].

D'Souza et al. [10] also conducted further experiments, this time looking at the performances of Inner Product, Highest-available Similarity and Skew, in addition to CORI, and several n-term indexing algorithms. However, the primary aim of this work was to investigate the effect of different formulations of test data set on the performances of the algorithms. To this end, D'Souza et al. created a *managed* test data set, where the documents from TREC discs 1 and 2 were grouped into collections, such that they have some common topic. One strategy for this was to group documents by author. For comparison, additional test data sets with documents ordered by source and date were created.

Using the recall performance measure employed by Zobel, and the RBR and SBR baselines, the results of D'Souza et al. showed the Highest-available Similarity algorithm to outperform Inner Product. Skew was shown to be the weakest of Zobel's algorithms. However, all three tended to perform better than CORI.

The most important observation of this work however was the effect of the different test data formulations: in the data sets where documents were split by source and date, the algorithm performances were very close, making it difficult to distinguish between the algorithms. However, on the managed test data sets, differences were more pronounced, with a wider range of algorithm effectiveness observed.

In addition to the work of D'Souza et al., Thomas and Hawking [56] investigated the suitability of Inner Product for use in a personal metasearch environment, along with bGlOSS, vGlOSS, CORI and CVV. Personal metasearch includes smaller collections such as calendar, email and mailing lists, in addition to web collections [56]. Inner Product was found (using the $\mathcal{R}_n$ metric) to perform poorly in this application, with CORI and bGlOSS performing well.

Thomas and Hawking also conducted experiments over more traditional TREC corpus based data sets. Performances were varied over the different data set configurations; on

some data sets Inner Product performed better than CORI and bGlOSS, while on others it was worse. vGlOSS was consistently poor.

It is evidenced here that algorithm performance can vary, depending on the context in which they are tested. We see further evidence of this in the following sections, as we continue to discuss developments in collection selection algorithms.

2.4.5 CVV

The Cue Validity Variance (CVV) algorithm was developed by Yuwono and Lee [63], as part of the Distributed WWW Index Servers and Search Engine (D-WISE) project, which aimed to develop an Internet resource discovery system. Like bGlOSS (discussed in Section 2.4.1), CVV uses only document frequency data to produce a collection ranking. Rather than predicting the number of relevant documents in a collection, or estimating the relevance of a collection's documents, CVV uses the document frequencies to measure the distribution of query terms in the collections, and the usefulness of query terms for distinguishing between collections [63, 12].

Using CVV, the *goodness* score for a collection c with respect to a query q is calculated as follows:

$$
\begin{aligned}
G(q,c) &= \sum_{t \in q} CVV_t \cdot df_{c,t} \\
CVV_t &= \frac{\sum_{c=1}^{|C|} (CV_{c,t} - \overline{CV_t})^2}{|C|} \\
CV_{c,t} &= \frac{\frac{df_{c,t}}{N_c}}{\frac{df_{c,t}}{N_c} + \frac{\sum_{k=1 \& k \neq c}^{|C|} df_{k,t}}{\sum_{k=1 \& k \neq c}^{|C|} N_k}} \\
\overline{CV_t} &= \frac{\sum_{c=1}^{|C|} CV_{c,t}}{|C|}
\end{aligned}
$$

where:

$df_{c,t}$ is the number of documents in collection c containing term t;

$|C|$ is the number of collections; and

N_c is the number of documents in collection c [63].

In these calculations $CV_{c,t}$ is the *cue validity* of term t in collection c; this measures the degree to which term t distinguishes documents in collection c from those in other collections. It follows that $\overline{CV_t}$ is the average cue validity of term t over all collections. The *CVV* component is the population variance of the cue validity, measuring the skewness of distribution of term t across the collections. As such, the larger the variance, the more useful the term for distinguishing between the collections [63].

Yuwono and Lee evaluated the CVV algorithm using their own 'accuracy' metric. To calculate the accuracy of a collection ranking for a given query, the top n documents within

each collection are determined, using *tf-idf*. The *tf-idf* scores for the documents are added together to give an 'actual goodness' score for each collection. The collection scores generated by a collection selection algorithm are 'estimated goodness' scores. The accuracy of the algorithm is calculated as the cosine similarity between the actual goodness and the estimated goodness over all collections [63].

In their evaluation, Yuwono and Lee compared the performance of CVV, in terms of accuracy, to bGlOSS, vGlOSS and CORI. The accuracy scores for each algorithm were averaged over all queries and plotted on a graph.

For their test data sets, Yuwono and Lee used four real collections, and queries from their associated query logs. The collections comprised a total of 7097 documents, which were used to form five different test data sets. For example, one data set used documents split into distinct groups of related documents, while in another, documents were randomly distributed, and therefore could be diverse in content.

In this test environment, CVV was shown to outperform bGlOSS, vGlOSS and CORI: it had accuracy scores that were frequently higher than those achieved by the algorithms. However, these findings are contrasted in subsequent, independent research, conducted using the established $\mathcal{R}_n$ performance measure.

For example, in their evaluation of the CORI, vGlOSS and CVV algorithms, using collections built from the TREC corpus, Powell and French [44] found the CVV algorithm to frequently perform the worst. In addition, its performance varied greatly (more so than CORI), on different test data sets. As discussed in Section 2.4.4, CVV was also found to perform worse than the Inner Product algorithm. In addition, studies by Sogrine and Patel [52] and Thomas and Hawking [56] found CVV to be inferior to CORI and the GlOSS algorithms, using their respective test environments.

In the following section we present another collection selection algorithm that utilises only document frequency data to rank collections.

2.4.6 DFPROP

So far we have examined two algorithms, bGlOSS (Section 2.4.1) and CVV (Section 2.4.5), that use only document frequency statistics to produce a score for a given collection. Another such algorithm is DFPROP, developed by Srinivasa et al. [54].

In their work, Srinivasa et al. describe the development of DFPROP, which is driven by a series of experiments. Their starting point is two simple metrics, *dfproportion* and *ctfproportion*, defined as follows:[1]

$$dfproportion_{c,t} = \frac{df_{c,t}}{\sum_{c=1}^{|C|} df_{c,t}}$$

$$ctfproportion_{c,t} = \frac{ctf_{c,t}}{\sum_{c=1}^{|C|} ctf_{c,t}}$$

where:

[1] We change the notation from that originally given by Srinivasa et al. [54], to follow notation previously used in this chapter.

$df_{c,t}$ is the number of documents in collection c containing term t;

$ctf_{c,t}$ is the number of occurrences of term t in collection c; and

$|C|$ is the number of collections [54].

As such, these metrics measure the fraction of documents containing a query term each collection has (*dfproportion*), and the fraction of all query term occurrences each collection has (*ctfproportion*) [54]. The larger the values, the more related to the query term the collection is.

For their initial algorithms, Srinivasa et al. combine these metrics to give two algorithms: SUM and PROD. Given a query q, the score for a collection c may be calculated as follows:

$$SUM(q,c) = \sum_{t \in q} f_{q,t} \cdot (dfproportion_{c,t} + ctfproportion_{c,t})$$

$$PROD(q,c) = \sum_{t \in q} f_{q,t} \cdot (dfproportion_{c,t} \cdot ctfproportion_{c,t})$$

where:

$f_{q,t}$ is the number of occurrences of term t in the query q [54].

These algorithms were evaluated using three test data sets formulated from the TREC corpus, specified by Powell [43]. The RBR and SBR baselines were used, and comparisons were made with the performances of CORI and vGlOSS, using the $\mathcal{R}_n$, $\hat{\mathcal{R}}_n$ and $\mathcal{P}_n$ performance measures. The empirical findings showed SUM and PROD to be superior to vGlOSS, with SUM being the stronger of the two. CORI performed best overall.

To investigate the effects of the *dfproportion* and *ctfproportion* components, two new algorithms were specified; DFPROP and CTFPROP use each of the components in isolation:

$$DFPROP(q,c) = \sum_{t \in q} f_{q,t} \cdot dfproportion_{c,t}$$

$$CTFPROP(q,c) = \sum_{t \in q} f_{q,t} \cdot ctfproportion_{c,t}$$

where:

$f_{q,t}$ is the number of occurrences of term t in the query q [54].

Over the same test environments as before, these two algorithms were tested against CORI, SUM and PROD. CORI again came out on top, however DFPROP was shown to be better than SUM, while CTFPROP and PROD produced similar results. As such, the *dfproportion* component is the more effective of the two.

A further experiment modified the SUM algorithm, by weighting the two components: *dfproportion* with 0.8, and *ctfproportion* with 0.2. The results were mixed: at some rank positions the weighted algorithm performed better than DFPROP, while at others DFPROP was superior.

Therefore, Srinivasa et al. conclude that DFPROP is the most effective of their algorithm formulations, however it is not able to match the performance of CORI. No independent evaluations of these algorithms have been carried out to date. However, the algorithms may have applicability to the collection suggestion task.

2.4.7 n-Term Indexing Based Techniques

The majority of the collection selection algorithms we have examined thus far have used data such as term and document frequencies, and general collection statistics to generate a score for a collection, with respect to a query. As such, a collection is often treated as if it were a large document, and collection ranking follows a similar approach to traditional document ranking.

D'Souza et al. [12] investigate a range of collection ranking techniques that use the rankings of the documents (with respect to the query) within a collection, to produce a score for the collection. That is, the documents in each collection are ranked (together) for the query, and the document rankings are used to calculate a collection score. D'Souza et al. specify five methods for calculating collection scores in this way:

Naive: The score for a collection c is equal to the similarity score of its highest ranked document:

$$naive(q,c) = \max_{d \in c} sim(q,d)$$

Sumsim: The score for a collection c is equal to the sum of its documents' similarity scores:

$$sumsim(q,c) = \sum_{d \in c} sim(q,d)$$

Invrank: The score for a collection c is equal to the sum of the inverse document ordinals:

$$invrank(q,c) = \sum_{d \in c} \frac{1}{r_d + K},$$

where:

r_d is the rank position of document d; and

K is a constant, set to 10 [12].

Sumsimsqr: The score for a collection c is equal to the sum of the squared document similarity scores:

$$sumsimsqr(q,c) = \sum_{d \in c} sim(q,d)^2$$

Simdivrank: The score for a collection c is equal to the sum of the similarity score of each document, divided by its rank position:

$$simdivrank(q,c) = \sum_{d \in c} \frac{sim(q,d)}{r_d},$$

where:

r_d is the rank position of document d [12].

In the above algorithms, $sim(q,c)$ is the similarity of a document d and the query q; however, it is not specified by D'Souza et al. specifically how this similarity is calculated.

As an additional element to their research, D'Souza et al. produce the document rankings using a *partial term index*, whereby a sample of n terms is used to represent each document in a collection. This allows additional information and statistics for each term to be stored, using no more space than that required for a complete index [11]. This concept, also referred to as n-Term Indexing was first investigated by D'Souza and Thom [11]. They suggested four methods for choosing a sample of n terms to index:

1. Take the first n unique terms in each document;
2. Take the rarest n unique terms in each document;
3. Take n/s unique terms from each of s structural components (such as title, abstract and first paragraph) of each document; and
4. Take n/s rarest terms from each of s structural components of each document [11].

D'Souza and Thom evaluated only the 'first n unique terms' approach, by comparing precision and recall scores achieved, against those of a full term index. Using the documents on the TREC disc 3 corpus, and queries 202 to 205, they found that retrieval effectiveness over the full index was superior to the partial index. However, they speculated that selection of better quality terms (such as those from the structural components) may lead to improvements in retrieval performance [11]. Despite these findings, the 'first n unique terms' approach was used to build the partial term index used in the experiments of D'Souza et al. [12].

D'Souza et al. [12] first compare their five collection ranking algorithms against each other. They used the TREC disc 3 based data set used by Zobel [66]. The recall-based evaluation metric used by Zobel is also utilised, as are the RBR and SBR baselines. Of the five algorithms, *simdivrank* was found to have performed best in this test environment.

To investigate how the partial index document rank based collection selection algorithms compare to other, full index collection statistic based approaches, a further experiment tested *simdivrank* against Inner Product and CVV. The results showed Inner Product to perform best, with *simdivrank* and CVV exhibiting similar results.

These findings suggest that the partial indexing document rank based approaches are inferior to the more traditional techniques that use collection statistics derived from full term indexes. However, only one strategy for building the partial term index was used; there

has been no further investigation of the other proposed techniques. It is possible that selection of stronger index terms would improve the performance of the document rank based algorithms (the 'first n unique terms' technique was shown by D'Souza and Thom [11] to be inferior to a full term index).

A notable drawback to basing collection scores on document rankings is the fact you must rank every document, across all collections, for each query. Regardless of whether the underlying term index is a full or partial one, this will be an expensive task overall; the collection statistics approaches of algorithms such as CORI, Inner Product and CVV are much less computationally expensive.

2.4.8 Decision-Theoretic Framework

The collection selection algorithms we have discussed so far use statistical data (such as term and document frequencies) derived from the content of the collections to produce a collection ranking.

The Decision-Theoretic Framework (DTF) [19, 20, 21, 39] takes a different approach to collection selection: given a specific number n of relevant documents a user wants to retrieve, DTF computes an optimum collection selection strategy, such that the costs of retrieving documents from multiple collections are minimised [20].

Cost can be defined in a variety of ways, such as: charges per document, connection time and computation time [19, 21]. The primary cost discussed in the relevant literature is that of retrieval quality; that is, the cost of a user being shown irrelevant documents.

The user can specify their own cost model, by specifying the importance of different costs. For example, they may be willing to pay to ensure high quality results, or alternatively they may want results returned very quickly, at the expense of retrieval quality. The costs associated with retrieving documents from a particular collection can be summed, to give an overall cost for that collection [39].

Specifically, DTF is specified as follows:

$$DTF(n, q) = min_{|s|=n} \sum_{i=1}^{|C|} C_i(s_i, q),$$

where:

n is the number of documents the user wants to retrieve, for the query q;

$|C|$ is the number of collections;

s is a vector $(s_1, \ldots, s_{|C|})$ where s_i is the number of documents to retrieve from collection c_i;

$|s|$ is defined as $\sum_{i=1}^{|C|} s_i$, which is equal to n; and

$C_i(s_i, q)$ is the total cost associated with retrieving s_i documents from collection c_i, for the query q [39].

The $C_i(s_i, q)$ component is the total cost associated with retrieving documents from a collection. The primary example in the literature is that of the cost of retrieval quality (though Nottelmann and Fuhr [41] also specify time costs). This may be defined as follows:

$$C_i^{rel}(s_i, q) = r_i(s_i, q) \cdot C^+ + [s_i - r_i(s_i, q)] \cdot C^-,$$

where:

$r_i(s_i, q)$ is the number of relevant documents in the result set when s_i documents are retrieved from collection c_i;

C^+ is the cost (for example, 0) of viewing a relevant document; and

C^- is the cost (for example, 1) of viewing an irrelevant document [39].

One advantage of DTF over other collection selection algorithms is that the optimal solution indicates how many collections to forward the query to, and how many documents to retrieve from each, in order to ensure costs are kept to a minimum. The more traditional collection selection algorithms only give an ordering of collections; the decision of how many of those to query and how many documents to retrieve is an additional step, requiring further algorithms.

There has been limited evaluation of DTF, and certainly no independent testing. Nottelmann and Fuhr [40] evaluated DTF against CORI on a TREC-based test data set of 100 collections, derived by grouping documents by source and date. Due to the nature of DTF, the evaluation examined the overall retrieval quality of the final list of returned documents (rather than on the quality of the collection ranking). A range of short and long TREC queries were used; the findings showed DTF to be comparable to CORI on long queries, however it suffered on short queries.

The issue of cost consideration in collection selection is an important and interesting one; however, to implement the framework it is necessary to have details of each cost, for each collection: an arduous task.

The essence of DTF is highly relevant to our collection suggestion problem: costs such as document charges, and impact and reputation of a collection could be important to the user. Despite this however, it is unclear whether DTF could be applied to collection suggestion: it is very much embedded in document retrieval, as collections are chosen based on their costs for retrieving a given number of documents. A collection suggestion search service will not retrieve documents for the user, but rather direct them to a few collections they may wish to browse themselves.

2.4.9 Collection Selection in Uncooperative Environments

The collection selection algorithms we have discussed thus far are intended for use in *cooperative* environments, where it is assumed we have access to collection statistics: such as term and document frequencies, and collection sizes. Such statistics may have been exported by the collections, or are derived from access to the full collection contents.

More recent collection selection research has focused on algorithms that can operate in *uncooperative* environments, where the collection statistics are not readily available: they are not exported by the collections. In an uncooperative environment, the algorithms must generate a collection ranking based on only a small sample of documents from each collection.

A representative document sample is gained by issuing queries to the collections, and retrieving the resulting documents [48]. These documents are used to produce an estimate of term and document frequencies within a collection, and the total size of a collection. In effect, each document in the sample represents some number of documents in the collection from which it originates.

Collection selection algorithms designed for use in uncooperative environments are often based on the same underlying principle, but with differences in implementation. We provide a brief summary of four collection selection algorithms for uncooperative environments, followed by a discussion of their effectiveness.

Relevant Document Distribution Estimation (ReDDE)

To generate a collection ranking for a given query, ReDDE [49] first requires the sample documents from all collections to be ranked for the query. The first n ranked documents are used to count (estimate) the number of relevant documents in each collection. The collections are then ranked by the percentage of the relevant documents they are estimated to contain.

In addition to the uncooperative environment, ReDDE may also be used in cooperative environments. Here, calculations are performed on accurate figures rather than estimates. However, as a ranking of documents is required, execution in an environment where access to all documents is available would be time consuming.

Central-Rank-Based Collection Selection (CRCS)

Like ReDDE, the CRCS [47] algorithm first ranks sampled documents for the given query. However, rather than count the number of relevant documents within each collection, CRCS uses the ranks of the documents to calculate a weighting for each collection. Thus, a collection with highly relevant documents will feature highly in the collection ranking.

The CRCS algorithm can also be applied to a cooperative setting. However like ReDDE it would suffer from cost and performance issues: ranking every document for a given query, prior to ordering the collections, is time intensive when we have hundreds of large collections.

Unified Utility Maximization (UUM)

The UUM algorithm [50] can be used to support two variations of collection selection: a high-recall focused task, where the goal is to find collections with as many relevant documents as possible, and a high-precision task, where the aim is to identify collections containing highly relevant documents.

The implementation of UUM is complex, first using training queries (for which relevance judgements are required) to learn how to infer the probabilities of relevance of all (mostly unseen) documents across the collections. Collections are ranked using the inferred probabilities of document relevance.

Scoring Scaled Samples (SUSHI)

The SUSHI algorithm [57] uses sampled documents to extrapolate scores for unseen documents within the collections. As for ReDDE and CRCS, SUSHI first requires the sampled documents to be ranked for the query. The ranks of the sample documents are then adjusted, to reflect the property that each sampled document is representative of several documents in its original collection.

Following this, the scores of documents not sampled are estimated by fitting a curve to the rank-adjusted documents: linear, logarithmic and exponential curves are tried at query time, with the best fit chosen. Given the interpolated document scores, collections can be ordered by the sum of the document scores.

Thus, the algorithm is optimised for precision: favouring collections with highly relevant documents. However, the algorithm may also be optimised for recall, where collections are ordered by the number of relevant documents each is estimated to contain.

Algorithm Performance

These four algorithms have been evaluated over several, varied test data sets, though primarily these have stemmed from the original TREC document corpus.

Of the algorithms, ReDDE is perhaps considered the benchmark for comparison: in their initial evaluation, Si and Callan [49] found it to perform at least as good as CORI (using the $\mathcal{R}_n$ metric). ReDDE and CORI have been used as benchmarks in the evaluations of the UUM, CRCS and SUSHI algorithms.

Si and Callan's [50] evaluation of UUM found it to always outperform CORI, and on average to perform better than ReDDE. However, possibly due to the complexity of the algorithm, no comparison between UUM and the other two algorithms has been made.

The SUSHI algorithm appears to be the weakest of the four, often exhibiting performance below that of CORI. CRCS showed similar performance to ReDDE, in evaluations in which it features [47, 57].

Summary of Collection Selection Algorithms for Uncooperative Environments

The four algorithms described above have been designed for use in uncooperative environments, where collection statistics are not readily available. As such, collection orderings are produced on the basis of a sample of documents, taken from each collection. The implementations of the algorithm vary, but in essence the techniques used aim to either estimate the number of relevant documents in each collection (recall focused), or estimate which collections feature the highly relevant documents (precision focused).

The two strongest algorithms, ReDDE and CRCS, may also be applied to a cooperative environment, where document statistics are available. However, execution of these algo-

rithms in such an environment would be expensive. It is essentially a brute force approach, ranking the thousands or millions of documents first, before calculating which collections contain the most or best documents.

2.5 Query Performance Predictors

In this section we present an additional class of algorithm that may be applicable to the collection suggestion problem: namely, *query performance predictors.* We give an overview of this domain in the following section, followed by details of evaluation techniques and discussion of the algorithms.

2.5.1 Overview of Query Performance Prediction

The aim of query performance prediction (also know as query difficulty estimation [26]) is to estimate the retrieval effectiveness of a query, on the search system to which it was submitted [26]. That is, the quality of the results the system returns for the query, for example in terms of precision [61, 27]. If a query is estimated to perform poorly at the search engine, attempts can be made to improve the retrieval performance: for example, by performing query term expansion, requesting that the user reformulate the query, or submitting the query to specialist collections [27, 64].

In the collection suggestion scenario, query performance predictors could be used to rank collections, based on the estimated quality of the returned results. Indeed, it has previously been suggested that query performance prediction techniques could be applied to both the collection selection and results merging problems associated with federated search [61, 65, 62].

There are two categories of query performance predictors, namely, *pre-retrieval* and *post-retrieval.* Pre-retrieval predictors use query term and collection statistics to estimate the performance of the query, *before* the search is executed. Conversely, post-retrieval predictors operate on the returned result list, and thus more accurately measure the performance of the query. However, if the results are found to be poor by post-retrieval prediction, the search must be performed again with the reformulated query [27].

For application to collection suggestion, post-retrieval predictors are not suitable: if we were to evaluate a query against all collections and their documents, we would waste computation resources (similar to the problems associated with federated search), particularly if we were to amend and resubmit the query. As such, the statistic-based pre-retrieval predictors are favourable for collection suggestion; it is algorithms of this type that we evaluate for suitability for collection suggestion.

Hauff [26] discusses and evaluates a great variety of query performance prediction algorithms. Here, we discuss a sample of algorithms that may be applicable to collection suggestion and choose a smaller number to evaluate with respect to this task. We cover two groups of predictor: those that operate using term and collection statistics, and those that measure query-collection similarity and query term distribution. Prior to discussing these algorithms, in the following section we give a brief overview of the evaluation practices for query performance prediction.

2.5.2 Overview of Query Performance Prediction Evaluation Techniques

For query performance prediction, algorithms are evaluated based on their ability to produce scores (for a search system or collection) that correlate to *actual* retrieval quality at that resource [28].

As such, the evaluation strategy in this domain is as follows. For the test queries and a set of documents, document ranking algorithms such as *tf-idf* and BM25 are used to rank the documents. This represents the actual performance of a search system. This performance is quantified using traditional document retrieval performance measures, such as average precision, or precision at document cut-off values (P@10, for example) [28, 64].

Given a score (estimate of retrieval performance) for the test collection, produced by a query performance predictor, the correlation between the query performance predictor and the measure of actual retrieval effectiveness is calculated [28, 64]. That is, we are examining whether the query performance prediction score accurately estimates actual retrieval effectiveness.

Given this overview of evaluation practices for query performance prediction, in the following sections we discuss the state of the art algorithms in this domain, and their reported effectiveness.

2.5.3 Term and Collection Statistic-Based Predictors

In their work on pre-retrieval query performance predictors, He and Ounis [28] present and test six algorithms they believe may be useful for indicting the retrieval performance of a query. The algorithms, which use basic term and collection data, are specified as follows:

Query Length: The query length (*ql*) is the number of non-stop words in the query [28].

Distribution of Informative Amount (1): The Distribution of Informative Amount ($\gamma 1$) for a given query is represented as:

$$\begin{aligned} \gamma 1 &= \sigma_{idf} \\ idf(t) &= \frac{\log_2(\frac{(N+0.5)}{N_t})}{\log_2(N+1)} \end{aligned}$$

where:

σ_{idf} is the standard deviation of the *inverse document frequency* (*idf*) of the query terms;

N_t is the number of documents the query term t appears in; and

N is the number of documents in the whole collection [28].

Distribution of Informative Amount (2): A second definition for the Distribution of Information Amount ($\gamma 2$) may be specified as follows:

$$\gamma 2 = \frac{idf_{\max}}{idf_{\min}}$$
$$idf(t) = \frac{\log_2(\frac{(N+0.5)}{N_t})}{\log_2(N+1)}$$

where:

$idf_{\max}$ is the maximum idf among the terms in the query q;

$idf_{\min}$ is the minimum idf among the terms in the query q;

N_t is the number of documents the query term t appears in; and

N is the number of documents in the whole collection [28].

Query Scope: This predictor measures the generality of a query, and is defined as follows:

$$\omega = -\log(\frac{n_q}{N}),$$

where:

n_q is the number of documents containing at least one of the query terms; and

N is the number of documents in the whole collection [28].

Simplified Clarity Score: The clarity of a query is inversely proportional to its ambiguity. The Simplified Clarity Score (SCS) for a query is given by:

$$SCS = \sum_q P_{ml}(w|q) \cdot \log_2 \frac{P_{ml}(w|q)}{P_{coll}(w)}$$
$$P_{ml}(w|q) = \frac{qtf}{ql}$$
$$P_{coll}(w) = \frac{tf_{coll}}{token_{coll}}$$

where:

qtf is the number of occurrences of a query term in the query;

ql is the number of terms in the query;

tf_{coll} is the number of occurrences of a query term in the whole collection; and

$token_{coll}$ is the number of terms in the collection [28].

Average Inverse Collection Term Frequency: According to Kwok [34], the *inverse collection term frequency* can be seen as a replacement for *idf*. This predictor uses the average inverse collection term frequency of the query terms (AvICTF) to infer query performance, and is given by:

$$AvICTF = \frac{\log_2 \prod_q (\frac{token_{coll}}{tf_{coll}})}{ql},$$

where:

ql is the number of terms in the query;

tf_{coll} is the number of occurrences of a query term in the whole collection; and

$token_{coll}$ is the number of terms in the collection [28].

The effectiveness of these predictors was evaluated by He and Ounis, using data from discs 4 and 5 of the TREC document corpus. The documents within were combined into a single collection. A total of 249 associated test queries were executed, divided into three types: *short queries* (2.62 terms on average), *normal queries* (7.94 terms on average) and *long queries* (21.75 terms on average) [28]. As such, the effect of query length on the performance of the predictors could be investigated.

To assess the performance of the predictors, He and Ounis measured the correlation between the scores produced by the predictors, and the average precision of the actual retrieval effectiveness, which was simulated using both BM25 and PL2. Average precision scores from both document ranking algorithms yielded similar correlation values; we summarise the findings as follows:

- Query length (ql) as a predictor is weakly correlated with average precision.
- Distribution of Informative Amount (1) ($\gamma 1$) has strong correlation with average precision in all test cases. It is most effective over normal and long queries.
- Distribution of Informative Amount (2) ($\gamma 2$) is not as effective as $\gamma 1$; the correlation with average precision is not statistically significant in all test cases.
- Query Scope (ω) has significant correlation with average precision for short and normal length queries. However, the effectiveness of the predictor decreases as query length increases.
- Simplified Clarity Score (SCS) has strong correlation with average precision in all test cases, however there is some decrease in effectiveness as query length increases.
- Average Inverse Collection Term Frequency (AvICTF) shows similar performance to SCS, and is the most effective predictor over short queries [28].

As such, He and Ounis [28] state that AvICTF and SCS were the most effective predictors for short queries, and $\gamma 1$, AvICTF and SCS were the most effective for normal to long queries. Therefore, these three predictors may be suitable for practical applications [28]. In addition, He and Ounis note that, although query length can affect retrieval performance (as observed in the findings above), query length on its own is not an effective performance predictor.

In the following section we consider additional query performance predictors, that consider query-collection similarity and distribution of query terms.

2.5.4 Similarity and Variability Evidence-Based Predictors

Zhao et al. [64] specify and investigate pre-retrieval query performance predictors that may be classified into two groups: algorithms that measure the *similarity* of a query to the collection as a whole, and algorithms that measure the *distribution* of query terms among the documents in the collection. Both of these strategies have been used in the collection selection domain, and thus they may be of interest for the collection suggestion problem.

We discuss the similarity- and distribution-based algorithms in the following sections, followed by an overview of Zhao et al.'s empirical findings.

Similarity-Based Measures

There are three similarity-based measures, defined by Zhao et al. as follows:

SCQ: Given a query q consisting of terms $t_1, ..., t_n$, the similarity score between the collection and query may be defined as:

$$SCQ = \sum_{t \in q} (1 + \ln(f_{c,t})) \cdot \ln\left(1 + \frac{N}{f_t}\right),$$

where:

N is the number of documents in the collection c;

$f_{c,t}$ is the number of occurrences of term t in the collection c; and

f_t is the number of documents that contain term t [64].

Since this metric accumulates the contributions of the collection term frequencies and inverse document frequencies of all query terms, Zhao et al. state that it will be biased towards longer queries. As such, they suggest a normalised metric, given below.

NSCQ: This metric is the SCQ score divided by the number of query terms that occur in the collection:

$$NSCQ = \frac{SCQ}{|q|_{t \in \mathcal{V}}},$$

where:

$\mathcal{V}$ is the vocabulary of the collection [64].

MaxSCQ: This predictor assumes that the performance of a query is determined by the query term that has the highest SCQ score [64]:

$$MaxSCQ = \max_{t \in q} \left[(1 + \ln(f_{c,t})) \times \ln\left(1 + \frac{N}{f_t}\right) \right],$$

where:

N is the number of documents in the collection c;

$f_{c,t}$ is the number of occurrences of term t in the collection c; and

f_t is the number of documents that contain term t [64].

Distribution-Based Measures

In addition to the three similarity-based measures given above, three distribution-based measures (following a similar format) are also defined by Zhao et al.; we repeat them here, as follows:

σ_1: Given a query q consisting of terms $t_1, ..., t_n$, the basic distribution score is defined as the sum of the deviations:

$$\begin{aligned} \sigma_1 &= \sum_{t \in q} \sqrt{\frac{1}{f_t} \sum_{d \in \mathscr{D}_t} (w_{d,t} - \overline{w}_t)^2} \\ w_{d,t} &= (1 + \ln(f_{d,t})) \times \ln\left(1 + \frac{N}{f_t}\right) \\ \overline{w}_t &= \frac{\sum_{d \in \mathscr{D}_t} w_{d,t}}{|\mathscr{D}_t|} \end{aligned}$$

where:

N is the number of documents in the collection c;

$f_{d,t}$ is the frequency of term t in document d;

f_t is the number of documents that contain term t; and

$\mathscr{D}_t$ is the set of documents that contain query term t [64].

σ_2: Here, we normalise σ_1 by the number of query terms that occur in the collection:

$$\sigma_2 = \frac{\sigma_1}{|q|_{t \in \mathscr{V}}}$$

σ_3: This predictor estimates the performance of a query based on the maximum deviation from the mean, that is observed for any one query term [64]:

$$\begin{aligned} \sigma_3 &= \max_{t \in q} \left[\sqrt{\frac{1}{f_t} \sum_{d \in \mathscr{D}_t} (w_{d,t} - \overline{w}_t)^2} \right] \\ w_{d,t} &= (1 + \ln(f_{d,t})) \times \ln\left(1 + \frac{N}{f_t}\right) \\ \overline{w}_t &= \frac{\sum_{d \in \mathscr{D}_t} w_{d,t}}{|\mathscr{D}_t|} \end{aligned}$$

where:

N is the number of documents in the collection c;

$f_{d,t}$ is the frequency of term t in document d;

f_t is the number of documents that contain term t; and

$\mathscr{D}_t$ is the set of documents that contain query term t [64].

Combination Measure

Finally, Zhao et al. give a seventh predictor, which combines the attributes of the two groups of algorithm. This is specified as follows:

joint: For each query term t in query q, this predictor combines both the MaxSCQ and σ_1 scores:

$$joint = \alpha \cdot MaxSCQ + (1 - \alpha) \cdot \sigma_1,$$

where:

α is a parameter that specifies the weight given to the SCQ and distribution score components; Zhao et al. [64] suggest values between 0.7 and 0.85.

Summary of Predictor Performance

Zhao et al. evaluate their query performance predictors using a number of document and query sets, so that the performance over different types of data can be examined.

The data sets used are: TREC GOV2 (consisting of HTML documents, and text from PDF and Word documents), WT10g (a crawl of the web in 1998), and the 2005 Robust Track collection (newswire data). Each data set has queries and relevance judgements associated with it.

As in Section 2.5.3, with the work of He and Ounis [28], the correlation between the predictor scores and the actual performance (average precision) is measured. In this instance, AvICTF and SCS are also tested, to serve as baselines for the new predictors.

Zhao et al. conclude from their results that both similarity between the query and the collection, and the distribution of terms are important in the prediction of query performance, over their test collections. The *joint* predictor is evaluated as the most effective; it strongly outperformed AvICTF and SCS on web data, and gave comparable performance to those over newswire data. Zhao et al. also state that "the new predictors offer a significant advantage over previously proposed pre-retrieval predictors, because the performance of the latter varies drastically between data types."

2.6 Summary

In this chapter we have provided a comprehensive discussion of existing literature, relevant to addressing the collection suggestion problem specified in Chapter 1. Specifically, relevant material addresses work in the domains of traditional document retrieval evaluation, collection selection algorithms and their associated evaluation techniques, and query performance prediction algorithms.

An effective collection suggestion search service (for identifying authoritative collections) requires an effective algorithm for ranking collections with respect to a user's query. As we have discussed in this chapter, there are a large range of existing collection ranking algorithms (from the domain of collection selection), and algorithms that could legitimately be applied to the task of ranking collections (query performance predictors).

As such, in Chapters 6 and 7, we examine whether these existing algorithms may be effectively applied to the collection suggestion task. To this end, we select a sample of those algorithms discussed in this chapter: bGlOSS, CORI, Zobel's four lexicon inspection algorithms, CVV, DFPROP, Distribution of Informative Amount (1), SCS, AvICTF and NSCQ. We have selected the algorithms found to be effective in their original applications, and those whose underlying theories relate to the objectives of collection suggestion.

Notably, we omit from evaluation the n-term indexing based techniques (Section 2.4.7), Decision-Theoretic Framework (Section 2.4.8), and the algorithms for uncooperative environments (Section 2.4.9). The n-term indexing and uncooperative environments based algorithms require all documents across all collections to be ranked for each query. As such, these algorithms are expensive to execute on large collections (more so than algorithms using summary collection statistics). We omit the Decision-Theoretic Framework due to its requirement of cost data (monetary and access time, for example) for the collections, and its emphasis on retrieving documents from the best collection.

In addition to investigating the performance of existing algorithms, with respect to the collection suggestion task, we also develop our own algorithm, with the view of achieving superior performance.

In conducting an evaluation of algorithms for collection suggestion, we look to follow accepted and established evaluation techniques where possible. In instances where these techniques are not entirely suitable, or are deficient, we amend them to suit the requirements of the collection suggestion problem, and ensure that our work draws from previous practices.

For example, as discussed in this chapter, the majority of collection selection algorithm evaluations have utilised synthetic test data collections formed from the TREC document corpus. These collections offer a controlled environment in which to test algorithms. However, previous work has also highlighted that these collections may be unsuitable for evaluation purposes, as they are not representative of a realistic operational environment (see Section 2.3.3).

As such, for our experiments we will utilise the commonly used TREC-based test data sets (in order to observe the strengths and weaknesses of algorithms in a controlled setting). However, we will also develop and use a large test data set, formed from real collection data. While previous work has utilised real collection test data, such data sets have been small, featuring only a handful of collections. We discuss our choice of test data and its justification further in Chapter 4.

We have seen throughout this chapter that information retrieval system evaluations (both traditional document-based, and collection selection) tend to take a descriptive approach to analysis of results. Deductive analysis, which quantifies the significance of results and differences between results, is often neglected. As such, in our evaluation of algorithms we will perform statistical testing on performance scores, where appropriate. Thus, we can indicate the quality of scores achieved: whether an algorithm is *good enough*, as well as whether it is *better* than another algorithm. We discuss our evaluation approach in detail in the following chapter.

Chapter 3

Collection Suggestion Evaluation Methodology and Toolkit

As discussed in Chapter 1 of this thesis, our work is motivated towards the development of a search service operating over digital collections. Such a service is intended to support users looking to find domain-specific, authoritative collections (we refer to this task as collection suggestion). However, there are several prerequisites for any such effective service to be developed; it is these prerequisites that we address in our work.

We suggest that a collection suggestion service should take a similar form to a traditional document search service: a search interface allows the user to enter their query, comprising keywords from their topic of interest. The user will then be presented with a ranked results list of collections, with those deemed most suitable by the service appearing at the top of the list. The user may then follow links to the recommended collections, to search and browse for material of interest.

However, in order to effectively rank collections with respect to the user's query, we require a suitable algorithm; this is one precondition of developing a collection suggestion service. In the previous chapter we discussed a variety of algorithms, from the domains of collection selection and query performance prediction, that may be applicable to the collection suggestion task. Investigating the suitability of these algorithms for collection suggestion (and identifying an optimal algorithm for the task) calls for a methodical and rigorous evaluation methodology. Ideally, this should be consistent with the expectations of information retrieval research. Therefore, developing an appropriate algorithm evaluation methodology is another precondition of developing an effective collection suggestion search service.

As such, in this chapter we discuss our approach to evaluating the suitability and performance of algorithms, with respect to the collection suggestion task. We begin, in Section 3.1, with a brief overview of the methodology, and follow with in-depth descriptions of the various components, in Sections 3.2 to 3.4. In addition, in Section 3.5 we discuss the various tools we have developed to support our empirical work.

We note here that in our empirical work, we are solely interested in the effectiveness

of algorithms at ranking collections. We do not conduct any user-based evaluations to investigate the value or effectiveness of a collection suggestion search service from a user's standpoint. Such evaluations require a mature prototype (implementing a suitable ranking algorithm) to produce useful and reputable results; as we have discussed, development of such a prototype first requires fundamental research.

3.1 Overview of Evaluation Methodology

In order to conduct a rigorous evaluation of the effectiveness of algorithms, with respect to the collection suggestion task, we employ two methods.

Our principal method follows the evaluation strategy used in the related domain of collection selection: as discussed in Chapter 2, a variety of performance measures are used to measure how closely an algorithm produces an ordering of collections that matches a baseline ranking of collections. This baseline often represents the ideal ordering of collections for a given query. However, lower bound baselines are also utilised to help put performance scores into context.

The objectives of collection suggestion differ from those of collection selection. In collection selection, the aim is to maximise relevant documents returned to the user, while minimising the number of collections to search; as such, collections are ranked by their likelihood of containing many relevant documents. Collection suggestion however, aims to recommend collections about the query to the user; we propose that such collections should have a large quantity of relevant documents, which account for a large proportion of the whole collection.

Given these differences, we make adjustments to the original collection selection evaluation methodology, to suit the specific needs of collection suggestion. As such, in Section 3.2 we begin to discuss our approach to evaluating against baselines; this includes the specification of a new baseline ranking. In Section 3.3 we continue by presenting the performance measures we use to evaluate the algorithm-produced collection rankings, with respect to the baseline rankings.

The approach of testing against baselines is key to determining the performance of algorithms. However, as shown in Chapter 2, scores associated with the performance measures are often averaged over a set of test queries. As a result, this approach could be said to take a broad view of algorithm performance: how an algorithm reacts in special or interesting cases is hidden.

Therefore, as an initial step in our algorithm evaluation methodology, we have developed a set of abstract scenarios where we model hypothetical situations, involving only a few collections. By performing this scenario-based testing, we can essentially put algorithms under a microscope: identifying their strengths and weaknesses, and determining whether they rank collections in accordance with the specific goals of collection suggestion. We discuss this technique and its motivation further in Section 3.4.

The scenario-based testing and testing against baselines approaches are complementary: the former allows us to closely scrutinise the suitability of algorithms, and therefore discard those that are clearly poor. The latter considers more general algorithm perfor-

mance, with regard to a variety of measures, and with findings averaged over a large set of test queries.

The execution of scenario-based and baseline evaluations of a large number of algorithms presents the need for apparatus to support experimental work. As such, we have developed programmatic tools to support evaluation of algorithms; we give an overview of these in Section 3.5.

3.2 Testing Against Ranking Baselines

Our primary approach to evaluating the performance of algorithms, with respect to the collection suggestion task, is based on the established technique for evaluating collection selection algorithms. As discussed in Chapter 2, a ranking of collections produced by an algorithm (where collections are ordered by decreasing score, according to the algorithm) is compared to a baseline ranking.

To perform testing against baselines, we require suitable test data. For collection suggestion this comprises a set of collections (groups of documents), a set of test queries, and a set of document relevance judgements, indicating which documents are relevant to each query. This follows established practices from document retrieval and collection selection evaluation. We cover our choice of test data in detail in Chapter 4. However, we note here that document relevance judgements are required for the generation of some baseline rankings.

A baseline ranking can take many forms, depending on the specific goals and focus of the research. For example, in the related domain of collection selection, two baselines are commonly used. The first, Relevance-Based Ranking (RBR), orders collections by the number of relevant documents they contain (determined by document relevance judgements) for the given query. This is considered an optimal baseline: the best performance, which an algorithm should aspire to match, such that the higher ranked collections are most useful to the user. The RBR baseline represents the goals of collection selection: to choose a small number of collections to which the user's query can be submitted. As such, it is prudent to choose those collections likely to contain many relevant documents.

The second baseline often used in collection selection experiments is Size-Based Ranking (SBR). Here, collections are ranked in decreasing order of the number of documents they contain. It is considered to be a lower bound on performance, and as such algorithms should aim to perform better than this baseline.

For our collection suggestion evaluation, we will also use two baseline rankings. Like collection selection, we will use the SBR baseline to represent a lower bound on performance. Algorithms that score worse than this baseline, according to the various performance measures, are unlikely to be suited to our task. As such, those algorithms can be omitted from further evaluation.

Naturally, we want to identify the most effective algorithm; therefore our second baseline is an optimal one. We specify and use a new baseline, that represents the specific goals of collection suggestion: we aim to recommend useful collections to the user, such that they may search and browse them themselves. For this reason, a collection should be relevant,

as a whole, to the user's query; we do not want to user to have to wade through irrelevant material.

Given this, for collection suggestion, a highly ranked collection should contain a large number of relevant documents, but these documents should also constitute a significant proportion of the collection (indicating the collection is *about* the query, or relevant as a whole). It is these ideas from which we formulate an appropriate optimal baseline, for collection suggestion evaluations.

We specify two metrics: for a collection c, RS_c is the *share* of all relevant documents the collection has, while RP_c is the *proportion* of documents in the collection that are relevant. We formulate these as follows:

$$RS_c = \frac{|\,RD_c\,|}{|\,R\,|} \qquad\qquad RP_c = \frac{|\,RD_c\,|}{|\,D_c\,|}$$

where:

$|\,R\,|$ is the number of relevant documents for a given query;

$|\,D_c\,|$ is the number of documents in a collection c; and

$|\,RD_c\,|$ is the number of relevant documents in collection c.

The RS_c and RP_c metrics are essentially equivalent to the recall and precision measures of traditional document retrieval. RS_c is basically the recall score for the collection as a whole, while RP_c can be interpreted as within-collection precision.

To generate an optimal collection ranking for a given query, we utilise a technique used (in document retrieval performance evaluation) to combine recall and precision into a single measure of performance. This measure is called the F-score [2], and is produced by calculating the harmonic mean of recall and precision. The F-score has a value of between 0 and 1, and is highest when both components score highly [2]. Therefore, given the RS_c and RP_c scores for each collection, we calculate their F-score by:

$$F_c = \frac{2}{\frac{1}{RS_c} + \frac{1}{RP_c}}$$

Our decision to combine RS_c and RP_c in this way was influenced by their similarity to recall and precision, and the equal importance of each of the two metrics (we wish to maximise both).[1]

In the context of collection suggestion, an F-score of 0 shows that there are no relevant documents in the collection, while a score of 1 shows that a collection contains all available relevant documents, and these make up the entire collection. Therefore, the higher the F-score, the more relevant the collection as a whole, relative to the other collections.

[1] An alternative approach to generating an optimal collection ranking, using the RS_c and RP_c metrics, could be to calculate their weighted harmonic mean, where one component is assigned more importance than the other. For example, giving more weight to the RP_c component would emphasise the importance of the overall relevance of a collection.

We order the collections in decreasing order of their F-score; we refer to this optimal ranking as the F-score Based Ranking (FsBR). The FsBR baseline is used to evaluate the performance of algorithms, using a variety of performance measures, discussed in the following section.

3.3 Performance Measures

In this section we discuss the range of performance measures we will use to evaluate algorithms with respect to collection suggestion. Each measure considers a different aspect of performance, allowing us to determine any particular strengths or weaknesses an algorithm has.

Several of the performance measures presented have previously been used in evaluations of algorithms for collection selection (see Chapter 2), and are applicable to collection suggestion. However, we also utilise additional measures, that reflect the specific objectives of collection suggestion.

We first discuss a series of rank correlation coefficients, which allow us to examine the similarity of two collection rankings. Following this we look at measures analogous to recall and precision. Finally, we introduce new measures that look closely at algorithm effectiveness within the top rank positions.

3.3.1 Rank Correlation Coefficients

A rank correlation coefficient measures the similarity between two different orderings of a set of objects. As such, we can use rank correlation coefficients to examine how closely an algorithm produces an ordering of collections that matches a baseline ranking.

For our evaluation, we use three rank collection coefficients. The first, Spearman, is a traditional and well-known rank correlation coefficient, and has previously been used in evaluations for collection selection, by the likes of French et al. [16, 17, 44]. The other two, Blest and Da Costa, are examples of weighted rank correlation coefficients. That is, they place importance on similarity within the top portions of the rankings.

The Blest and Da Costa weighted rank correlation coefficients are particularly useful for measuring algorithm effectiveness for collection suggestion. Since we are recommending a small number of collections to the user (those that score highest), correctly identifying and ordering the best collections is important.

Each of these correlation coefficients produces a score between 1 and -1. A positive score indicates a positive similarity between the two rankings, with 1 indicating total agreement. In contrast, a negative score shows an inverse relationship, with a score of -1 showing that the two rankings are the exact opposite of each other.

We describe the three correlation measures in the following sections. We note here that a correlation score between a baseline ranking and an algorithm-produced collection ranking is calculated for each test query. The scores over all test queries are then averaged, to give a single estimate of algorithm performance (this follows the convention of work in the collection selection domain). As discussed later in this section, we perform statistical sig-

nificance testing on these average correlations, to help interpret the strength of the scores achieved.

Spearman Rank Correlation

As described previously in Chapter 2, some evaluations for collection selection (such as those by French et al. [16, 17, 44]) have used the Spearman rank correlation coefficient to measure the similarity between the lower bound Size-Based Ranking (SBR), and the collection rankings produced by the algorithms under test. In this instance, a positive correlation score suggests that an algorithm has a tendency to favour large collections, over smaller (possibly more suitable) ones. Given this previous practice, we will also utilise Spearman in this way.

However, the primary focus of our evaluation of algorithms for collection suggestion, is to examine how closely the algorithm performances match that of our optimal baseline: the FsBR baseline, described in Section 3.2.

As such, in contrast to collection selection studies, we also use Spearman rank correlation to examine the similarity between an algorithm-produced ordering of collections, and our FsBR baseline.[2]

In our calculation of the Spearman rank correlation coefficient, we use an alternative equation to that specified in Chapter 2. Given below, the equation, specified accounts for the presence of tied ranks: that is, where collections are given the same score by an algorithm. Accounting for tied ranks is necessary, as all test collections are included in the rankings, regardless of whether they are deemed by the algorithm to have any relevance. As such, several collections within the rankings (both baseline and algorithm-produced) may have a score of zero.

Therefore, we calculate the Spearman rank correlation coefficient, r_s between two collection orderings X (the baseline) and Y (the algorithm-produced ranking), as follows [37]:

$$
\begin{aligned}
r_s &= \frac{\sum x^2 + \sum y^2 - \sum d^2}{2\sqrt{(\sum x^2)(\sum y^2)}} \\
\sum x^2 &= (n^3 - n)/12 - \sum T_X \\
\sum y^2 &= (n^3 - n)/12 - \sum T_Y \\
T_X &= (t^3 - t)/12 \\
T_Y &= (t^3 - t)/12 \\
\sum d^2 &= \sum_{i=1}^{n} (X_i - Y_i)^2
\end{aligned}
$$

where:

n is the number of collections we are ranking;

[2] We note here that other rank correlation coefficients, such as Kendall Tau, could also be used for this task. Our choice of Spearman simply follows from its previous use in collection selection literature, as discussed in Chapter 2.

T_X and T_Y are calculated for each group of tied rankings, in rankings X and Y;

t is the number of items within a particular group of tied ranks;

$\sum d^2$ is the sum of the squared rank differences of the collections, between the baseline ranking and algorithm-produced ranking; and

X_i and Y_i are the ranks of collection i, according to the X and Y rankings.

Weighted Rank Correlation

In addition to the Spearman rank correlation coefficient, we also utilise two weighted rank correlation coefficients; we refer to these as Blest weighted rank correlation [3] and Da Costa weighted rank correlation [8].

These two correlation measures place an importance on agreement between rankings within the top rank positions. This is particularly relevant to the collection suggestion task: the ultimate goal is to recommend to the user a small number of collections they would find most useful, that they can search and browse themselves. As such, correctly identifying the top few collections is very important: it is costly to the user (for example, in terms of time) if they are sent to a sub-par collection.

As these two metrics are fairly new, and their application in this instance is novel, we use both in our evaluation: agreement or similarity between the correlation scores produced by each measure suggests reliability.

The Blest weighted rank correlation coefficient v, between two rankings, is calculated as follows [3]:

$$\begin{aligned} v &= 1 - \frac{24W}{n(n+1)^2(n-1)} \\ W &= \frac{1}{2}\sum_{i=1}^{n}(n+1-i)^2 q_i - \frac{n(n+1)^2(n+2)}{24} \end{aligned}$$

where:

n is the number of collections we are ranking;

i is the current rank position, in the baseline ranking; and

q_i is the rank position within the algorithm-produced ranking, of the ith item in the baseline ranking.

The Da Costa weighted rank correlation coefficient r_w, between two rankings, is calculated as follows [8]:

$$r_w = 1 - \frac{6\sum_{i=1}^{n}(R_i - Q_i)^2((n-R_i+1)+(n-Q_i+1))}{n^4+n^3-n^2-n}$$

where:

n is the number of collections we are ranking;

R_i is the rank position of the current collection, within the baseline ranking; and

Q_i is the rank position within the algorithm-produced ranking, of the ith item in the baseline ranking.

For these two weighted rank correlation measures, we only calculate the correlation between the optimal FsBR baseline and the collection rankings produced by the algorithms. Since the measures focus on correct ranking at the top positions, it makes little sense to perform analysis in this way against SBR: it is a lower bound, rather than the output we are trying to achieve.

Statistical Significance

So far, we have discussed the rank correlation measures we will use to determine how closely an algorithm-produced collection ranking matches a baseline ranking. As described, the scores fall between -1 and 1, with a negative score indicating two ranks have an inverse correlation, and a positive score indicating similarity.

However, as we touched upon in Chapter 2, simply stating a correlation score is not enough, and it is inaccurate to use terms such as 'strong' or 'weak' to describe a correlation score: they are open to interpretation. As such, how do we interpret what is a 'good' score, and whether a score achieved by one algorithm is actually better than another?

The answer is statistical significance: a correlation score is statistically significant if it is unlikely to have occurred by chance. The closer the correlation score is to one (either in the positive or negative direction), the less likely it is to have occurred by accident (though the likelihood also depends on how many items we are ranking).

In our evaluation, we will apply two tests of statistical significance. The first, (one-tailed) t-test [37, p. 350], simply indicates whether a particular correlation score is significant (unlikely to have occurred by chance).

The second test, (two-tailed) Z-test [32, p. 35], is used to determine whether two correlations, for example from two different algorithms, are statistically significantly different. That is, is one algorithm better than another.

When presenting our results from these tests (in Chapters 5, 6 and 7), we will state whether the correlations are statistically significant at a 5% significance level ($p < 0.05$).

3.3.2 Recall and Precision Analogues

We recall from Chapter 2 that three performance measures that are analogous to recall and precision have previously been used in collection selection evaluations. We summarise these measures as follows:

- $\mathcal{R}_n$ is analogous to recall, and measures the fraction of the available merit from the top n collections in the baseline ranking, that has been accumulated by the top n collections in the algorithm-produced ranking. That is, with respect to the optimal baseline, $\mathcal{R}_n$ measures how well an algorithm selects the best collections.

- $\hat{\mathcal{R}}_n$ is a slight adaption of $\mathcal{R}_n$, and shows how much of the total merit (from all collections, with respect to the baseline) has been accumulated by the first n collections in the algorithm-produced collection ranking.
- $\mathcal{P}_n$ is analogous to precision, and gives the fraction of the top n collections in the algorithm-produced ranking that have non-zero merit. That is, the number of collections that have some relevance to the query.

Despite a recent trend in collection selection literature to present performance results using only the $\mathcal{R}_n$ measure, we will use all three measures in our evaluation of algorithms for collection suggestion. This approach allows a thorough analysis of algorithm performance from different angles. As such, we are better able to understand the collection suggestion problem, and learn which performance measures are most useful.

For each measure, we present the results in the form of tables (showing scores achieved at particular values of n) and graphs (plotting scores for all values of n). For each measure we average the scores for each value of n, over all test queries. This gives a single measure of performance at each value of n.

To support the interpretation of the scores, when producing graphs, in addition to the algorithm scores, we also plot the optimal performance (the FsBR baseline against itself), and the performance of the Size-Based Ranking (SBR) in relation to FsBR. Therefore, we can observe the upper and lower bounds of performance.

Note that for these performance measures, we do not evaluate algorithms against SBR directly: SBR is merely a lower bound, not a ranking we aspire to achieve.

3.3.3 Evaluation in Top Rank Positions

The performance measures we have presented so far are generally concerned with how closely an algorithm produces a ranking of collections that emulates an optimal baseline ranking. That is, we are aiming to rank every collection correctly.

However, for collection suggestion, the aim is to recommend a small number of collections to the user, so that they may search and browse them themselves. As such, it is highly important to correctly identify and rank the top few collections; since the user will likely not be interested in the less suitable collections, accurately ranking those beyond the top few positions is secondary.

We introduce and use two performance measures, Precision@5 and Correct@1, that allow us to closely examine algorithm performance at the top of the collection ranking, rather than over the ranking as a whole. While an algorithm may perform well overall, it may not perform well enough at the top end of the ranking. We describe these two measures below:

Precision@5: For the top five collections in the algorithm-produced collection ranking, we calculate the fraction of collections that *should* be ranked within the top five, according to the FsBR baseline. Specifically, for a given query, Precision@5 is calculated as follows:

$$P@5 = \frac{|\text{Top 5 collections in baseline ranking} \cap \text{Top 5 collections in algorithm ranking}|}{5}$$

We average this measure over all test queries, to give a single representation of algorithm performance. An optimal score is 1: for all queries, the algorithm correctly includes the best five collections within the first five rank positions (regardless of order).

Correct@1: Here, we calculate the number and percentage of test queries for which an algorithm correctly ranked the first collection: this is the first (and possibly only) collection a user will visit, and therefore it is important to get it correct.

3.4 Scenario-Based Testing

In Sections 3.2 and 3.3 we discussed our main technique for evaluating the performance of algorithms with respect to the collection suggestion task: using a range of performance measures to compare an algorithm-produced collection ranking to upper and lower bound baseline rankings.

While basket-of-averages testing against a baseline via different performance metrics gives a comprehensive view of algorithm performance, it lacks an ability to discriminate specific cases. By examining specific searches, we may better understand the detailed effects of different algorithms, and we can then seek to improve them in a systematic manner.

Therefore, as a preliminary stage in our evaluation of algorithms, we perform scenario-based testing. Here, we specify a series of hypothetical situations, involving only a small number of collections. This enables us to easily reason about the best ordering of the collections, according to the specific objectives of collection suggestion.

The scenario-based testing essentially provides a 'health-check' for algorithms. Due to their simplicity, execution of these tests is cheap, enabling us to quickly determine which algorithms are likely to be suitable for the collection suggestion task. As a result, we can rule out poor performers, prior to executing the more comprehensive and expensive baseline testing.

We provide further details of our scenario-based testing approach in the following sections: in Section 3.4.1 we describe the format of the scenarios. Section 3.4.2 gives a summary of the scenarios we model; a comprehensive specification of each scenario is given in Appendix A. Finally, in Section 3.4.3 we give an overview of apparatus we have developed to support the creation and management of test scenarios.

3.4.1 Format of a Scenario

The scenarios we develop to support our evaluation of algorithms for collection suggestion comprise two main components: a query, and a set of collections. This is similar to the standard TREC model, where we have a search problem statement (query topic) and matching documents (a set of documents and relevance judgements).

The query represents the information need of the user. In a scenario, the query is specified in abstract terms, in the form $t_1 \ldots t_n$.

For each scenario, we model only three collections; we refer to these as C_A, C_B and C_C. For each collection we specify the statistics that are likely to be used by the algorithms we

are testing: the number of documents it contains, the total number of terms it contains, and the term and document frequencies of each of the query terms.

By varying the quantities and ratios of the collection statistics, in addition to the query length and number of query terms matched, we can create different test cases. As such, we can identify any strengths and weaknesses of the algorithms, and determine whether they rank collections in accordance with the goals of the collection suggestion task.

Due to the small number of collections modelled, it is easy to reason about the optimal ordering of collections within a scenario. For collection suggestion, a highly ranked collection should contain a large number of relevant documents, and these documents should comprise a large proportion of the collection; this suggests the collection is *about* the query topic.

These properties are expressed in criteria for highly ranked collections, specified by Zobel [66]. They state that a collection should be ranked highly if for each query term:

1. The term occurs in the collection;
2. The term is common in the collection (relative to the other collections);
3. The collection contains a relatively high proportion of documents featuring the term; and
4. There are likely to be documents in the collection in which the term is relatively frequent [66].

We use these criteria to reason about the best ordering of collections within a scenario. However, in the interests of simplicity and consistency, we specify the attributes of the collections within a scenario such that the best ordering is always C_A, C_B, C_C. That is, C_A is always the best collection to match the query. C_B is considered a 'runner up', while C_C is always distinctly worse than the other two.

We note here that in the scenario-based tests, the aim of an algorithm is to produce a collection ordering that exactly matches the optimal ordering. As such, for each scenario, an algorithm may either 'pass' or 'fail' the test: it either produces an exactly correct ranking, or it does not.

We provide comprehensive specifications of our set of scenarios in Appendix A; a summary of each of the scenarios is given in the following section.

3.4.2 Summary of Scenarios

In Appendix A we provide complete specifications for our set of scenarios. However, a descriptive summary of the attributes of each scenario is provided below:

S_1: Collections are equal in size. We vary the number of documents that contain the query terms.

S_2: Collections differ in size, but the *proportion* of documents that contain the query terms (in C_A and C_B) is the same.

S_3: Collections differ in size, but the *quantity* of documents that contain the query terms (in C_A and C_B) is the same.

S_4: Collections are equal in size. A single-term query is used. We vary the number of documents that contain the query term.

S_5: Collections are equal in size. C_B has the same term occurrences as C_A for some query terms, but does not match all query terms.

S_6: Collections are equal in size. C_B has higher occurrences of some query terms than C_A, but does not match all query terms.

S_7: Collections differ in size. C_B is larger than C_A, with a higher quantity (but smaller proportion) of matched documents. This scenario is intended to represent polysemy.

We note that these scenarios are not intended to be comprehensive or cover all possible situations. They are a sample of situations (where we vary different collection attributes), that are useful for 'ruling out' weak algorithms early on.

3.4.3 Scenario Administration Tool

To support the creation and management of test scenarios, we have developed a Java application (which interfaces with a MySQL database) The interface for the tool is shown in Figure 3.1.

The functionality implemented allows the display of all data associated with the scenarios. In addition, the user can add or remove a scenario, or append or remove collection and term statistics from an existing scenario.

In the following section we given an overview of additional apparatus we have developed, to support the execution of our evaluation experiments.

3.5 Test Execution Tools

Given the evaluation techniques we have discussed in this chapter, in Chapters 5, 6 and 7 of this thesis we undertake a large-scale evaluation of the suitability and effectiveness of existing algorithms (alongside our own algorithm), for the collection suggestion task. Such an undertaking presents the need for tool support: to enable us to easily execute tests on different test data sets, and produce and analyse performance results.

Therefore, to facilitate the execution of our algorithm evaluation (via scenario tests and test queries) we have developed two applications. The first application is a web service, dubbed *Doddle*, and implemented using Java Server Pages. As shown in Figure 3.2, we have developed a search interface which allows the execution of a query against a test data set, using one or more of the algorithms implemented. The total execution time of a single test query, using all implemented algorithms, is approximately two seconds.

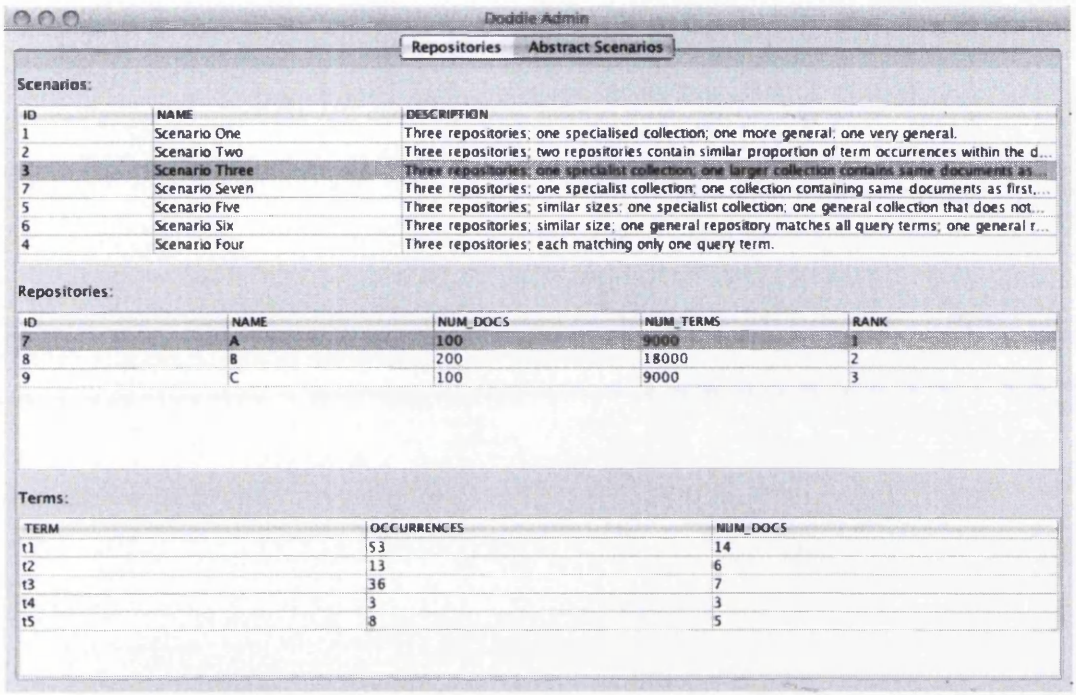


Figure 3.1: Interface for tool to support the management of test scenarios.

Upon execution of the query, a results page (see Figure 3.3) shows the collection ranking produced by each algorithm, the scores associated with each collection, and the contributions of each query term to a collection score. With these, we can gain insight into how a particular ordering of collections has been achieved on a given query.

The web service also includes functionality to execute the scenario-based tests (see Figure 3.4). A results page (Figure 3.5) shows the collection rankings produced by the algorithm, collection scores, and whether the rank is correct. The execution time of a single scenario, over all implemented algorithms, is less than half a second.

The second application is a (Java) command-line tool, which allows the batch execution of a set of test queries, for all algorithms under test, over a chosen data set. In addition to executing a set of test queries, the tool produces reports (CSV files) giving the performance scores according to the various measures discussed previously in this chapter. The report files can then be used to generate graphs (for example, using LaTeX and the PGFPLOTS[3] package) and tables, such as those given in Chapters 5, 6 and 7 of this document, to support the analysis of results and trends. Where appropriate, the application also calculates whether performance scores are statistically significant. The total execution time for a batch of test queries (including the generation of report files) ranges from between 20 minutes and three hours, depending on which data set (see Chapter 4) we are using.

[3] http://sourceforge.net/projects/pgfplots/

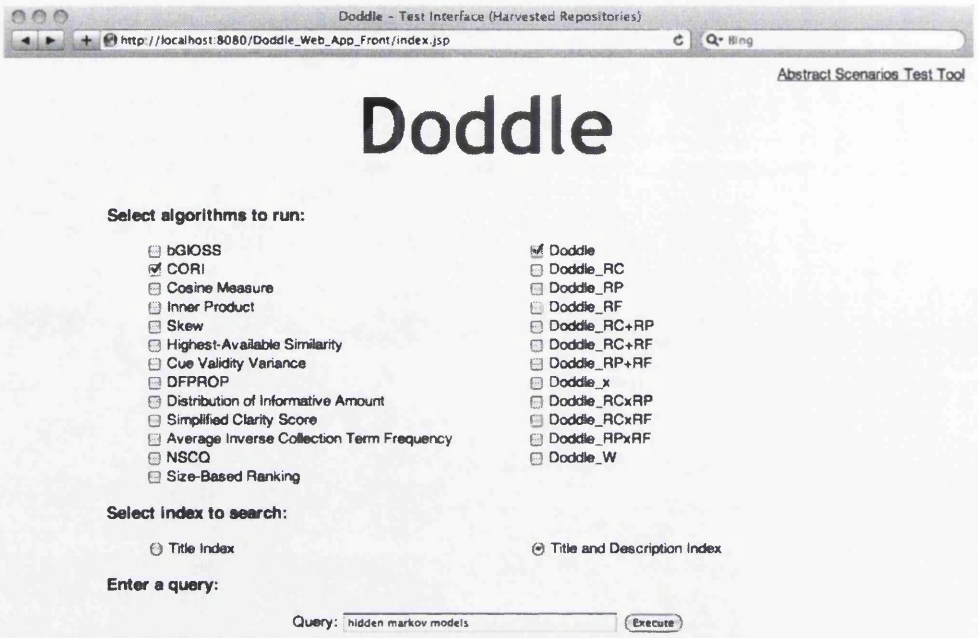


Figure 3.2: Test interface allowing the execution of a query, with multiple ranking algorithms.

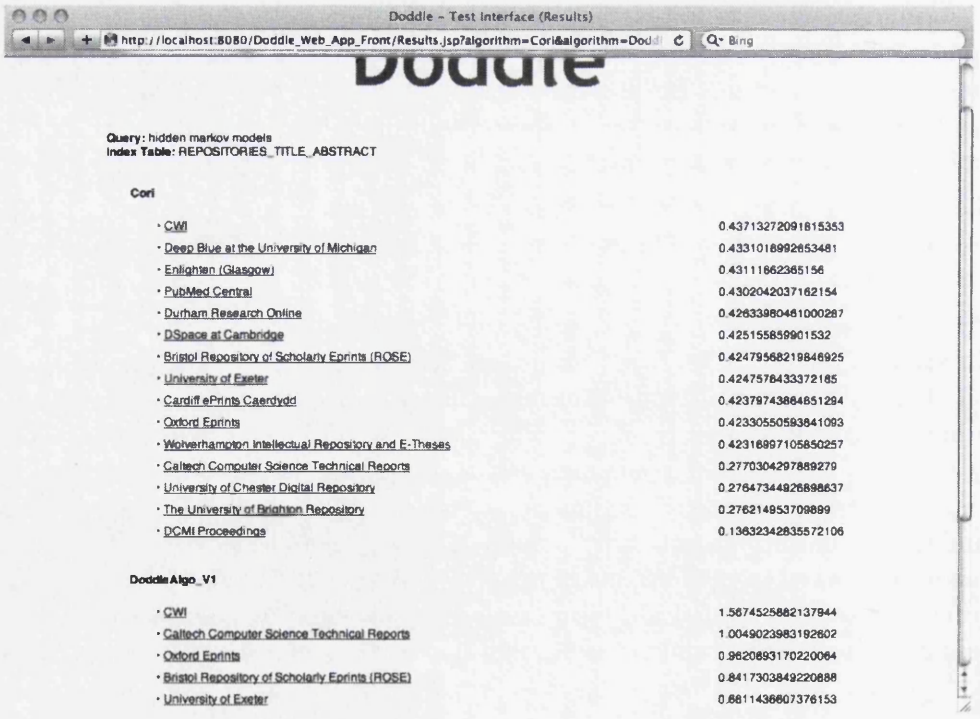


Figure 3.3: A results page from executing a single test query, using two algorithms.

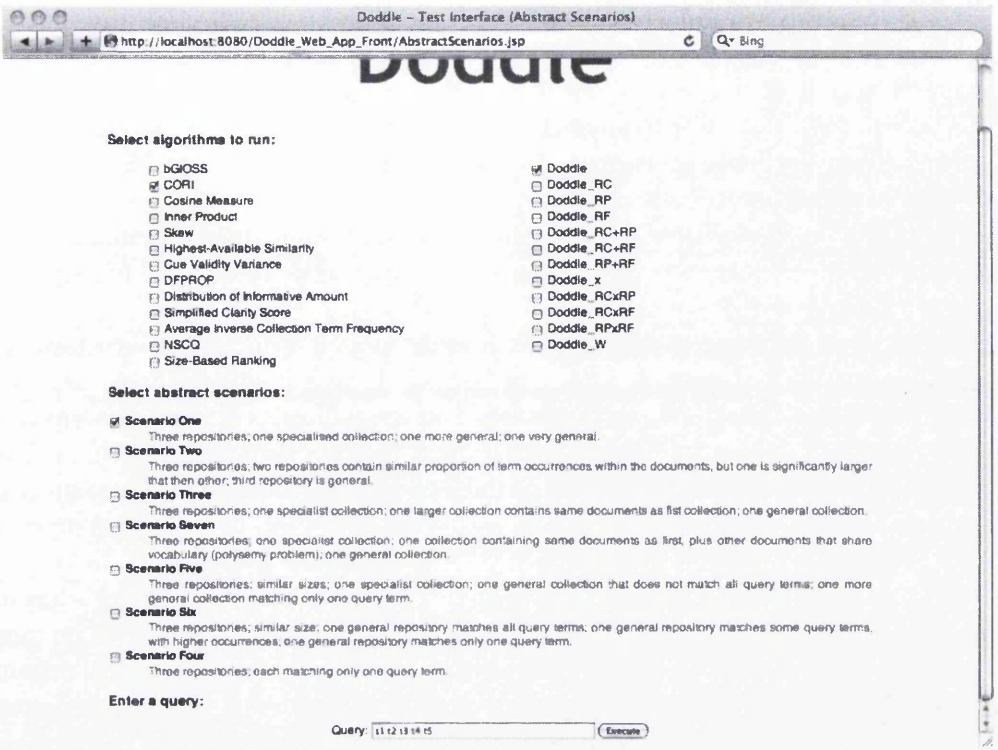


Figure 3.4: Test interface allowing the execution of scenario tests, with multiple ranking algorithms.

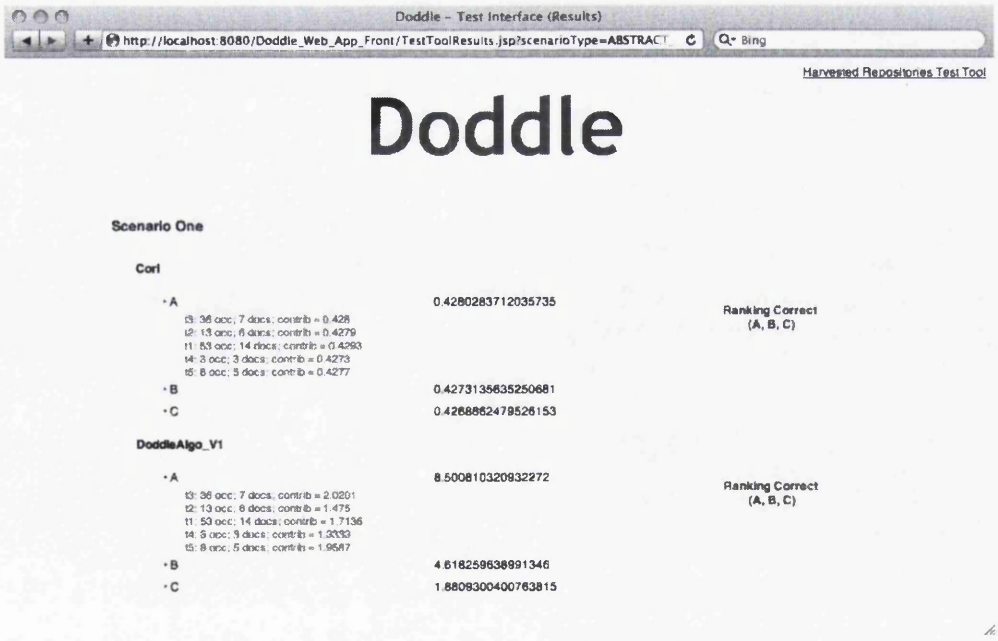


Figure 3.5: A results page from executing a scenario test, using two algorithms.

3.6 Summary

A primary contribution of this work is to conduct a large-scale evaluation of the effectiveness of a range of algorithms, with respect to collection suggestion. The motivation behind this is to identify an optimal collection ranking algorithm that could be employed in a search service for identifying authoritative collections.

To conduct such an evaluation, we require a rigorous and methodical evaluation strategy. For this, we will employ two complementary methods, as discussed in this chapter, and summarised here.

As an initial stage of our evaluation, we have developed a scenario-based testing approach. With this, we can examine algorithm behaviour on a small set of test cases, using only a few collections. In this way, we can look closely at whether an algorithm seems to rank collections in-line with the objectives of collection suggestion. In addition, we can observe any interesting algorithm characteristics. As these scenario tests are cheap to perform, we can use them to identify algorithms that are clearly unsuitable for the collection suggestion task.

The second stage of our evaluation methodology supports a comprehensive study of algorithm performance. For this, we adapt and extend the established evaluation methodology used to evaluate collection selection algorithms, whereby a range of performance measures are used to evaluate an algorithm-produced collection ranking, with respect to one or more baseline rankings. With this technique findings are averaged over a set of test queries, and the different performance measures used allow us to identify strong or weak aspects of algorithm performance.

For the baseline testing, we use several performance measures previously used in collection selection evaluations: Spearman rank correlation, $\mathcal{R}_n$, $\hat{\mathcal{R}}_n$ and $\mathcal{P}_n$. However, of particular interest to us for collection suggestion is algorithm performance within the top rank positions: these are the collections the user is most likely to visit, so they must be correct. To reflect this, we use two weighted rank correlation coefficients to measure performance; these have not previously been applied to experiments involving ranking of collections. In addition, we have specified two new performance measures, Precision@5 and Correct@1, to further scrutinise performance within the top rank positions.

In the following chapter we continue to discuss our approach to evaluating algorithms for collection suggestion, by exploring the test data we will utilise in the testing with baselines portion of our evaluation.

Chapter 4

Test Data Sets

In the previous chapter we discussed our methodology for evaluating algorithms for the collection suggestion task. Our principal strategy, testing against ranking baselines, follows the approach traditionally used for evaluating collection selection algorithms: a variety of metrics are used to determine how well an algorithm estimates an optimal ranking of collections. To conduct a rigorous and robust evaluation of algorithms with baseline-based testing, we require appropriate test data. There are several key properties the test data should exhibit: it should be reproducible, and consistent with both the intended operational environment, and the expectations of information retrieval research [25].

In document retrieval, evaluations of algorithms typically follow the Cranfield strategy [7]. Here, a set of test queries is executed over a single set of documents. Relevance judgements, indicating which documents are relevant to each query, are used to calculate performance scores such as precision and recall. As such, test data sets comprise a set of documents, queries, and associated relevance judgements.

In the domain of collection selection, an area highly relevant to collection suggestion, evaluations also follow the Cranfield strategy. However, there are two differences in the application of the strategy, compared to document retrieval. First, test queries are executed over multiple groupings (that is, collections) of documents. Secondly, the document relevance judgements are instead used to produce the optimal baseline ranking: in this instance, by ranking collections by the number of relevant documents they contain.

For our evaluation of algorithms for collection suggestion, we follow the composition of test data sets used in collection selection experiments: collections of documents, queries, and document relevance judgements. As such, the format of our test data sets will be consistent with commonly accepted practice. However, the question remains, how do we make sure our test data is also realistic?

Ensuring a test data set is representative of the operational environment is a significant challenge. In experiments involving ranking collections, a good data set would include a number of different collections of varying size and subject matter. In addition, the content of a given collection should, in general, follow a single topic or theme [10]. In collection selection, there are two approaches to compiling test data sets. As discussed in Chapter 2,

the favoured and conventional approach is to create synthetic collections, typically using discs 1-3 of the TREC document corpus. Here, documents are often grouped by date, and the source from which they originated. An advantage of this approach is that queries and relevance judgements are provided with the TREC corpus, making experiments easily reproducible. However, the TREC corpus comprises mostly news articles, and collections created from the corpus tend to be small and heterogeneous. Such collections are representative of neither the diverse material present in real digital collections, nor their size and subject scope.

An alternative to synthetic TREC-based collections is to use real collection data, which provides a more realistic sample of collection sizes and content. However, this presents complexities: in addition to gathering collection content, queries and document relevance judgements must be produced. It is likely for these reasons that the few evaluations of collection selection algorithms using real collection data (for example, Gravano et al. [22], Yuwono and Lee [63] and Thomas and Hawking [56]), have used only a handful of collections: a number too small to represent the operational environment.

It is clear, from empirical work in the collection selection domain, that it is difficult to create a test data set that: is consistent with the expectations of the intended operational environment, conforms to accepted information retrieval practices, and enables others to reproduce experiments. Indeed, achieving this in a single test data set is impractical.

As such, to fulfil the above criteria, we use a two-pronged approach in our evaluation of algorithms for the collection suggestion task: both established TREC-based data sets, *and* new data sets comprising real collection data. To follow established practice we will use several existing TREC-based data sets; these offer a controlled setting in which we can scrutinise algorithms on collections exhibiting particular properties. More importantly, we will also construct and use a realistic test data set, to form a critical part of our evaluation.

The remainder of this chapter discusses the test data sets we use. In Section 4.1 we systematically review the previously used TREC data approaches. In Section 4.2 we present our realistic data sets, and discuss their selection and construction in fine detail; given the lack of established good practice in the area, this will be studied in depth. Finally, in Section 4.3 we summarise.

4.1 TREC Test Data Sets

The primary strategy for evaluating algorithms for collection selection is to use test data sets (collections of documents and queries) that are derived from discs 1-3 of the TREC corpus. An advantage of this approach is the provision of relevance judgements for each query topic.

However, as discussed in Chapter 2, such test data sets have limitations. For example, the TREC corpus used contains documents from only seven sources (Wall Street Journal, Federal Register, Associated Press, Department of Energy abstracts, Computer Select discs, San Jose Mercury News and U.S. Patents), three of which contain exclusively news articles. As such, the overall topic diversity of the documents is likely to be limited. In addition, the partitioning techniques used to create the artificial collections within the data sets tend to result in collections containing documents on several different topics. These factors suggest

that TREC-based test data sets are not entirely realistic of the collection suggestion operational environment, in which we expect there to be a mix of both specialist and generalist collections, covering a diverse range of topics.

Despite this, using TREC-based data for our collection suggestion evaluation does have value: it allows us to test algorithms in a more controlled environment than that offered by real collection data sets (such as those we present in Section 4.2), and to mirror experiments in the related collection selection domain.

During our evaluation of algorithms for collection suggestion, we will utilise six established test data sets, built from the TREC corpus. Each test data set exhibits different properties, such as differences in document groupings, collection sizes and distribution of relevant documents. As such, we can examine the performance of algorithms over different conditions (recall from Chapter 2 that collection selection algorithms have been found to perform differently on different data sets).

We discuss these TREC-based test data sets in the following sections, with summary statistics given in Table 4.1: the first three data sets are from the work of French and Powell [18, 43][1], while the remaining data sets are attributed to Si and Callan [49], and are used in several studies [50, 47, 56]. In addition, in Section 4.1.7 we provide details of the queries associated with these TREC-based test data sets.

Table 4.1: Summary statistics for the TREC-based test data sets.

		Documents Per Collection				
Test Set	**Collections**	**Total**	**Min.**	**Avg.**	**Max.**	**Avg. Doc. Length**
SYM-236	236	691058	1	2928	8303	322.7
UDC-236	236	691058	2891	2928	3356	322.7
UBC-100	100	1078166	752	10782	39723	244.5
2LDB-60COL	62	1078166	752	17390	232031	244.5
AP-WSJ-60COL	62	1078166	752	17390	242918	244.5
FR-DOE-60COL	83	1078166	752	12990	226087	244.5

4.1.1 SYM-236 (Source-Year-Month)

When developing the SYM-236 test data set, French et al. specified several requirements: the data set should contain at least 100 collections, and documents should be grouped into collections of related material [18]. It was felt that these requirements would produce a suitably large and realistic test environment.

As such, SYM-236 was formed by partitioning the TREC corpus (discs 1-3) first by source, then by year, and finally by month. This resulted in 236 document collections of varying size, containing documents related by source and time of publication.

[1] Files mapping documents to each collection in French and Powell's data sets can be found at: `http://www.cs.virginia.edu/~cyberia/testbed.html` [43].

The SYM-236 data set contains a range of different size collections, as shown in Figure 4.1. However, the collections may be considered to be fairly small overall, with an average of just under 3000 documents per collection (see Table 4.1), and the largest collection containing only 8303 documents.

In Figure 4.2 we see the distribution of relevant documents (according to the relevance judgements) within the data set, and the number of queries for which each collection contains some number of relevant documents. When looking at the distribution of relevant documents, we see that the larger collections tend to contain the most relevant documents, and therefore match the most queries. As such, the SYM-236 test data set may favour an algorithm correlated to the Size-Based Ranking baseline.

4.1.2 UDC-236 (Uniform-Document-Count)

The UDC-236 test set was developed in conjunction with SYM-236, and follows the same requirements. However, UDC-236 also has the additional requirement that collections should be of roughly equal size (in terms of number of documents). By controlling the collection sizes, performance differences relating to an algorithm's preference of larger collections are factored out, and thus we can identify other biases in the algorithms [43].

Figure 4.3 shows the distribution of relevant documents for the UDC-236 test data set.[2] We see that the relevant documents are more evenly distributed amongst the collections than in SYM-236; however, the queries tend to be biased towards the collections whose documents originate from the Associated Press and Wall Street Journal sources.

Due to the even distribution of documents and relevant documents, algorithms may find this test data set more difficult than SYM-236, as differentiating between collections may be more challenging: although there are over 200 collections, they stem from only seven sources, and thus collections originating from the same source may appear similar.

4.1.3 UBC-100 (Uniform-Byte-Count)

The UBC-100 test data set was developed independently from SYM-236 and UDC-236, and therefore was not influenced by the same requirements. Here, documents were split into 100 collections of roughly equal size (approximately 30MB each), with a given collection containing documents from only one source [43].

As shown in Figure 4.4, like SYM-236, the collections in this data set vary in size. While the largest collections in SYM-236 tend to contain most of the relevant documents, this is not the case for UBC-100. As shown in Figure 4.5, some very large collections contain very few relevant documents. Given this, algorithms that correlate with the Size-Based Ranking baseline may struggle on this test data set.

Finally, we note that collections in this test data set are on average, over three times larger (in terms of number of documents) than those in SYM-236, and there is greater variation in the sizes of the collections.

[2] We omit a graph showing collection sizes in the UDC-236 data set as in this instance collection sizes are uniform.

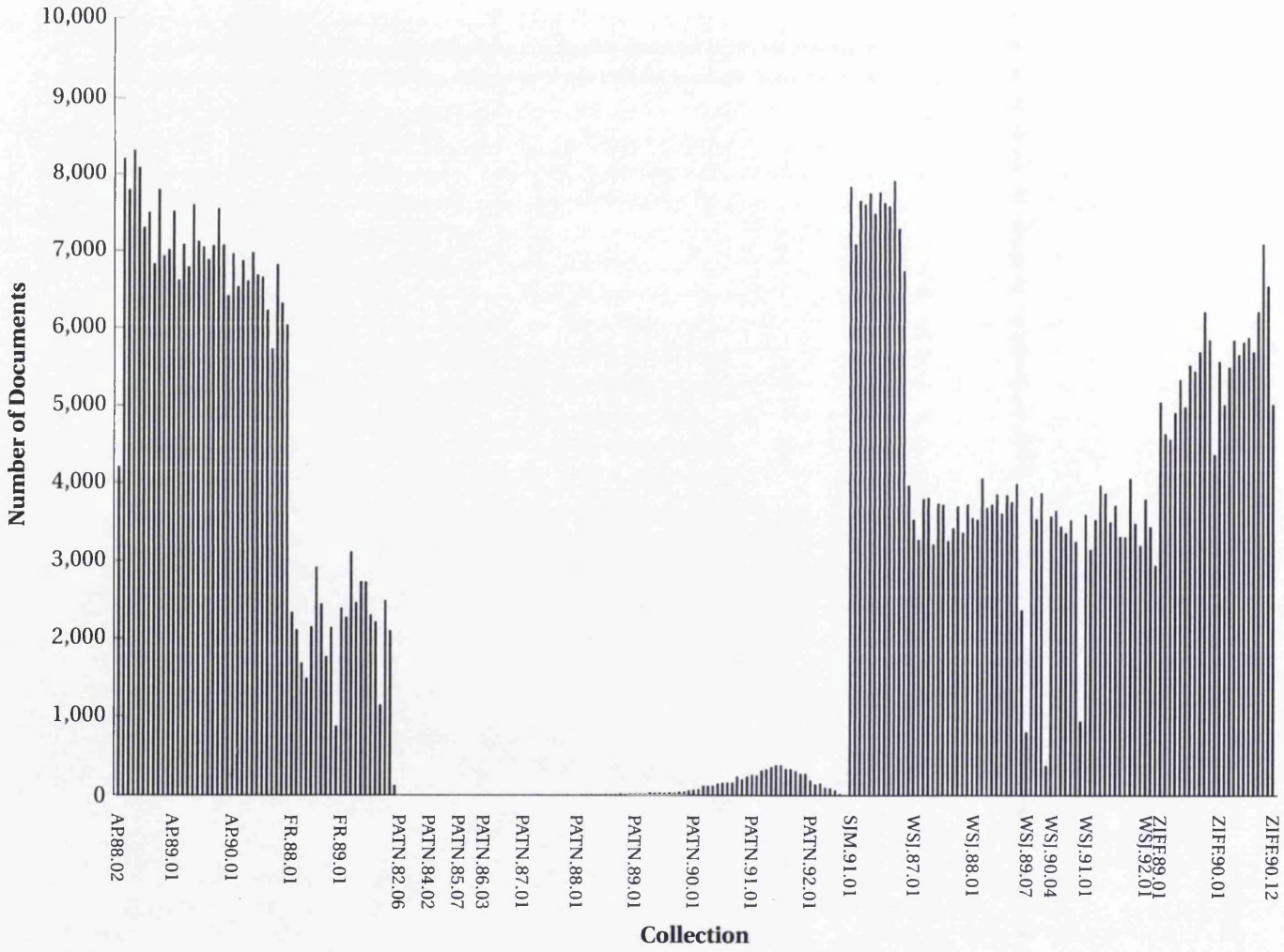


Figure 4.1: The distribution of documents within collections in the SYM-236 test data set (reproduced from [43]).

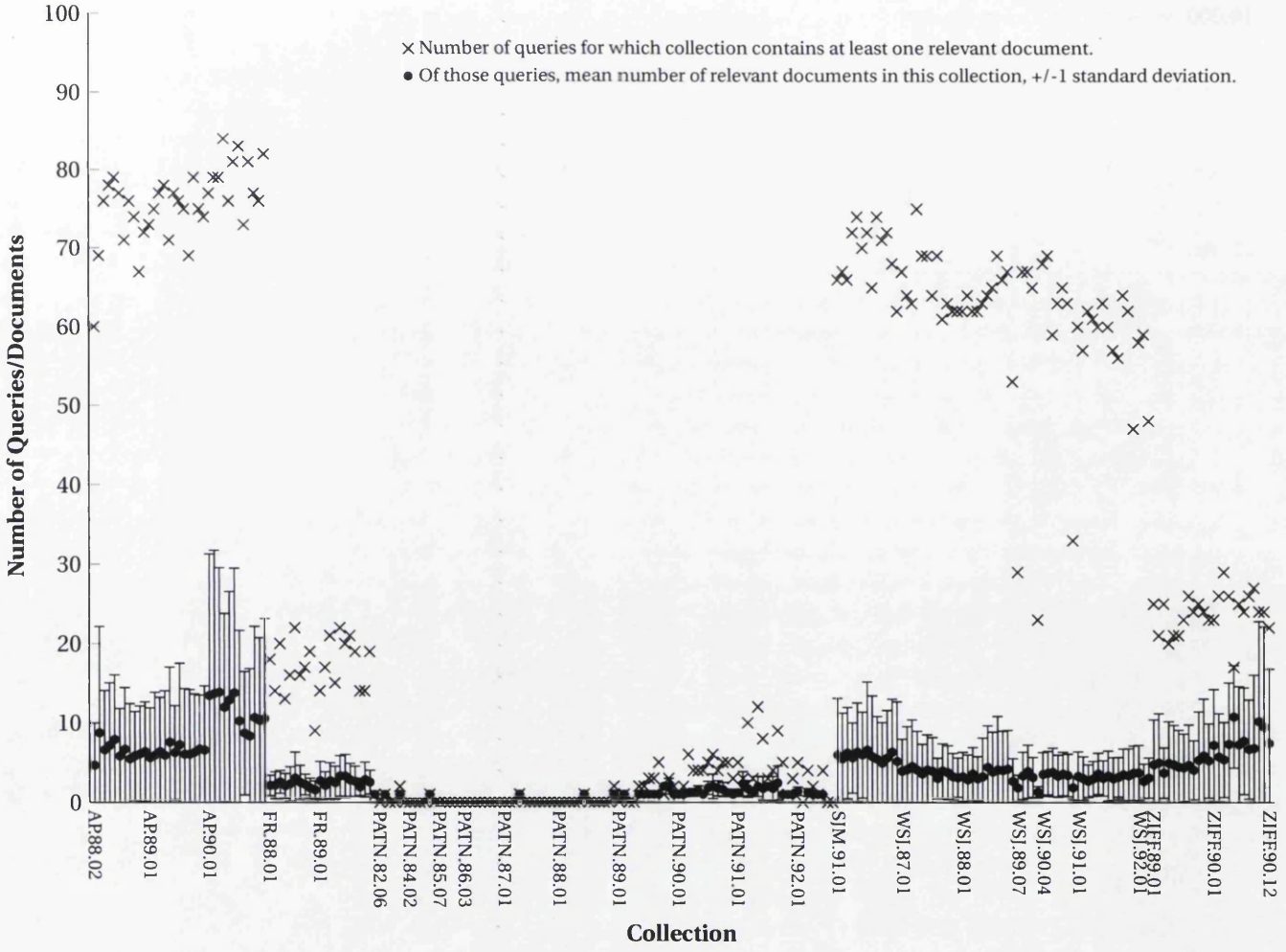


Figure 4.2: The distribution of relevant documents, and the number of queries for which each collection contains relevant documents, in the SYM-236 test data set (reproduced from [43]).

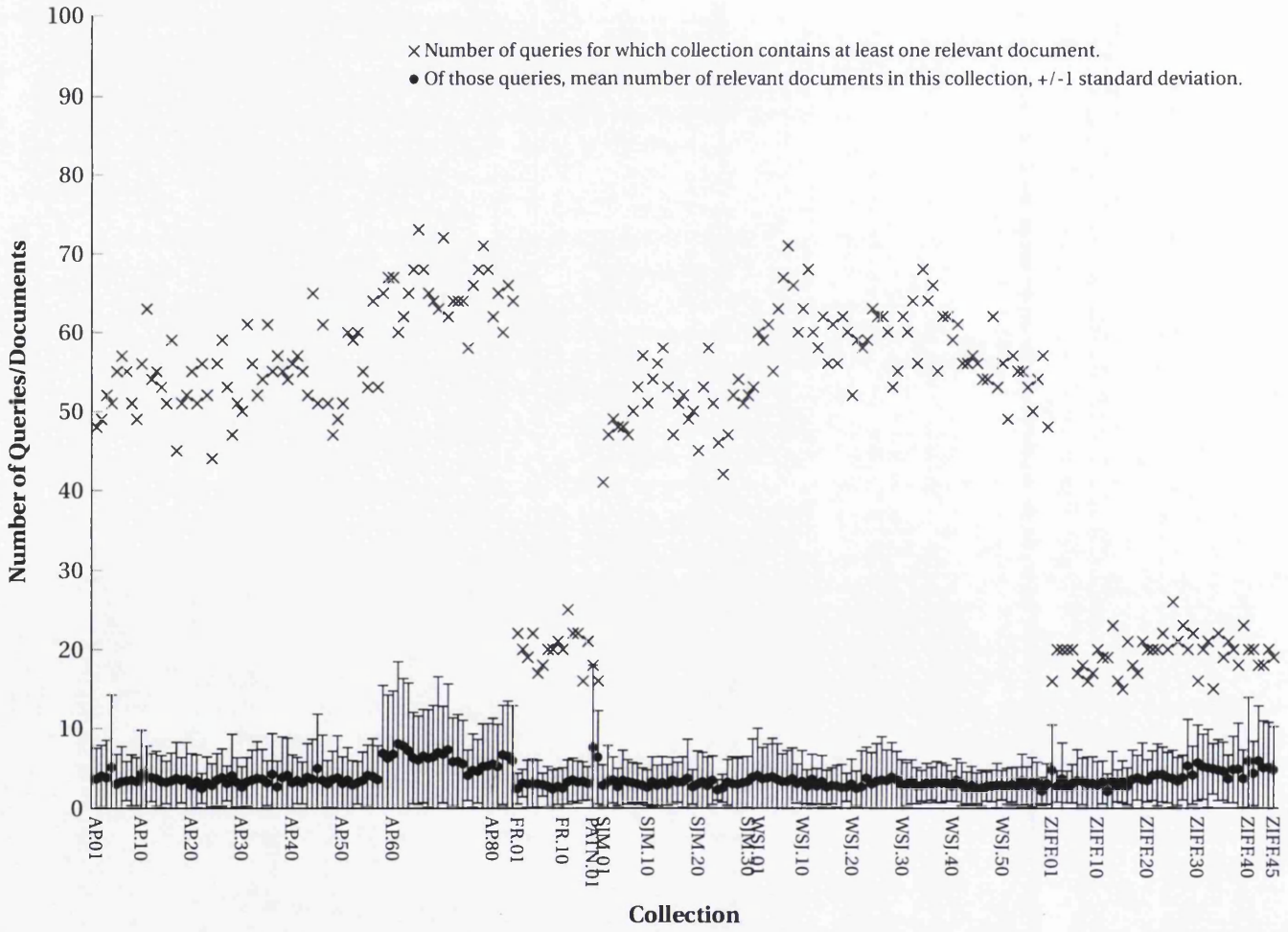


Figure 4.3: The distribution of relevant documents, and the number of queries for which each collection contains relevant documents, in the UDC-236 test data set (reproduced from [43]).

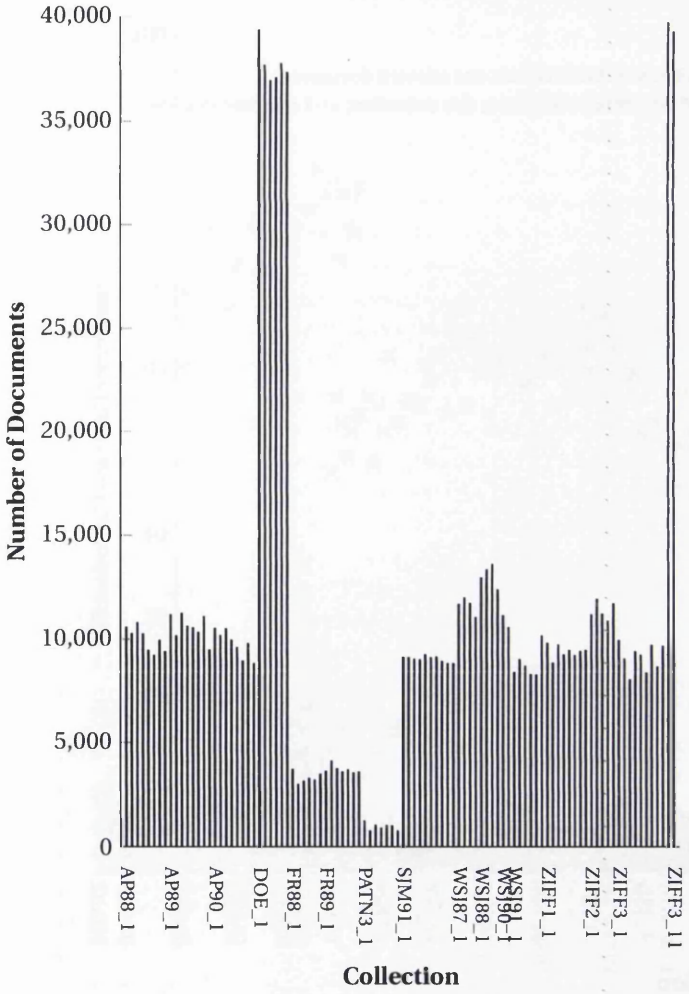


Figure 4.4: The distribution of documents within collections in the UBC-100 test data set (reproduced from [43]).

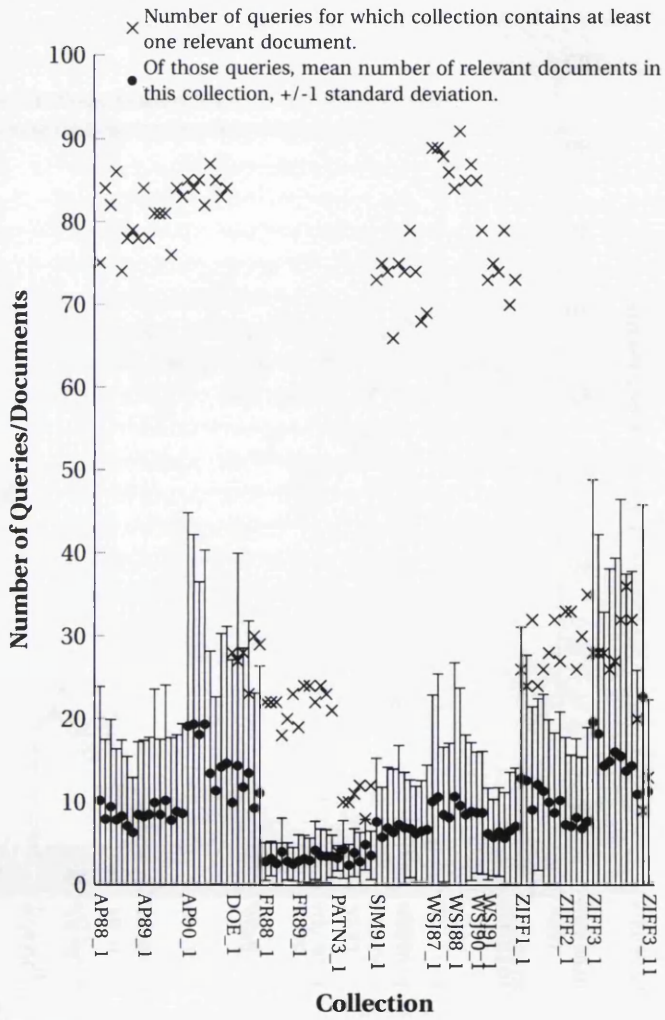


Figure 4.5: The distribution of relevant documents, and the number of queries for which each collection contains relevant documents, in the UBC-100 test data set (reproduced from [43]).

4.1.4 2LDB-60COL ("Representative")

The 2LDB-60COL test data set was specified by Si and Callan [49], and was used in the evaluation of their ReDDE collection selection algorithm. They also specified two additional test data sets, that we use for our experiments, discussed in the following sections. These test data sets have been used in many subsequent, independent studies [50, 47, 56].

The motivation of the 2LDB-60COL, and the following test data sets, was to investigate the effect of environments containing many "small" and a few "very large" collections [49]. The three test data sets investigate the effect of different types of very large collections; each data set is formulated from the UBC-100 data set, described in Section 4.1.3.

The first data set, also referred to as "Representative", is built by first sorting the collections in UBC-100 by alphabetical order. Two large databases (called LDB1 and LDB2) are then formed by grouping together every fifth collection, starting with the first collection, and every fifth collection, starting with the second collection. The remaining 60 collections remain as they were.

From Figure 4.6 (showing the number of documents in each collection), we observe that the test data set contains many small collections of similar size, and the two very large collections are around five times larger than the other collections.

Figure 4.7 gives the distribution of relevant documents within the collections. The two large collections contain at least one relevant document for every query; indeed, on average these collections have over 90 relevant documents per query: a much larger number than any of the smaller collections. As such, an optimal ranking of collections in this test data set may mimic the Size-Based Ranking baseline: on this test data set, ranking the larger collections highly may be a good strategy.

4.1.5 AP-WSJ-60COL ("Relevant")

The AP-WSJ-60COL test data set contains two very large collections called APALL and WSJALL. These collections were formulated by respectively combining the documents from all of the Associated Press collections, and all of the Wall Street Journal collections (from UBC-100). The remaining 60 collections from UBC-100 remain unchanged [49]. The sizes of each collection in this data set are displayed in Figure 4.8.

Since the Associated Press and Wall Street Journal collections contain a high concentration of relevant documents, this is also the case for the APALL and WSJALL collections, as shown in Figure 4.9. In addition, the majority of the query topics target these collections. As is the case for the 2LDB_60COL test data set, an algorithm that tends to rank large collections highly may perform well on this test data set.

4.1.6 FR-DOE-81COL ("Non-Relevant")

Following the same strategy used to produce the AP-WSJ-60COL test data set, FR-DOE-81COL contains two large collections called FRALL and DOEALL. These collections comprise all of the Federal Register and Department of Energy collections, respectively, from the UBC-100 test data set. The other 81 collections from UBC-100 remain unchanged [49].

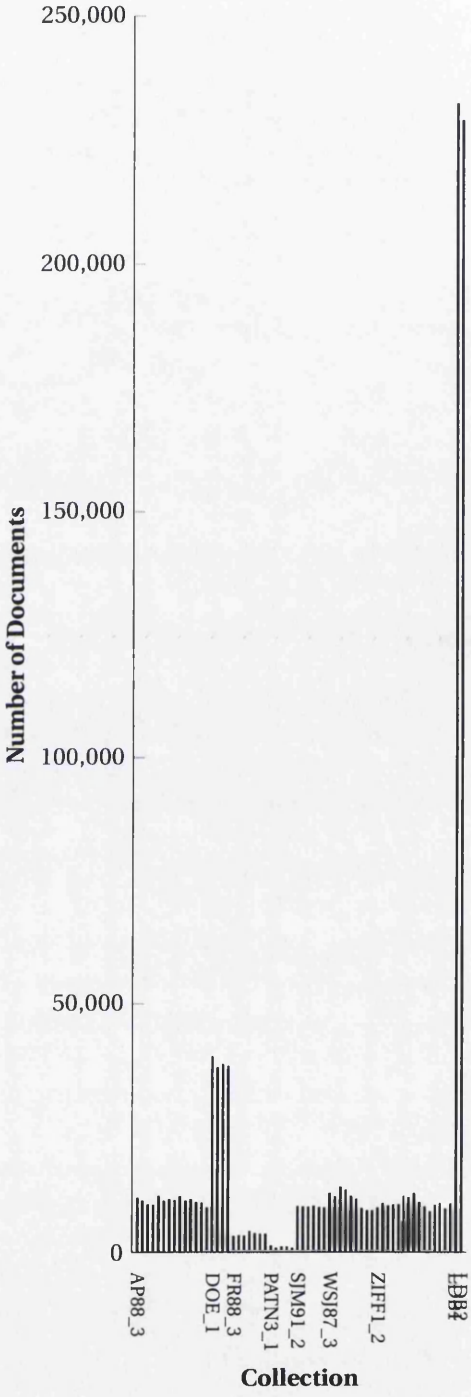


Figure 4.6: The distribution of documents within collections in the 2LDB-60COL test data set.

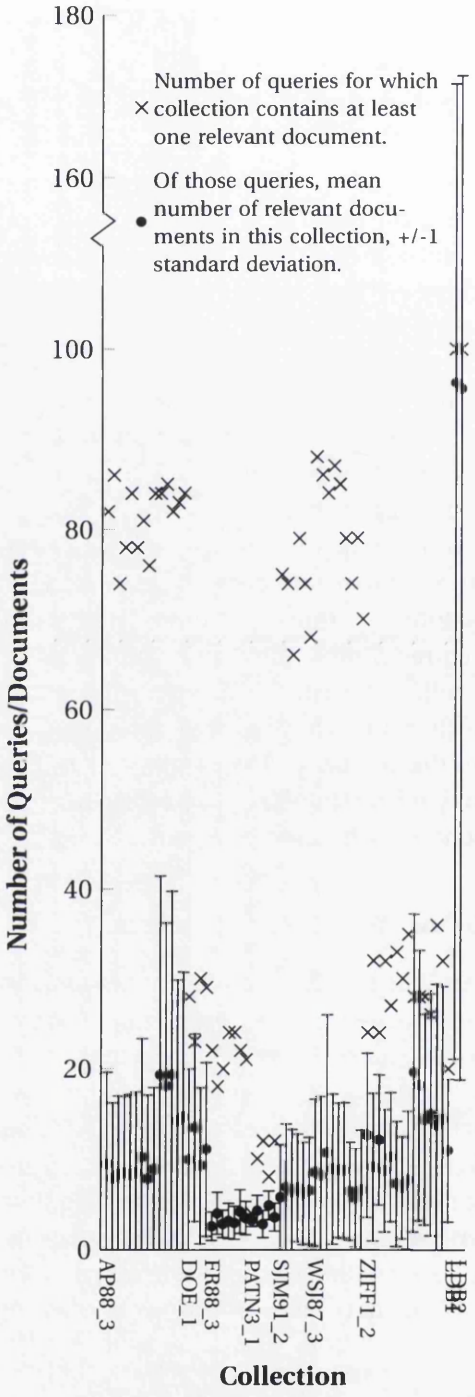


Figure 4.7: The distribution of relevant documents, and the number of queries for which each collection contains relevant documents, in the 2LDB-60COL test data set.

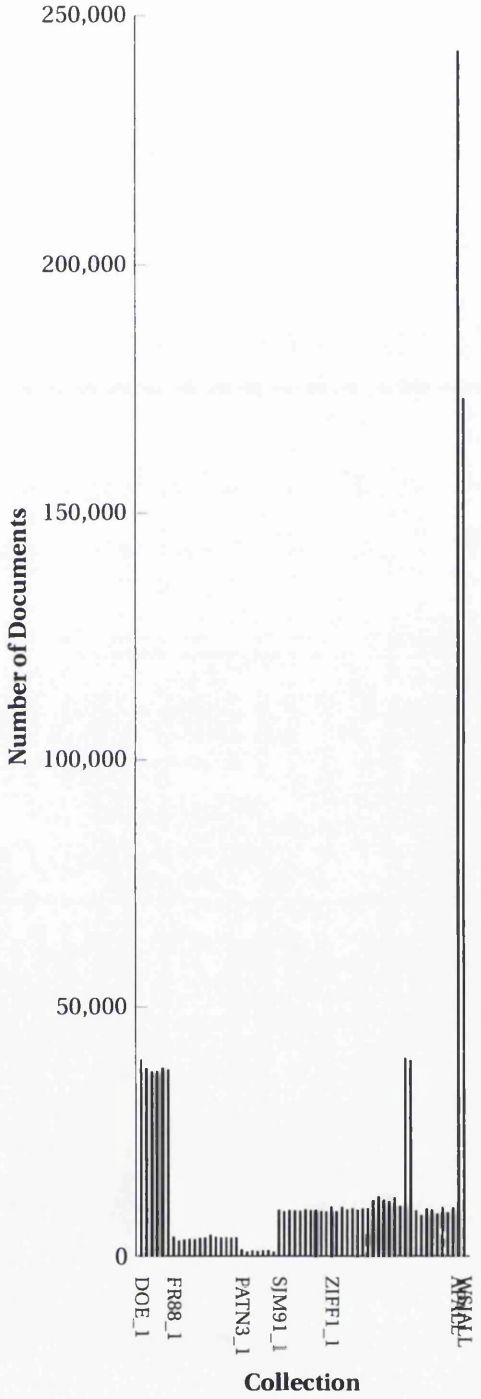


Figure 4.8: The distribution of documents within collections in the AP-WSJ-60COL test data set.

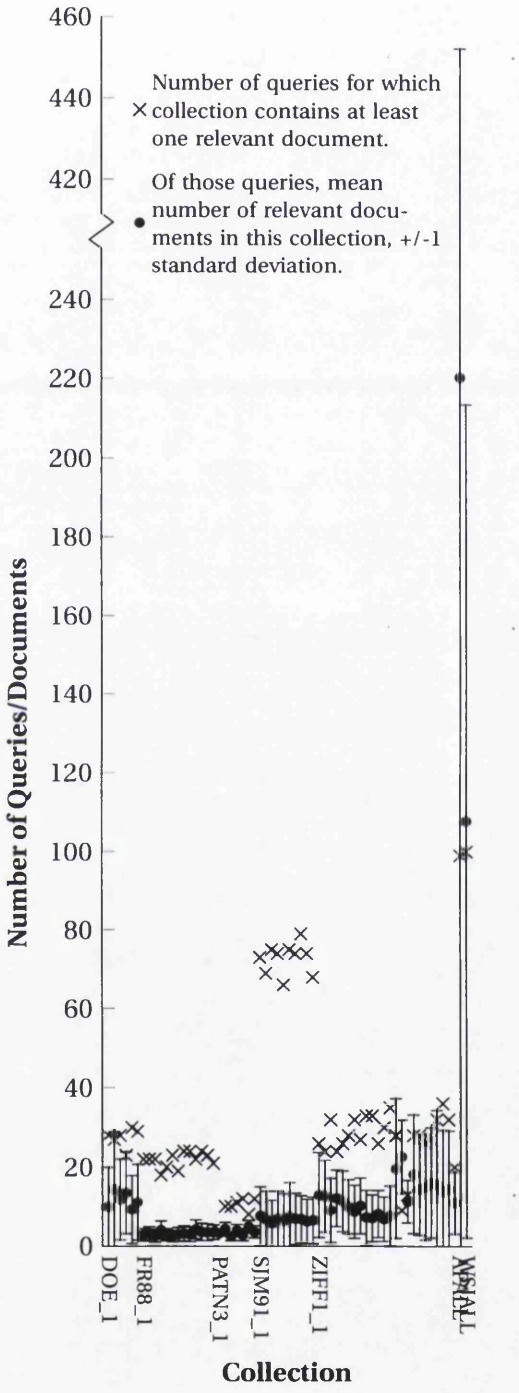


Figure 4.9: The distribution of relevant documents, and the number of queries for which each collection contains relevant documents, in the AP-WSJ-60COL test data set.

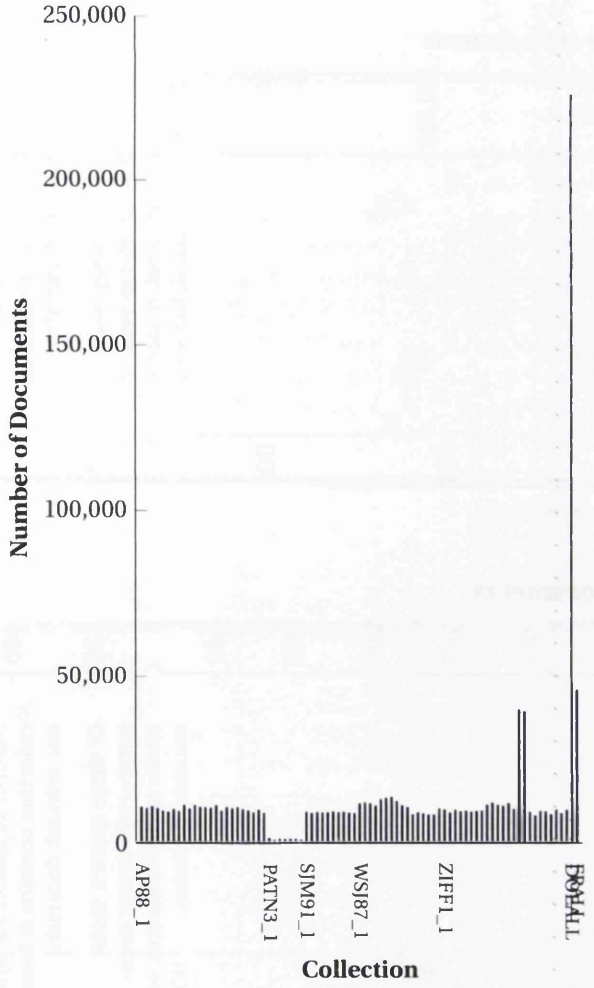


Figure 4.10: The distribution of documents within collections in the FR-DOE-81COL test data set.

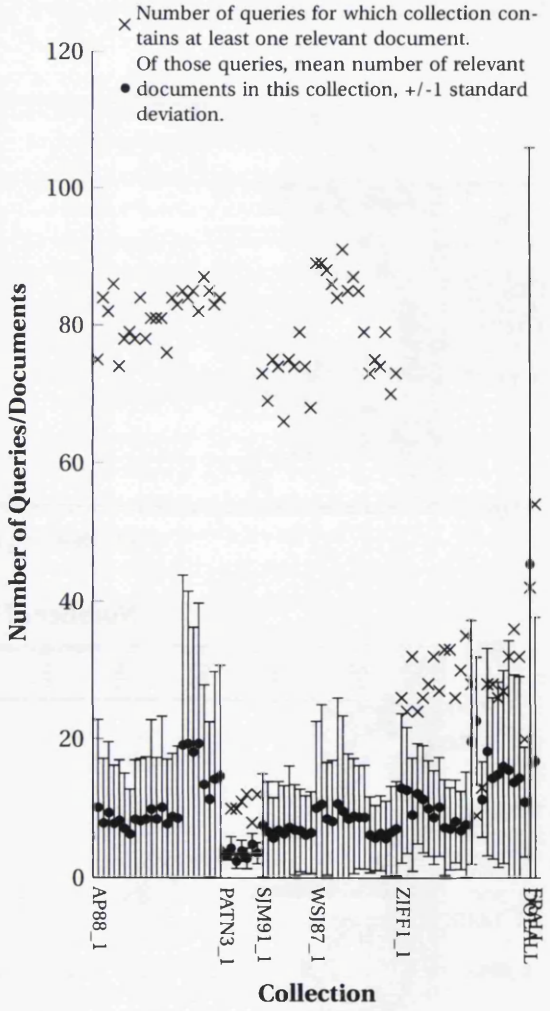


Figure 4.11: The distribution of relevant documents, and the number of queries for which each collection contains relevant documents, in the FR-DOE-81COL test data set.

While FRALL and DOEALL are larger than the other collections in this data set (observed in Figure 4.10), they have a fairly low density of relevant documents, as shown in Figure 4.11. As such, an algorithm that tends to select larger collections over smaller ones may perform poorly on this data set.

4.1.7 Queries

In selecting the query topics to use with the TREC-based data sets, we follow the work of Powell [43]: due to the coverage of relevance judgements across the three TREC discs, topics 51-150 are used in experiments; other topics do not have relevance judgements for all collections used in our test data sets.

Again, following Powell [43], the chosen topics are formulated into two query sets: short (Q_s) and long (Q_l). Short queries range in length from one to six terms, with an average of 3.35 terms; they are built from the Title field of the TREC topics. The Concepts field of the TREC topics is used to form the long queries, which range in length from three to 103 terms, with an average length of 23.63 terms. In this way, we can investigate the effect of query length on the performance of the algorithms being tested.

For reference, Figure B.7 in Appendix B shows the number of relevant documents associated with each of the TREC topics used. We note that, from the previous graphs showing distribution of relevant documents amongst collections, the queries are generally biased towards the Associated Press, Wall Street Journal and San Jose Mercury News sources. This may be because they are the largest original sources, and tend to be fairly generalist.

4.2 Building Realistic Test Data from Open Access Repositories

As discussed in the previous section, in the domain of collection selection, the conventional approach to algorithm evaluation is to utilise TREC-based test data sets, which include synthetic groupings (collections) of documents. These provide a controlled environment in which the specific attributes of algorithms can be investigated; however, they are not consistent with the intended operational environment. Investigating the performance of algorithms on realistic test data is important and valuable: it enables us to be confident that the performance and behaviour observed in tests transfers to the operational environment.

Using real collection data instead of TREC-based collections for testing presents some challenges, such as: gathering document data, compiling test queries, and arriving at document relevance judgements. Indeed, such is the difficulty of compiling an appropriate test data set from real data, previous attempts [22, 63, 56] have resulted in data sets comprising only a few collections. Consequently, although the collection data itself is realistic, the number of collections used is not. As such, there is evidently scope for improvement.

As part of our evaluation of algorithms for collection suggestion, our focal interest is on the construction and development of a large-scale test data set, built from real collection data. To this end, we present two test data sets, built from open access repository data: *Initial Test Data (ITD)* and *Refined Test Data (RTD)*. The ITD set is a preliminary test set, created as a proof of concept, and to support the development of our own algorithm for collection

suggestion. The RTD set is a much larger data set, built following a more systematic approach, and drawing on previous and accepted practice where appropriate; with this, we can verify our findings from the ITD set.

In the following sections, we discuss our technical approach to gathering and managing data from real collections, and present specific details of the composition of each of the test data sets.

4.2.1 Gathering and Processing Repository Data

In experiments involving ranking collections, a test data set that is consistent with the intended operational environment should contain a suitably large number of collections of varying size. In addition, there should be both specialist and generalist collections, covering a range of subject areas.

The obvious strategy for compiling such a large and diverse test data set for collection suggestion experiments is to use a sample of real digital collections. Thus, our test data will incorporate precisely the type of collections we aim to serve, and is therefore both realistic and representative of the operational environment.

However, gathering data from real collections presents a challenge: there are few protocols that facilitate the systematic retrieval of material from a given collection. For example, the Z39.50 [1] protocol supports search and (metadata) record retrieval from remote databases. However, using this protocol to gather collection contents is impractical. Since it facilitates search, probe queries would be required to obtain the records within each collection. Fortunately, a viable alternative does exist: the Open Archives Initiative's Protocol for Metadata Harvesting (OAI-PMH) [35].

OAI-PMH was developed specifically to enable digital repositories to expose metadata records for the resources they contain, and therefore support the dissemination of scholarly work. A client can access the metadata at a repository by submitting HTTP GET requests. Responses to the requests are given as well formed XML documents, making it easy to process the data returned.

The protocol is widely implemented by institutional repositories and digital collections: precisely the types of collections that a collection suggestion search service would aim to target. Two primary suppliers of digital repository software, DSpace[3] and EPrints[4], have implemented support for OAI-PMH. Locating repositories that implement the protocol is straightforward, as there are many lists of registered data providers; for example, the Open Archives Registered Data Providers list[5], and OpenDOAR[6]. For these reasons, we used OAI-PMH to harvest collection data for our realistic test data sets.

A consequence of using OAI-PMH, is that rather than retrieving full document text from repositories, we are only able to access metadata (however, this would also be true of the Z39.50 protocol, were it otherwise suited to our requirements). However, this is no bad

[3] http://www.dspace.org/
[4] http://www.eprints.org/
[5] http://www.openarchives.org/Register/BrowseSites
[6] http://www.opendoar.org/

thing: harvesting metadata offers smaller time and storage costs than gathering full document texts, making the building of a large-scale operationally realistic test data set more practical. In addition, accessing full document texts for indexing purposes may not always be possible anyway: some repositories do not allow free access to full document content, but do export metadata, to enable users to find the documents, in case they wish to pay for them. For example, documents within the ACM Digital Library may be found via a general purpose search engine, but many can only be downloaded with a subscription to the service.

Repositories that implement OAI-PMH are required (as a minimum) to return records in the Dublin Core metadata format [35]. This offers a basic description of objects, and includes fields such as: Title, Description, Subject and Type [31]. However, repositories may also implement additional metadata formats, if they require more complex resource descriptions.

Since OAI-PMH repositories are guaranteed to implement Dublin Core, we harvest documents in this format to construct our ITD and RTD sets. As such, we can be sure which fields are available to use for indexing purposes. An interesting side effect of representing collections with document metadata is that, as well as testing algorithms, we can also investigate the optimal and minimal metadata required to accurately rank collections.

To facilitate this investigation we construct two term indexes from the metadata: one containing terms from the Title metadata field, and one containing terms from both the Title and Description fields. We expect these fields to be present in the majority of records in a repository, and anticipate them to provide descriptive terms that accurately reflect the topic of the actual resource. Finally, when indexing the metadata we remove stop words, remove punctuation, and stem terms with the Snowball English stemmer:[7] processing terms in this way is common information retrieval practice [2].

In the following sections we provide further discussion of the construction of the Initial and Refined Test Data sets, and consider the properties of each. Section 4.2.4 discusses our strategy for generating document relevance judgements for these data sets.

4.2.2 Initial Test Data (ITD)

The Initial Test Data (ITD) set is a preliminary test set, created to investigate the feasibility of using OAI-PMH and metadata to form a test data set, and to conduct pilot experiments.

To create a realistic test environment, a suitable sample of collections of different types is desirable. Digital collections vary in their scope and subject coverage: some are specialist and cover only a single topic, while others are more generalist, and address several subjects (for example, institutional repositories).

To select a representative sample of collections for the ITD set, we browsed the Open Archives Initiative's list of Registered Data Providers.[8] This enabled us to select collections with varying topic coverage and size.

Table 4.2 lists the repositories that were harvested for this data set, and the number of documents they contained at the time (October 2010) they were harvested. Although the

[7] `http://snowball.tartarus.org/`

[8] `http://www.openarchives.org/Register/BrowseSites`

Table 4.2: Initial Test Data set collections.

#	Repository Name	Number of Documents
1	DCMI Proceedings	16
2	Annals of Genealogical Research	20
3	The University of Brighton Repository	381
4	Caltech Computer Science Technical Reports	441
5	Oxford Eprints	555
6	University of Chester Digital Repository	816
7	Bristol Repository of Scholarly Eprints (ROSE)	1504
8	Digital Repository of the University of Wolverhampton	1644
9	University of Exeter	1685
10	Cardiff ePrints Caerdydd	4878
11	Durham Research Online	6556
12	CWI	11936
13	Enlighten (Glasgow)	32438
14	Deep Blue at the University of Michigan	61258
15	DSpace at Cambridge	206271
16	PubMed Central	1970516

Table 4.3: Summary statistics for the open access repository test data sets.

		Documents Per Collection				Avg. Doc. Length	
Test Set	**Collections**	**Total**	**Min.**	**Avg.**	**Max.**	**Titles**	**Titles and Desc.**
ITD	16	2300915	16	143807	1970516	7.40	53.96
RTD	100	1204048	7	12040	196224	6.72	45.62

number of collections is small, 16 in total, they do vary considerably in size: the smallest has only 16 documents, while the largest contains nearly two million documents.

Table 4.3 provides a summary of the statistics associated with the data set. We note that the length of documents in this data set are shorter on average than those in the TREC-based data sets discussed in Section 4.1. This is because in this instance, the documents comprise metadata, rather than full document text.

In a test data set, the test queries are just as important as the documents: they should be relevant to the documents in the data set, as there is little benefit to testing a system on queries that are unlikely to match any relevant material. For evaluation in the collection suggestion scenario we require queries that challenge the algorithms: queries that aim to target specialised collections, as well as queries that are much broader, and could match several collections.

The ITD set contains a set of 50 test queries (listed in Section B.1.1 of Appendix B), a quantity commonly used in collection selection experiments [6, 66]. The queries are in-

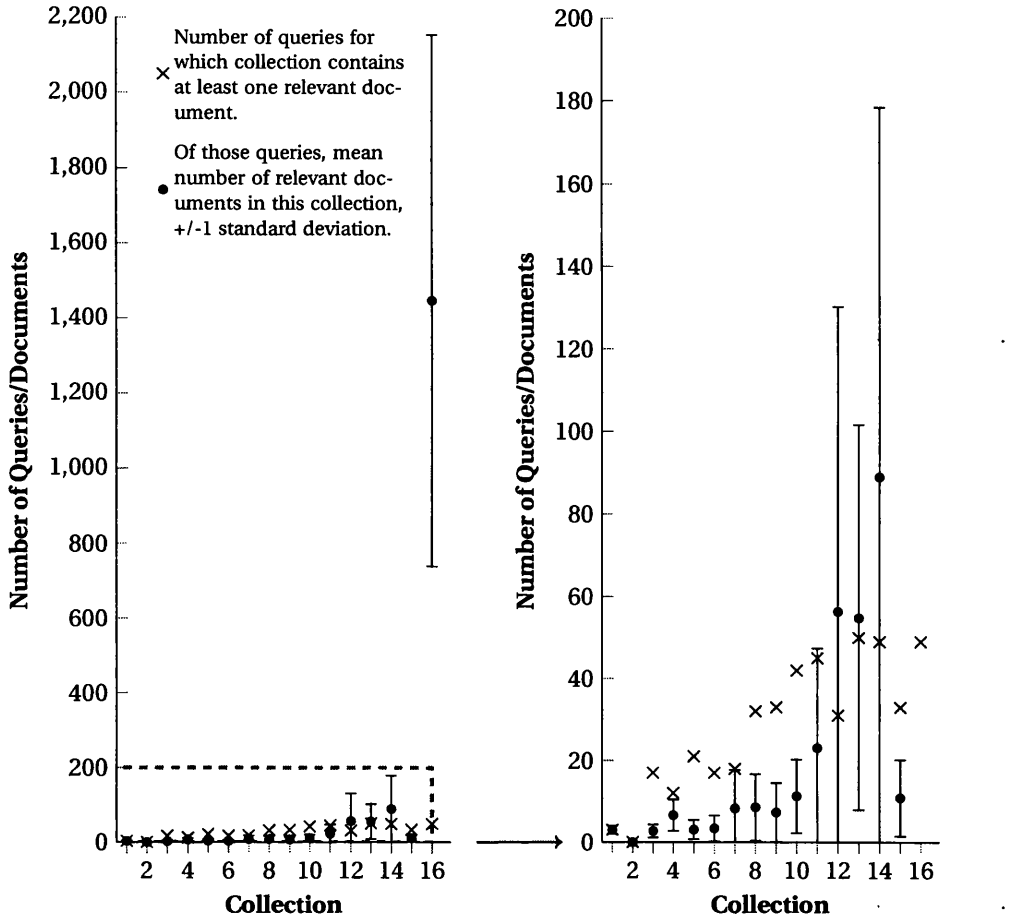


Figure 4.12: The distribution of relevant documents, and the number of queries for which a collection contains relevant documents, for the ITD test data set, on the Title metadata. The graph on the right is a magnification of the lower portion of the graph on the left.

formed by the harvested repositories: some queries consist of document titles (for example, "fluorescence imaging endoscope for early detection of gastrointestinal malignancy"), and are therefore intended to target specific collections. Other queries were assembled by observing terms present in the Set and Keyword metadata fields, across all collections, and devising general descriptions of a wide subject area (for example, "paediatric dentistry" or "culture in tudor england").

The queries range in length from one to ten terms, with an average length of 3.86 terms. Although short, it is not unreasonable: Silverstein [51] has shown that web queries often consist of three terms or less.

For reference, Figures B.1 and B.2 in Appendix B show the number of relevant documents associated with each query, depending on whether we are using only document titles, or both titles and descriptions. The graphs in Figures 4.12 and 4.13 show how relevant documents are distributed between the collections, and for each collection, the number of

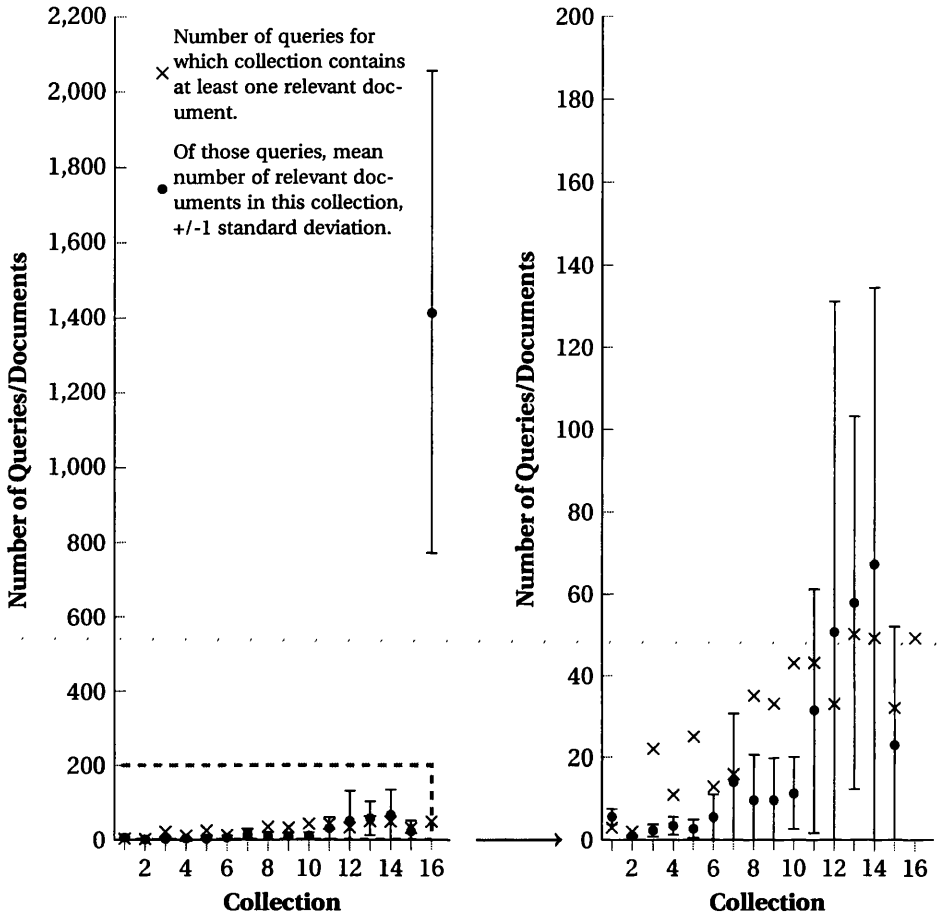


Figure 4.13: The distribution of relevant documents, and the number of queries for which a collection contains relevant documents, for the ITD test data set, on the Title and Description metadata. The graph on the right is a magnification of the lower portion of the graph on the left.

queries for which it has some relevance. These graphs allow us to observe the properties of the data sets: specifically, the spread of relevant documents and scope of the queries.

We observe that the larger collections tend to be relevant to more queries, and they also contain more relevant documents. The largest collection, PubMed Central, has on average considerably more relevant documents than the others. This may be a side effect of our technique for automatically generating relevance judgements (discussed later in Section 4.2.3), as few of our queries are specifically targeted at this collection. Due to the size of PubMed Central, it may have more documents that have partial relevance to the query, causing it to monopolise the relevant documents.

The presence of the PubMed Central repository may make the ITD set a challenging environment for algorithms: while it is a large collection and may show some relevance to most queries, it will often not be the most appropriate collection. As such, algorithms that

favour large collections with many moderately relevant documents may perform worse than those that choose smaller collections, with documents of higher relevance.

In the following section we present our second open access repository test data set, which contains a more balanced range of collection sizes.

4.2.3 Refined Test Data (RTD)

Our second set of test data built from open access repositories is the Refined Test Data (RTD) set. Its construction followed a more methodical approach than that used for compiling the Initial Test Data set. We will later use the RTD set to verify the results from the ITD set.

To select collections for the RTD set, we utilise the OpenDOAR[9] repository directory, which provides an API to programmatically retrieve data about the (approximately 2000) collections it lists. Using the API we identified English language collections that implement OAI-PMH. After processing the list of collections to remove any we found to be inaccessible, we selected every n^{th} collection, to give a sample of 100 collections. The selected collections range in size from 7 to 196224 documents; full summary statistics are given in Table 4.3, while the complete list of collections and their sizes is given in Table B.1, in Appendix B.

As with the ITD set, the test queries are formed from the contents of the test collections, as this ensures our queries are relevant to the collections. Such an approach is taken by TREC, where experts peruse documents in the test data in order to build query topics [25]. However, rather than manually writing queries, we use an automated approach.

Using the Kea[10] key phrase extraction tool, we identify key phrases in each metadata record (which consist of Title and Description metadata fields). We construct a term index from the key phrases, and select the second quartile of most frequent terms as *seed terms*. We use the second quartile of terms as they are likely to be common, but discriminatory; terms from the first quartile are likely to occur in most collections, while terms from the third and fourth quartile may be too specialist. Each seed term is submitted to the Yahoo! Related Suggestions[11] web service to generate query suggestions. From these query suggestions we take two random samples: one set of 50 queries, and another of 200 (see Section B.2.2 in Appendix B for lists of these queries). The chosen queries are between one and seven terms in length, with an average of 2.39 terms. Again, this is short, but these are based on real user queries, submitted to the Yahoo! search engine.

The graphs in Figures 4.14, 4.15, 4.16 and 4.17 show the distribution of relevant documents amongst the collections, and the number of queries for which a collection is relevant, for both sets of queries and both term indexes. As with the ITD set in the previous section, we see that as collection size increases, so does the number of queries for which the collection is relevant, and the number of relevant documents it contains. This is not surprising: a large collection has the opportunity for a more varied vocabulary, and therefore can match more query terms. In contrast to ITD, the distribution of documents, while not even amongst collections, is much more balanced: there is no single collection that has a considerably larger share of relevant documents.

[9] `http://www.opendoar.org/`

[10] `http://nzdl.org/Kea`

[11] `http://developer.yahoo.com/search/web/V1/relatedSuggestion.html`

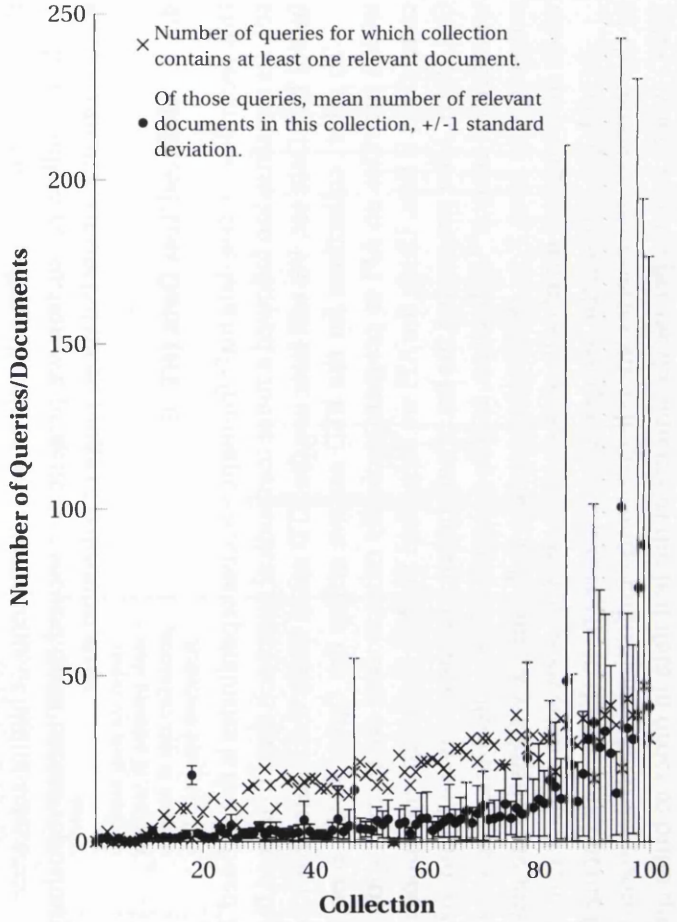


Figure 4.14: The distribution of relevant documents, and the number of queries for which a collection contains relevant documents, for the RTD test data set, on the set of 50 queries and Title metadata.

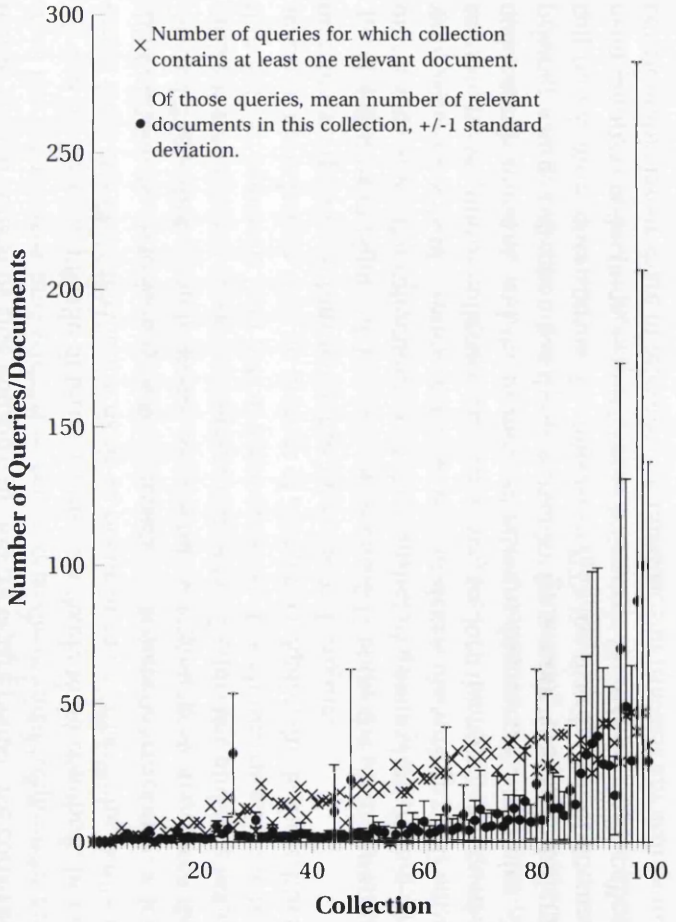


Figure 4.15: The distribution of relevant documents, and the number of queries for which a collection contains relevant documents, for the RTD test data set, on the set of 50 queries and Title and Description metadata.

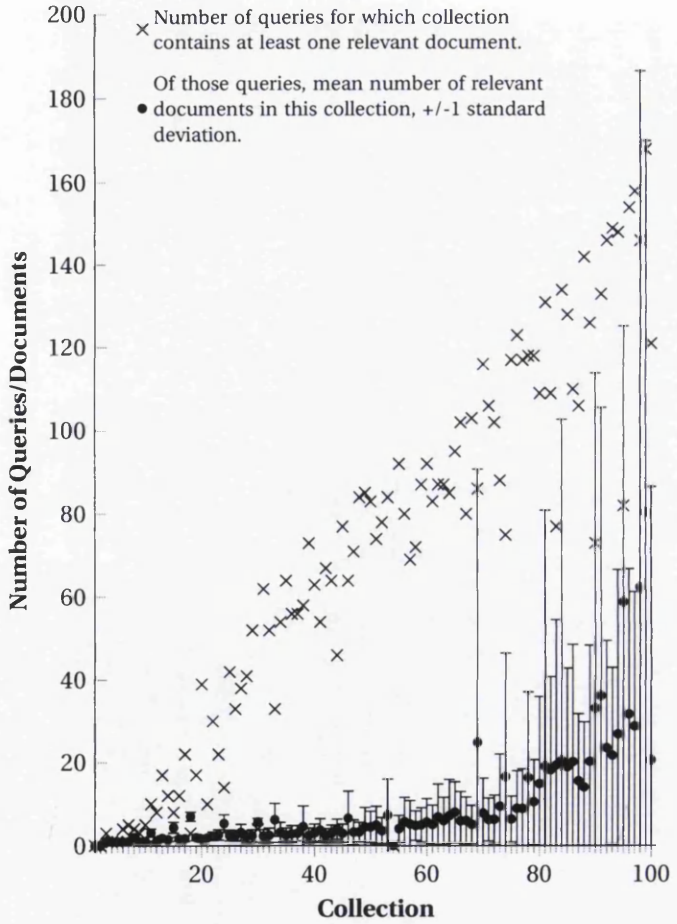


Figure 4.16: The distribution of relevant documents, and the number of queries for which a collection contains relevant documents, for the RTD test data set, on the set of 200 queries and Title metadata.

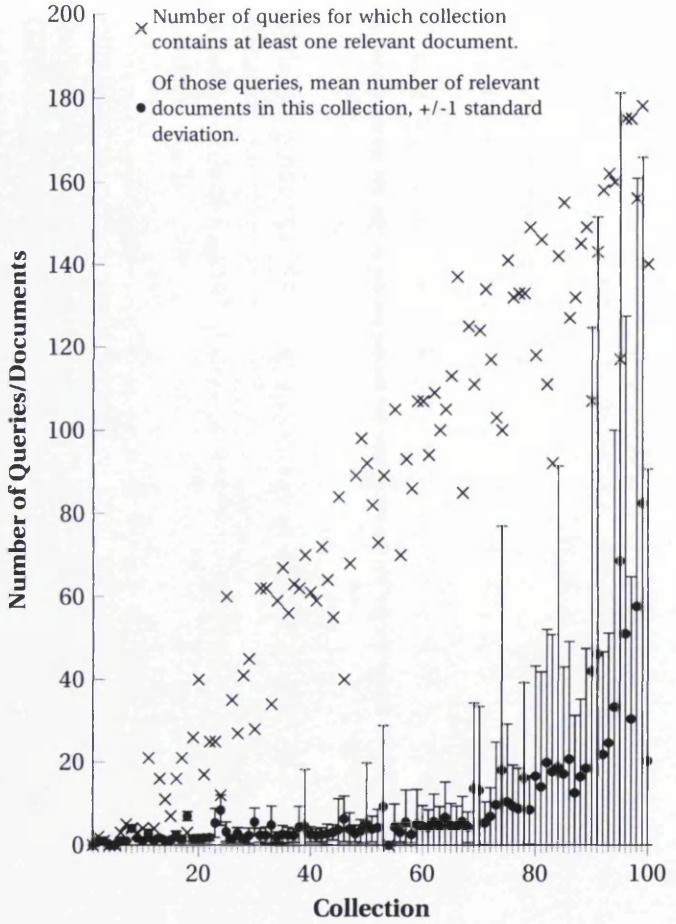


Figure 4.17: The distribution of relevant documents, and the number of queries for which a collection contains relevant documents, for the RTD test data set, on the set of 200 queries and Title and Description metadata.

For reference, Figures B.3, B.4, B.5 and B.6 in Appendix B show the number of relevant documents associated with the queries in the two query sets for the RTD set. The number of relevant documents varies between queries, although on average there appear to be fewer relevant documents per query than in the ITD set. This may be because of a bias towards partially relevant documents in the PubMed Central collection in ITD.

For each of our open access repository data sets, we require document relevance judgements for each query: these enable us to produce an optimal ranking of collections, with respect to the goals of collection suggestion. We discuss the production of the document relevance judgements in the following section.

4.2.4 Relevance Judgements

In order to facilitate the evaluation of algorithms over our open access repository test data sets, we require a set of relevance judgements (the documents that are relevant) for each query. These relevance judgements are used to generate an *optimal* ordering of collections, which the algorithms should aspire to match. We discuss the generation of the optimal ranking in Section 3.2 of Chapter 3.

Test data sets comprising TREC documents have relevance judgements associated with them. However, the two open access repository data sets are built from real repositories, and as such we do not have relevance judgements linking documents to queries. For the ITD and RTD sets, we apply a novel, programmatic approach, to produce surrogate document relevance judgements.

As we are developing a test environment that is a realistic reflection of the likely operational setting, we are dealing with sets of test collections that total millions of documents. Assessing each document for relevance to every test query is impractical: not only for our work, but also for the original TREC experiments [25]. As discussed in Chapter 2, to produce relevance judgements TREC used the *pooling method*, whereby for each topic, each participating document retrieval system submits its top n ranked documents into a topic results pool. After duplicate documents are removed, the pool of results is passed to human assessors, who decide which of the documents are actually relevant to the topic [25]. We draw on this technique to create relevance judgements for our data. However, we forgo the human assessor element, and instead opt for a fully automated strategy; given the size of our data sets, assessing even a pooled sample of documents for every query would be very labour intensive, and impractical within the scope of this work.

Using the Apache Lucene[12] search engine library, we build a document index for each open access repository data set. For each query the documents are ranked using two algorithms (facilitated by the Apache Lucene library): BM25 [42] and the Lucene search algorithm [15]. Each returns the top n relevant documents (where n is 0.1% of the total number of documents harvested for the data set). The final set of relevant documents for each query is produced by taking the intersect of the documents returned by the two algorithms. Thus, a document is classed as relevant if both algorithms agree it is relevant.

[12] `http://lucene.apache.org/`

A comprehensive description of how we arrived at this technique, through empirical means, is given in Appendix C. We note here that while we have taken a considered approach to programmatically generating document relevance judgements, the technique has limitations. We rely on the effectiveness of two document ranking algorithms; while these algorithms are well established, their effectiveness is not optimal, as is evidenced in Appendix C.

4.2.5 Repository Administration Tool

The use of test data sets built from open access repositories requires apparatus to facilitate the collection and management of this data. We have constructed such apparatus using the Java programming language, and a MySQL database. Figure 4.18 shows the primary interface for this tool, which provides the following functionality:

Add repository: add the details (such as name and URL) of a repository to the database, so that it may be harvested in the future.

Delete repository: remove a repository, and all its associated data, from the database.

Full harvest: contact the repository via OAI-PMH, and retrieve the metadata (titles and descriptions) for every document currently held in the collection. Upon completion of harvesting, the metadata is processed to produce a term index.

Update harvest: contact the repository via OAI-PMH, and retrieve metadata for documents added, updated or removed since the last harvest of data. The associated term index is rebuilt to reflect changes.

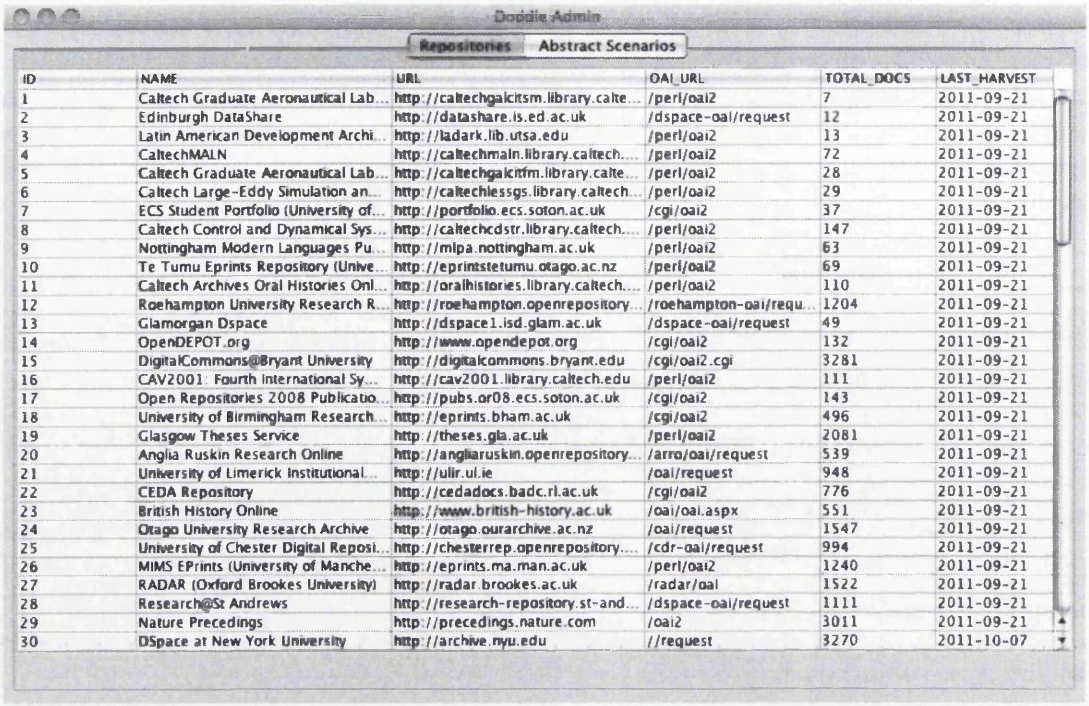
4.3 Summary

In seeking an effective algorithm for the collection suggestion task, we require a suitable methodology and appropriate test data, to support the evaluation of algorithms with respect to the task.

As we have discussed in this chapter, a good test data set should be reproducible, representative of the operational environment, and consistent with the expectations of information retrieval research. It is clear, from previous work in the related domain of collection selection, that it is difficult to produce a single test data set that exhibits these properties.

For example, the common approach of creating synthetic document collections from the TREC document corpus offers a controlled test environment, where experiments are easily reproducible by others. However, the synthetic collections built from TREC data are generally not representative of a realistic operational environment: the TREC corpus contains primarily news articles, and so the collections generated tend to be small, with many containing documents on a mix of unrelated topics.

Previous work in developing test data sets from real collection data has also been deficient: only a small number of real collections have been used. This is likely due to the



ID	NAME	URL	OAI_URL	TOTAL_DOCS	LAST_HARVEST
1	Caltech Graduate Aeronautical Lab...	http://caltechgalcitsm.library.calte...	/perl/oai2	7	2011-09-21
2	Edinburgh DataShare	http://datashare.is.ed.ac.uk	/dspace-oai/request	12	2011-09-21
3	Latin American Development Archi...	http://ladark.lib.utsa.edu	/perl/oai2	13	2011-09-21
4	CaltechMALN	http://caltechmaln.library.caltech....	/perl/oai2	72	2011-09-21
5	Caltech Graduate Aeronautical Lab...	http://caltechgalcitfm.library.calte...	/perl/oai2	28	2011-09-21
6	Caltech Large-Eddy Simulation an...	http://caltechlessgs.library.caltech...	/perl/oai2	29	2011-09-21
7	ECS Student Portfolio (University of...	http://portfolio.ecs.soton.ac.uk	/cgi/oai2	37	2011-09-21
8	Caltech Control and Dynamical Sys...	http://caltechcdstr.library.caltech....	/perl/oai2	147	2011-09-21
9	Nottingham Modern Languages Pu...	http://mlpa.nottingham.ac.uk	/perl/oai2	63	2011-09-21
10	Te Tumu Eprints Repository (Unive...	http://eprintstetumu.otago.ac.nz	/perl/oai2	69	2011-09-21
11	Caltech Archives Oral Histories Onl...	http://oralhistories.library.caltech....	/perl/oai2	110	2011-09-21
12	Roehampton University Research R...	http://roehampton.openrepository...	/roehampton-oai/requ...	1204	2011-09-21
13	Glamorgan Dspace	http://dspace1.isd.glam.ac.uk	/dspace-oai/request	49	2011-09-21
14	OpenDEPOT.org	http://www.opendepot.org	/cgi/oai2	132	2011-09-21
15	DigitalCommons@Bryant University	http://digitalcommons.bryant.edu	/cgi/oai2.cgi	3281	2011-09-21
16	CAV2001: Fourth International Sy...	http://cav2001.library.caltech.edu	/perl/oai2	111	2011-09-21
17	Open Repositories 2008 Publicatio...	http://pubs.or08.ecs.soton.ac.uk	/cgi/oai2	143	2011-09-21
18	University of Birmingham Research...	http://eprints.bham.ac.uk	/cgi/oai2	496	2011-09-21
19	Glasgow Theses Service	http://theses.gla.ac.uk	/perl/oai2	2081	2011-09-21
20	Anglia Ruskin Research Online	http://angliaruskin.openrepository....	/arro/oai/request	539	2011-09-21
21	University of Limerick Institutional...	http://ulir.ul.ie	/oai/request	948	2011-09-21
22	CEDA Repository	http://cedadocs.badc.rl.ac.uk	/cgi/oai2	776	2011-09-21
23	British History Online	http://www.british-history.ac.uk	/oai/oai.aspx	551	2011-09-21
24	Otago University Research Archive	http://otago.ourarchive.ac.nz	/oai/request	1547	2011-09-21
25	University of Chester Digital Reposi...	http://chesterrep.openrepository....	/cdr-oai/request	994	2011-09-21
26	MIMS EPrints (University of Manche...	http://eprints.ma.man.ac.uk	/perl/oai2	1240	2011-09-21
27	RADAR (Oxford Brookes University)	http://radar.brookes.ac.uk	/radar/oai	1522	2011-09-21
28	Research@St Andrews	http://research-repository.st-and...	/dspace-oai/request	1111	2011-09-21
29	Nature Precedings	http://precedings.nature.com	/oai2	3011	2011-09-21
30	DSpace at New York University	http://archive.nyu.edu	//request	3270	2011-10-07

Figure 4.18: Interface for tool to support the management of an open access repository data set.

difficulties associated with gathering collection content, and producing test queries and relevance judgements.

Given these difficulties, to provide a robust and diverse test environment (comprising collections of documents, test queries and associated relevance judgements) for investigating algorithms for collection suggestion, we use a combination of several data sets. Specifically, we use six existing TREC-based data sets (attributed to French and Powell [18, 43] and Si and Callan [49]). Collection attributes varied in these data sets include collection sizes, and distribution of relevant documents amongst the collections.

In addition to using the TREC-based data sets, we have developed two test data sets formulated from data harvested from real digital repositories. While the first of these (the Initial Test Data set) contains only a few collections and is considered as a proof of concept, the second (Refined Test Data) is a large-scale data set (containing both general and domain-specific collections), constructed in a methodical manner. We draw on previous practices and techniques to select collections, and address the challenges of producing queries and associated document relevance judgements.

As such, with this combination of data sets, we evaluate algorithms in both a controlled setting (in which we can identify specific strengths and weaknesses of algorithms) and a realistic setting. Such an approach is prudent: previous work in the collection selection domain has shown algorithm performances can vary greatly over different data sets exhibiting different properties.

In Chapters 6 and 7 we utilise the test data sets discussed here, to conduct a large-scale evaluation of the effectiveness of a variety of algorithms, with respect to collection suggestion. Prior to this, in the following chapter, we discuss the development of our own algorithms for collection suggestion, and measure their performance.

Chapter 5

An Algorithm for Collection Suggestion

In this chapter, we develop a new algorithm, designed specifically for the collection suggestion task. In addition, we test the performance of this algorithm, and that of its component parts and alternative configurations.

5.1 Overview

In Chapter 1, we identified the need for a search service to assist users seeking domain-specific digital collections. Given this need, our work focuses on addressing the fundamental research required, towards developing a search service for collections. Of particular importance for such a service is an effective algorithm that can rank collections with respect to a user's query (in much the same way a traditional search algorithm ranks documents).

As demonstrated in Chapter 2, many algorithms have been developed for other domains, such as collection selection and query performance prediction, that may be effectively applied to collection ranking for the collection suggestion task. As such, towards identifying an optimal algorithm, we thoroughly examine the suitability of these algorithms for collection suggestion, in Chapters 6 and 7.

However, the algorithms within these domains were designed with objectives different to those of collection suggestion in mind. For example, collection selection algorithms aim to emulate a collection ranking in which collections are placed in decreasing order of the number of relevant documents they contain. In query performance prediction, algorithms aim to estimate the quality of document search results: that is, how easy it is to discriminate between the documents in the collection.

The goal of collection suggestion is to identify authoritative collections: those that are *about* the query topic. As such, a highly ranked collection should contain a large number of relevant documents, and these documents should make up a significant proportion of the collection. These criteria are not necessarily represented in the design of the algorithms

discussed in Chapter 2 (however the underlying theories of some algorithms suggest they may be promising candidates for collection suggestion).

To ensure we have an optimal collection ranking algorithm for a collection suggestion search service, a key contribution of this research is to develop a new algorithm, geared specifically towards the objectives of collection suggestion. Thus, we aim to develop an algorithm that exhibits performance superior to that achieved by the existing collection selection and query performance prediction algorithms.

To this end, in this chapter we discuss the development of our algorithm. Specifically, in Section 5.2 we explore the foundations and theories underpinning the algorithm. In Section 5.3 we present the specification of the algorithm,[1] and conduct an initial test of its performance, in Section 5.4. In addition, in Section 5.5, we investigate the performances of the individual components of the algorithm, and experiment with alternative configurations. In this way, we can compare the performance of our algorithm to that of the collection selection and query performance prediction algorithms, with respect to the collection suggestion task, using the strongest variants.

5.2 Algorithm Origin

The algorithm we develop specifically for the collection suggestion task is inspired by and based upon Zobel's criteria for highly ranked collections [66]. While Zobel's work falls into the collection selection domain, the criteria provide a strong representation of the objective of collection suggestion. The criteria state that a collection should be highly ranked, if for each query term:

1. The term occurs in the collection;
2. The term is common in the collection (relative to the other collections);
3. The collection contains a relatively high proportion of documents featuring the term; and
4. There are likely to be documents in the collection in which the term is relatively frequent [66].

In his work, Zobel specifies four algorithms for collection selection (discussed in Chapter 2). Only one of these algorithms appears to embody the criteria: the Skew algorithm represents an implementation of criterion 3. The other algorithms (Cosine Measure, Inner Product and Highest-available Similarity) are similarity-based.

As such, we utilise all four criteria to formulate a new algorithm, and therefore test the underlying theories. We begin by representing the criteria mathematically as follows:

1. ***The term occurs in the collection:***
   A term that does not occur in the collection will simply not contribute to the collection's score. Therefore, no metric is assigned to this criteria.

[1] We have previously published a specification of the algorithm, together with early experimental results, in Dodd et al. [9].

2. ***The term is common in the collection (relative to the other collections):***
Mathematically, we represent the commonness of a term t in collection c by:

$$C_{t,c} = \frac{f_{c,t}}{tokens_c},$$

where:

$f_{c,t}$ is the number of occurrences of term t in collection c; and

$tokens_c$ is the number of terms in collection c.

It follows that the commonness of a query term in a collection, relative to its commonness in the other collections (relative commonness) can be represented by:

$$RC_{t,c} = \frac{C_{t,c}}{\sum_{i=1}^{|C|} C_{t,i}},$$

where:

$|C|$ is the number of collections.

3. ***The collection contains a relatively high proportion of documents featuring the term:***
Mathematically, we represent the proportion of documents in collection c containing a term t by:

$$P_{t,c} = \frac{df_{c,t}}{docs_c},$$

where:

$df_{c,t}$ is the number of documents in collection c containing term t; and

$docs_c$ is the number of documents in collection c.

It follows that the proportion of documents in a collection that contain the query term, relative to the other collections (relative proportion) can be represented by:

$$RP_{t,c} = \frac{P_{t,c}}{\sum_{i=1}^{|C|} P_{t,i}},$$

where:

$|C|$ is the number of collections.

4. ***There are likely to be documents in the collection in which the term is relatively frequent.***
Mathematically, we represent the average occurrences of term t in documents in collection c by:

$$F_{t,c} = \frac{f_{c,t}}{df_{c,t}},$$

where:

$f_{c,t}$ is the number of occurrences of term t in collection c; and

$df_{c,t}$ is the number of documents in collection c containing term t.

It follows that the frequency of a query term within documents in a collection, relative to the other collections (relative frequency) can be represented by:

$$RF_{t,c} = \frac{F_{t,c}}{\sum_{i=1}^{|C|} F_{t,i}},$$

where:

$|C|$ is the number of collections.

Given these mathematical representations of Zobel's criteria, in the following section we present the specification of our collection suggestion algorithm, which is composed from these components.

5.3 Algorithm Specification

Our algorithm for collection suggestion is formulated from mathematical representations of Zobel's criteria for highly ranked collections [66]. Therefore, for a query q, our algorithm (henceforth referred to as the *Doddle* algorithm) calculates a score for a collection c by:

$$Doddle(q, c) = \sum_{t \in q} f_{q,t} \cdot (RC_{t,c} + RP_{t,c} + RF_{t,c}),$$

where $f_{q,t}$ is the number of occurrences of a term t in the query. The more occurrences of a term in the query, the larger its contribution to the collection score.

We summarise the algorithm components (the origin of which was discussed in the previous section) as follows. The $RC_{t,c}$ and $RP_{t,c}$ components of the algorithm are collection level statistics: $RC_{t,c}$ (relative commonness) is derived from Zobel's second criterion, and represents how common a term is in a collection, relative to the other collections. The $RP_{t,c}$ (relative proportion) component is derived from Zobel's third criterion, and looks at the proportion of documents in a collection that contain the query term, relative to the other collections.

In contrast, $RF_{t,c}$ (relative frequency) is a document level statistic, derived from Zobel's fourth criterion. It looks at how frequently terms occur within the documents of a collection, relative to the other collections.

Specifically, the three main components of the algorithm are defined as follows:

Relative Commonness:	**Relative Proportion:**	**Relative Frequency:**
(derived from 2.)	(derived from 3.)	(derived from 4.)
$RC_{t,c} = \frac{C_{t,c}}{\sum_{i=1}^{|C|} C_{t,i}}$	$RP_{t,c} = \frac{P_{t,c}}{\sum_{i=1}^{|C|} P_{t,i}}$	$RF_{t,c} = \frac{F_{t,c}}{\sum_{i=1}^{|C|} F_{t,i}}$

where:

$C_{t,c} = \frac{f_{c,t}}{tokens_c}$ (commonness of term t in collection c);

$P_{t,c} = \frac{df_{c,t}}{docs_c}$ (proportion of documents in collection c containing term t);

$F_{t,c} = \frac{f_{c,t}}{df_{c,t}}$ (average occurrences of term t in documents in collection c);

$f_{c,t}$ is the number of occurrences of term t in collection c;

$tokens_c$ is the number of terms in collection c;

$df_{c,t}$ is the number of documents in collection c containing term t;

$docs_c$ is the number of documents in collection c; and

$|C|$ is the number of collections.

In the following section we conduct initial investigations into the performance of this algorithm.

5.4 Algorithm Performance

In this section we provide an initial evaluation of the performance of the Doddle algorithm; we will later compare its performance to that of existing algorithms from the collection selection and query performance prediction domains.

We begin by testing the Doddle algorithm against the test scenarios introduced in Chapter 3. Following this, we conduct a more rigorous investigation of the algorithm's performance. For this we utilise the baseline testing strategy and performance measures, also discussed in Chapter 3, and the two open access repository test data sets, described in Chapter 4.

5.4.1 Scenario-based Test Results

We first examine the suitability of our Doddle algorithm for the collection suggestion task, by testing it on our abstract scenarios. Recall from Chapter 3 that the scenarios offer basic test cases with controlled data, where we can reason clearly about the optimal ordering of collections. They act as a "health check" for algorithms; those producing incorrect rankings are unlikely to follow the collection suggestion notion of how collections should be ordered.

The results of the scenario tests for the Doddle algorithm are given in Table 5.1. For each scenario, we give either a tick (✓) or a cross (✗), showing that an algorithm has produced the correct ordering of collections for the given scenario, or conversely, that an incorrect ordering was produced. For comparison, the results achieved when ordering collections according to the Size-Based Ranking (SBR) lower bound baseline are also given in the table. The SBR performance gives a frame of reference, from which we can interpret how well Doddle performs.

Table 5.1: The performance of the Doddle algorithm over scenario-based tests. For comparison, the results of the SBR baseline are also given.

	Scenarios							
Algorithms	S_1	S_2	S_3	S_4	S_5	S_6	S_7	
Doddle	✓	✓	✓	✓	✓	✓	✓	(7 correct)
SBR	✓	✓	✗	✓	✓	✓	✗	*(5 correct)*

We observe that the Doddle algorithm produces the desired ordering of collections for all seven scenarios. However, the SBR baseline, which merely orders collections by the number of documents they contain, fails on two scenarios: S_3, where collection sizes vary, but the quantity of relevant documents is static; and S_7, where the best collection, while smaller than the second-placed collection, has a higher concentration (proportion) of relevant documents.

As such, these initial results suggest that the Doddle algorithm may be well suited to ranking collections for collection suggestion. However, the scenario-based testing technique is not comprehensive. Therefore, in the following sections we investigate the performance of the algorithm more closely, using test data sets built from open access repositories, and a variety of performance measures.

5.4.2 Baseline Testing Results – Initial Test Data Set

In this section we present and discuss the results achieved by the Doddle algorithm, on our open access repository Initial Test Data (ITD) set. Recall from Chapter 4 that this is a preliminary data set, used to investigate and test our approach to gathering real repository data, and to conduct initial tests on our collection suggestion algorithm. As such, this data set is small, comprising only 16 collections and 50 test queries. Within this data set, we utilise two separate term indexes: one comprising terms from Title metadata, the other comprising terms from both Title and Description metadata.

We address the algorithm's performance (over each term index) with respect to each class of performance measure in turn.

Rank Correlation

Recall from Chapter 3 that for each of the three rank correlation coefficients (Spearman, Blest and Da Costa) we use, correlation scores between the algorithm-produced collection ranking and a baseline ranking are averaged over all test queries.

We first consider the performance of Doddle according to the Spearman rank correlation coefficient (the results are shown in Table 5.2). Over both Title only, and the Title and Description metadata term indexes, there is statistically significant correlation between the collection rankings produced by Doddle, and the optimal FsBR baseline: $r_s(14) = 0.82, p < 0.05$ for Titles, and $r_s(14) = 0.72, p < 0.05$ for Titles and Descriptions.

Table 5.2: The average Spearman rank correlations (over 50 test queries and the ITD set) for the Doddle algorithm, comparing against both the FsBR and SBR baselines. Statistically significant correlations are given in bold.

	FsBR		**SBR**	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
Doddle	**0.82**	**0.72**	0.42	0.29
SBR	***0.51***	***0.48***	—	—

Table 5.3: The average Blest and Da Costa weighted rank correlations (over 50 test queries and the ITD set) for the Doddle algorithm, comparing estimate rankings to FsBR. Statistically significant correlations are given in bold.

	Blest		**Da Costa**	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
Doddle	**0.84**	**0.75**	**0.81**	**0.72**
SBR	***0.50***	***0.46***	***0.48***	***0.44***

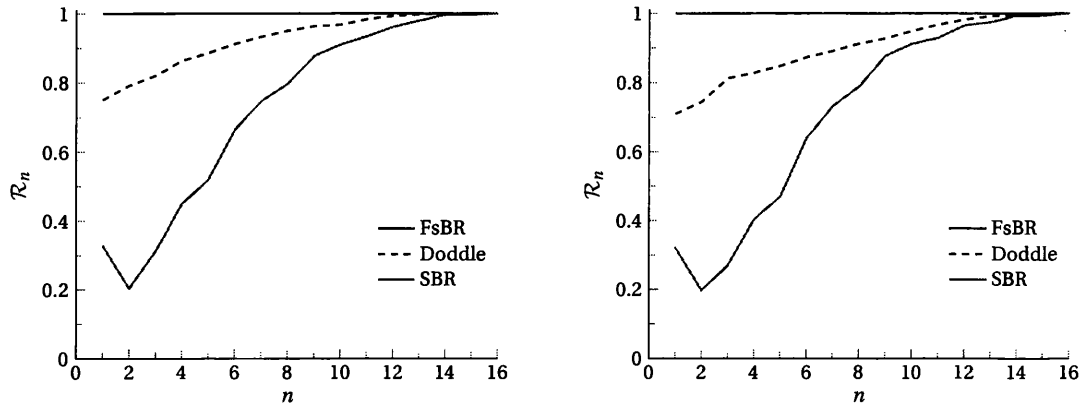
However, we find there is no significant correlation between the Doddle algorithm and a size-based ranking: $r_s(14) = 0.42, p > 0.05$ for Titles, and $r_s(14) = 0.29, p > 0.05$ for Titles and Descriptions. This result suggests that Doddle does not tend to favour large collections over (possibly more useful) smaller ones.

The Blest and Da Costa correlation coefficients (given in Table 5.3) are *weighted* rank correlation coefficients: they place an emphasis on similarity within the top portions of the rankings. The Doddle algorithm also exhibits strong performance according to these two measures, showing statistically significant correlation with FsBR. For Blest: $v(14) = 0.84, p < 0.05$ for Titles, and $v(14) = 0.75, p < 0.05$ for Titles and Descriptions; and for Da Costa: $r_w(14) = 0.81, p < 0.05$ for Titles, and $r_w(14) = 0.72, p < 0.05$ for Titles and Descriptions.

We observe that for the three coefficients, correlation scores between Doddle and the optimal FsBR baseline tend to be lower over Title and Description data, than on the Title data alone. However, the differences are not statistically significant.

Recall and Precision Analogues

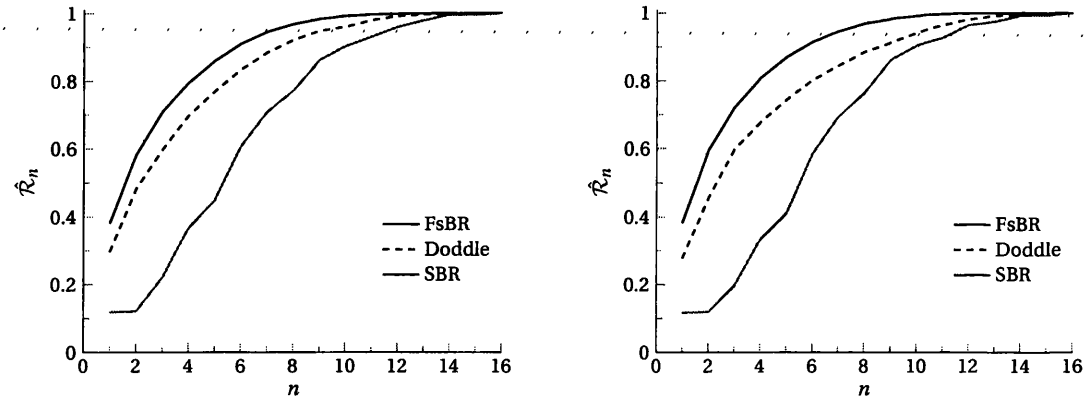
In this section we discuss the performance of the Doddle algorithm in terms of the performance measures analogous to precision and recall: $\mathcal{R}_n$, $\hat{\mathcal{R}}_n$ and $\mathcal{P}_n$. For each of these measures, the score achieved by Doddle at each value of n (where n is a rank position) is plotted on a graph; scores are averaged over the whole set of test queries. To put Doddle scores into context, we also plot the optimal performance (the FsBR baseline evaluated against itself), and the performance of the lower bound SBR baseline in relation to FsBR.



(a) Title data.

(b) Title and Description data.

Figure 5.1: The $\mathcal{R}_n$ values achieved by the Doddle algorithm, on the ITD set.



(a) Title data.

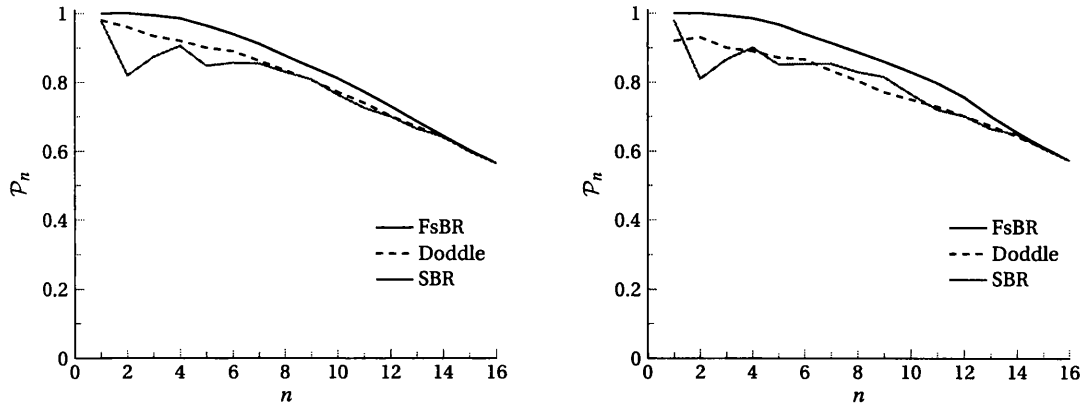
(b) Title and Description data.

Figure 5.2: The $\hat{\mathcal{R}}_n$ values achieved by the Doddle algorithm, on the ITD set.

The performance of Doddle, according to the $\mathcal{R}_n$ measure, is shown in Figure 5.1; specifically, Figure 5.1a shows performance when Title metadata is used to rank collections, and Figure 5.1b shows performance when Title and Description metadata is used. Recall from Chapter 3 that $\mathcal{R}_n$ indicates how well an algorithm selects the best collection available at each rank position (those collections with the most merit, according to the optimal ranking).

We observe that Doddle exhibits encouraging performance: while it does not always select the best collections at the early rank positions, its performance is notably stronger than that of the SBR baseline.

Doddle also shows strong performance according to the $\hat{\mathcal{R}}_n$ measure, which indicates how much of the total merit (from all collections) has been accumulated by the collection



(a) Title data.

(b) Title and Description data.

Figure 5.3: The $\mathcal{P}_n$ values achieved by the Doddle algorithm, on the ITD set.

selected at each stage. Figure 5.2 shows the scores achieved on this metric: on the Title data, Doddle is only 8 percentage points behind the optimal performance, and 10 percentage points behind on Title and Description data. By comparison, the SBR baseline visibly performs much worse.

Figure 5.3 plots the scores achieved by Doddle with respect to the $\mathcal{P}_n$ measure, which indicates the fraction of useful collections that are currently in the ranking. Note that as we get further down the optimal ranking, we start to include collections that have little or no relevance to the query, and as such the $\mathcal{P}_n$ score gradually declines.

On the $\mathcal{P}_n$ measure, Doddle performs well; however there is room for improvement. At the first rank position, a score of 1 is desirable, meaning the algorithm always selects a collection with *some* merit (though note this does not necessarily mean it selects the best available collection). Doddle does not always achieve this, suggesting it sometimes chooses a collection that FsBR has deemed unsuitable. On the Title data, the performance of Doddle is generally better than that of SBR. However, on the Title and Description data its performance drops, and is only roughly on-par with SBR.

We observe that, like the rank correlation coefficients, the performance of Doddle drops slightly on Title and Description data, over all three of these performance measures.

Top Rank Performance

A key aspect of our evaluation of algorithms with respect to the collection suggestion task, is to examine the performance within the top rank positions. As we are recommending collections to the user, it is important to get the collections within the top rank positions correct.

In Table 5.4 we provide the scores for our two metrics that examine performance within the top rank positions: Precision@5 and Correct@1 (described in Chapter 3). According to

Table 5.4: The Precision@5 and Correct@1 scores achieved by the Doddle algorithm, on the ITD set. Scores achieved by SBR are also given for comparison.

	Precision@5		**Precision@5**			
Algorithms	Titles	Titles & Desc.	Titles		Titles & Desc.	
Doddle	0.73	0.68	27	*(54%)*	25	*(50%)*
SBR	*0.43*	*0.42*	*7*	*(14%)*	*7*	*(14%)*

the Precision@5 measure, the Doddle algorithm does well at including the best collections (regardless of order) within the top five rank positions.

The Correct@1 measure shows the percentage of queries for which the algorithm correctly predicts the collection in the first rank position. On the Title data, Doddle correctly identifies the best collection on 54% of the queries, and 50% of the queries for Title and Description data. As such, there is room for improvement here; however, the algorithm does perform much better than the SBR lower bound baseline, which succeeds on predicting the top ranked collection on only 7% of the queries, over both sets of data.

5.4.3 Baseline Testing Results – Refined Test Data Set

In this section we summarise the results achieved by the Doddle algorithm, on our open access repository Refined Test Data (RTD) set. The results are generally similar to those from the Initial Test Data (ITD) set in the previous section, however there are some differences.

The RTD set is much larger than the ITD set used in the previous section, containing 100 collections. The approach to constructing this data set followed a methodical process to select repositories to include (see Chapter 4). In this evaluation, we use the set of 50 test queries associated with the RTD set, to test the performance of Doddle.

The Spearman, Blest and Da Costa correlation coefficients achieved by Doddle are given in Tables 5.5 and 5.6; Doddle again shows significant correlation to the optimal FsBR baseline. However, in this instance the difference between the correlation scores achieved on the Title metadata, and the Title and Description metadata, is statistically significant.

Table 5.5: The average Spearman rank correlations (over 50 test queries and the RTD set) for the Doddle algorithm, comparing against both the FsBR and SBR baselines. Statistically significant correlations are given in bold.

	FsBR		**SBR**	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
Doddle	**0.92**	**0.76**	**0.40**	**0.35**
SBR	***0.42***	***0.45***	—	—

Table 5.6: The average Blest and Da Costa weighted rank correlations (over 50 test queries and the RTD set) for the Doddle algorithm, comparing estimate rankings to FsBR. Statistically significant correlations are given in bold.

	Blest		**Da Costa**	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
Doddle	**0.85**	**0.63**	**0.86**	**0.66**
SBR	*0.05*	***0.17***	***0.48***	***0.49***

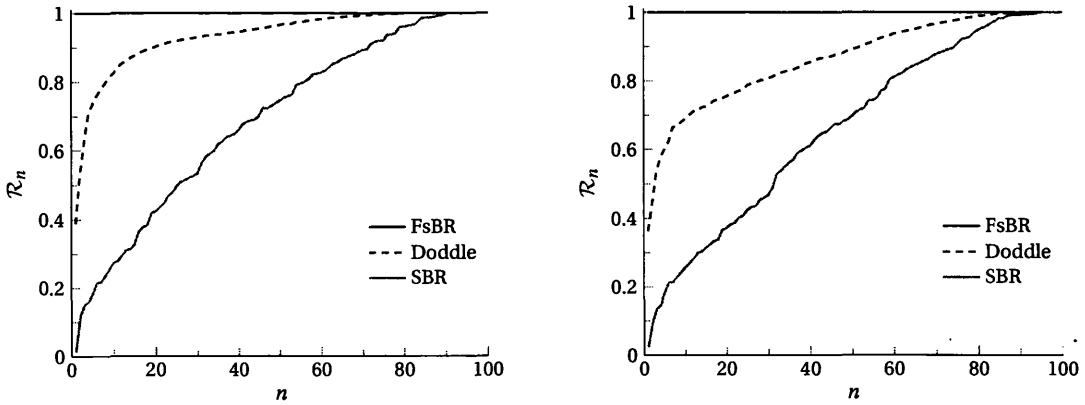
Also in contrast to the findings from the ITD set, Doddle shows a statistically significant correlation to the Size-Based Ranking on this data set.

In Figures 5.4, 5.5 and 5.6 we present the graphs for the recall and precision analogous measures. On $\mathcal{R}_n$, we observe that Doddle appears to get off to a poor start, suggesting it does not tend to select the best collections at the early rank positions. However, in comparison, the SBR lower bound baseline performs particularly poorly (suggesting the largest collections have little merit, according to the optimal baseline).

The performance of Doddle is more encouraging according to the $\hat{\mathcal{R}}_n$ measure. Looking at the graphs in Figure 5.5, we see that at rank position one, Doddle is only 4 percentage points behind the optimal on Title metadata, and 7 percentage points on Title and Description metadata. By comparison, SBR again performs poorly.

Figure 5.6 gives the graphs for the scores achieved on the $\mathcal{P}_n$ measure. Like on the ITD set, Doddle fails to choose a useful collection at rank position one for all queries. On the Title metadata, Doddle is consistently stronger than SBR. However, on Titles and Description metadata, despite a good start, it drops below the level of the SBR lower bound baseline.

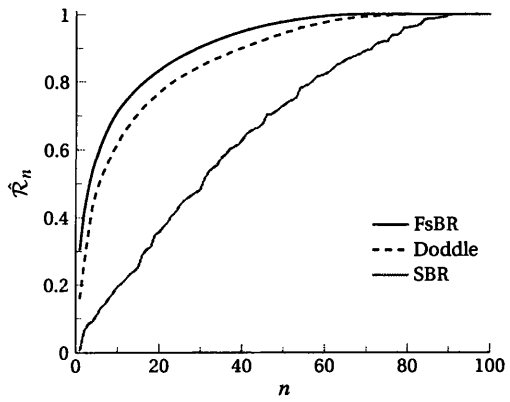
Finally, Table 5.7 gives Precision@5 and Correct@1 scores; these are noticeably lower



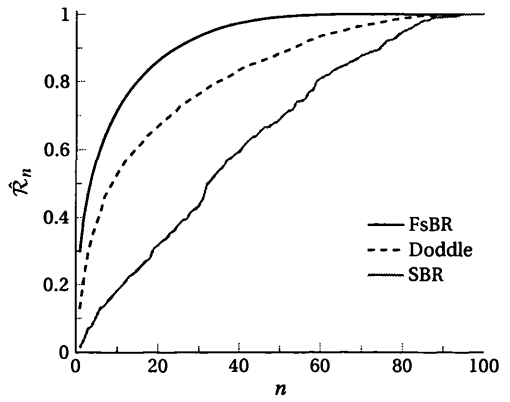
(a) Title data.

(b) Title and Description data.

Figure 5.4: The $\mathcal{R}_n$ values achieved by the Doddle algorithm, on the RTD set.

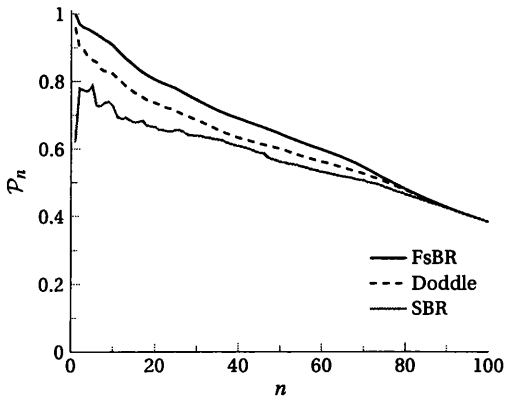


(a) Title data.

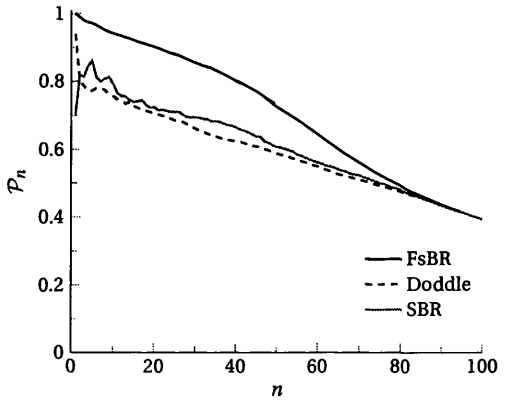


(b) Title and Description data.

Figure 5.5: The $\hat{\mathcal{R}}_n$ values achieved by the Doddle algorithm, on the RTD set.



(a) Title data.



(b) Title and Description data.

Figure 5.6: The $\mathcal{P}_n$ values achieved by the Doddle algorithm, on the RTD set.

Table 5.7: The Precision@5 and Correct@1 scores achieved by the Doddle algorithm, on the RTD set. Scores achieved by SBR are also given for comparison.

	Precision@5		**Correct@1**			
Algorithms	Titles	Titles & Desc.	Titles		Titles & Desc.	
Doddle	0.53	0.38	13	*(26%)*	11	*(22%)*
SBR	*0.07*	*0.09*	*0*	*(0%)*	*1*	*(2%)*

than those achieved on the ITD set.

These results suggests that emulating an optimal collection ranking may be more challenging on a larger data set: several collections may appear to be similar.

In the following section we continue to develop our algorithm, by testing its component parts.

5.5 Evaluation of Algorithm Components

In the previous section, we tested the initial specification of our algorithm for collection suggestion. Performance of the algorithm varied according to different measures, however the results were generally encouraging.

In this section, we investigate the contributions of the individual components of the Doddle algorithm (and therefore each of Zobel's criteria discussed in Section 5.2), by using each as an independent ranking algorithm. In addition, we combine these components in different ways, to examine the performance of different configurations of the algorithm. As such, we can use the strongest formulations of the Doddle algorithm to evaluate against existing algorithms, in Chapters 6 and 7.

To this end, we present the configurations of the Doddle algorithm we will test in Section 5.5.1. Following this, the results of the scenario-based testing are discussed in Section 5.5.2, before examining the baseline testing results in Section 5.5.3.

5.5.1 Specification of Alternative Algorithm Configurations

In this section we present the additional configurations of the Doddle algorithm we will test. These include using the individual components of the algorithm as independent ranking algorithms, and combining the components in alternative ways. We summarise the configurations as follows:

$Doddle\_RC(q,c) = \sum_{t \in q} f_{q,t} \cdot (RC_{t,c})$	Only the *relative commonness* component is used.
$Doddle\_RP(q,c) = \sum_{t \in q} f_{q,t} \cdot (RP_{t,c})$	Only the *relative proportion* component is used.
$Doddle\_RF(q,c) = \sum_{t \in q} f_{q,t} \cdot (RF_{t,c})$	Only the *relative frequency* component is used.
$Doddle\_RC{+}RP(q,c) = \sum_{t \in q} f_{q,t} \cdot (RC_{t,c} + RP_{t,c})$	The *relative commonness* and *relative proportion* components are used.
$Doddle\_RC{+}RF(q,c) = \sum_{t \in q} f_{q,t} \cdot (RC_{t,c} + RF_{t,c})$	The *relative commonness* and *relative frequency* components are used.

$Doddle\_RP{+}RF(q,c) = \sum_{t \in q} f_{q,t} \cdot (RP_{t,c} + RF_{t,c})$	The *relative proportion* and *relative frequency* components are used.
$Doddle\_\times(q,c) = \sum_{t \in q} f_{q,t} \cdot (RC_{t,c} \cdot RP_{t,c} \cdot RF_{t,c})$	All three of the main components are used, but multiplied together.
$Doddle\_RC{\times}RP(q,c) = \sum_{t \in q} f_{q,t} \cdot (RC_{t,c} \cdot RP_{t,c})$	The *relative commonness* and *relative proportion* components are used, multiplied together.
$Doddle\_RC{\times}RF(q,c) = \sum_{t \in q} f_{q,t} \cdot (RC_{t,c} \cdot RF_{t,c})$	The *relative commonness* and *relative frequency* components are used, multiplied together.
$Doddle\_RP{\times}RF(q,c) = \sum_{t \in q} f_{q,t} \cdot (RP_{t,c} \cdot RF_{t,c})$	The *relative proportion* and *relative frequency* components are used, multiplied together.
$Doddle\_W(q,c) = \sum_{t \in q} f_{q,t} \cdot \frac{(2 \cdot RC_{t,c} + 2 \cdot RP_{t,c} + RF_{t,c})}{5}$	We calculate the weighted arithmetic mean of the three main components. The $RC_{t,c}$ and $RP_{t,c}$ components are given a higher weighting than $RF_{t,c}$.

We test and discuss the performance of these algorithm configurations, along with the original configuration of Doddle, in the following sections.

5.5.2 Scenario-based Test Results

The results achieved by the various configurations of the Doddle algorithm on the scenario-based tests are presented in Table 5.9. For comparison, the results of the original configuration of Doddle, and the SBR lower bound baseline are also given.

From the table we see that all configurations of the algorithm perform well on these tests. One exception is Doddle_RF, which uses only the 'relative frequency' component. This configuration fails to correctly rank the collections on S_7, by ordering collection C_B above C_A. The reason for this is straightforward: Doddle_RF is a document level metric, using the frequency of query terms within documents to produce a collection ranking; in S_7, documents in C_B are more likely to contain the query terms than those in C_A.

At this stage, the performances of the configurations of the Doddle algorithm appear fairly similar. In the following section, we conduct a more rigorous evaluation of the algorithm configurations, enabling us to more closely distinguish between their performances, and identify any strengths and weaknesses.

Table 5.9: The performance of the various configurations of the Doddle algorithm, over scenario-based tests. For comparison, the results of the SBR baseline are also given.

	Scenarios							
Algorithms	S_1	S_2	S_3	S_4	S_5	S_6	S_7	
Doddle	✓	✓	✓	✓	✓	✓	✓	(7 correct)
Doddle_RC	✓	✓	✓	✓	✓	✓	✓	(7 correct)
Doddle_RP	✓	✓	✓	✓	✓	✓	✓	(7 correct)
Doddle_RF	✓	✓	✓	✓	✓	✓	✗	(6 correct)
Doddle_RC+RP	✓	✓	✓	✓	✓	✓	✓	(7 correct)
Doddle_RC+RF	✓	✓	✓	✓	✓	✓	✓	(7 correct)
Doddle_RP+RF	✓	✓	✓	✓	✓	✓	✓	(7 correct)
Doddle_×	✓	✓	✓	✓	✓	✓	✓	(7 correct)
Doddle_RC×RP	✓	✓	✓	✓	✓	✓	✓	(7 correct)
Doddle_RC×RF	✓	✓	✓	✓	✓	✓	✓	(7 correct)
Doddle_RP×RF	✓	✓	✓	✓	✓	✓	✓	(7 correct)
Doddle_W	✓	✓	✓	✓	✓	✓	✓	(7 correct)
SBR	✓	✓	✗	✓	✓	✓	✗	*(5 correct)*

5.5.3 Baseline Test Results

In this section we present and discuss the results of the baseline testing, for the various configurations of the Doddle algorithm. We primarily present the results from the Refined Test Data (RTD) set, and simply provide a summary of findings from the Initial Test Data (ITD) set as follows: over all measures, the performances of the algorithm configurations is generally close. The exception to this is Doddle_RF, which is notably poorer than the other algorithms. Some algorithm configurations (Doddle_RC, Doddle_RP, Doddle_RC+RF, and Doddle_RP+RF) score marginally higher than others, but this varies from measure to measure; the differences between the algorithms are not statistically significant. Complete performance scores from the ITD set can be found in Appendix D.

We discuss the results from the RTD set in depth in the following sections.

Rank Correlation

Here we present and discuss the scores achieved by the configurations of the Doddle algorithm, according to the rank correlation measures.

Table 5.10 gives the Spearman rank correlation scores achieved by measuring the similarity between the algorithm-produced collection rankings, and both the FsBR optimal baseline and the SBR lower bound baseline; statistically significant scores ($p < 0.05$) are marked in bold.

From the table, we see that all configurations of the Doddle algorithm show significant correlation to the optimal FsBR baseline, on both the Title metadata and the Title and De-

Table 5.10: The average Spearman rank correlations (over 50 test queries) for the various configurations of Doddle, comparing against both the FsBR and SBR baselines. Statistically significant correlations are given in bold.

	FsBR		SBR	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
Doddle	**0.86**	**0.65**	**0.38**	**0.30**
Doddle_RC	**0.87**	**0.72**	**0.37**	**0.34**
Doddle_RP	**0.86**	**0.60**	**0.36**	**0.23**
Doddle_RF	**0.80**	**0.64**	**0.53**	**0.44**
Doddle_RC+RP	**0.86**	**0.66**	**0.37**	**0.28**
Doddle_RC+RF	**0.86**	**0.70**	**0.39**	**0.37**
Doddle_RP+RF	**0.85**	**0.60**	**0.39**	**0.29**
Doddle_×	**0.86**	**0.65**	**0.36**	**0.28**
Doddle_RC×RP	**0.86**	**0.65**	**0.36**	**0.27**
Doddle_RC×RF	**0.86**	**0.71**	**0.37**	**0.35**
Doddle_RP×RF	**0.85**	**0.61**	**0.37**	**0.26**
Doddle_W	**0.86**	**0.66**	**0.37**	**0.29**
SBR	***0.42***	***0.46***	—	—

scription metadata.

On the Title data, the Doddle_RC configuration achieves the highest correlation score, while Doddle_RF scores lowest. However, there is little variation between the scores of the different configurations, and there is no statistically significant difference between any of the configurations of the Doddle algorithm. However, all configurations exhibit performance that is significantly different from that achieved by the SBR baseline.

The correlation scores achieved on the Title and Description data are varied, with Doddle_RC again scoring highest. However, there is no statistically significant difference between the performance of any of the configurations. The majority of the algorithms show performance that is significantly different from that of the SBR baseline, with the exception of Doddle_RP, Doddle_RF, Doddle_RP+RF and Doddle_RP×RF.

On the Title and Description data, the correlation scores achieved are lower than those on the Title data alone. Indeed, on this occasion the differences between correlation scores on the two data sets are statistically significant.

Turning to the correlation scores between the algorithm-produced rankings and the SBR baseline, we see that all algorithm configurations show statistically significant correlation with SBR. This suggests that on this test data set, there is some tendency to select large collections over smaller ones. However, since SBR itself shows significant correlation with FsBR, it is expected that there should be some agreement between SBR, and the rankings produced by the algorithms.

In Table 5.11 we present the Blest and Da Costa weighted rank correlation scores the algorithms achieved, measuring the similarity between the algorithm-produced collection

Table 5.11: The average Blest and Da Costa weighted rank correlations (over 50 test queries) for the various configurations of Doddle, comparing algorithm-produced rankings to FsBR. Statistically significant correlations are given in bold.

	Blest		Da Costa	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
Doddle	**0.85**	**0.63**	**0.86**	**0.66**
Doddle_RC	**0.86**	**0.70**	**0.87**	**0.73**
Doddle_RP	**0.85**	**0.59**	**0.85**	**0.61**
Doddle_RF	**0.75**	**0.58**	**0.79**	**0.63**
Doddle_RC+RP	**0.85**	**0.65**	**0.86**	**0.67**
Doddle_RC+RF	**0.85**	**0.67**	**0.86**	**0.71**
Doddle_RP+RF	**0.84**	**0.58**	**0.85**	**0.61**
Doddle_×	**0.85**	**0.63**	**0.85**	**0.66**
Doddle_RC×RP	**0.85**	**0.64**	**0.86**	**0.66**
Doddle_RC×RF	**0.85**	**0.68**	**0.86**	**0.71**
Doddle_RP×RF	**0.84**	**0.58**	**0.85**	**0.61**
Doddle_W	**0.85**	**0.64**	**0.86**	**0.66**
SBR	*0.05*	***0.17***	***0.48***	***0.49***

rankings, and the FsBR baseline. The observations here are similar to those from the Spearman scores: performance of the algorithms is better over the Title data alone, with Doddle_RC performing strongest over both term indexes. The Doddle_RF configuration performs worst on the Title data, but is generally competitive on Title and Description data.

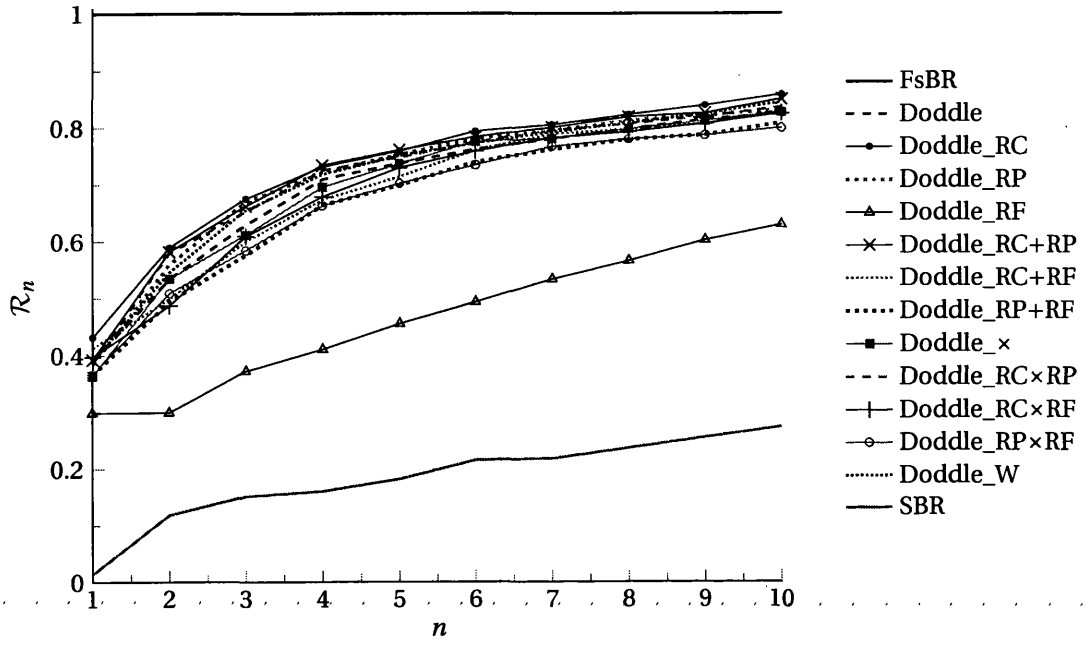
Recall and Precision Analogues

In this section, we examine the performances of the configurations of the Doddle algorithm according to the $\mathcal{R}_n$, $\hat{\mathcal{R}}_n$ and $\mathcal{P}_n$ measures. The algorithm performances are presented as graphs, given in Figures 5.7, 5.8 and 5.9. In the interests of readability, the graphs only present the scores achieved within the first 10 rank positions; scores achieved at additional rank positions are provided in Tables D.33, D.35, D.41, D.43, D.49 and D.51 in Appendix D.

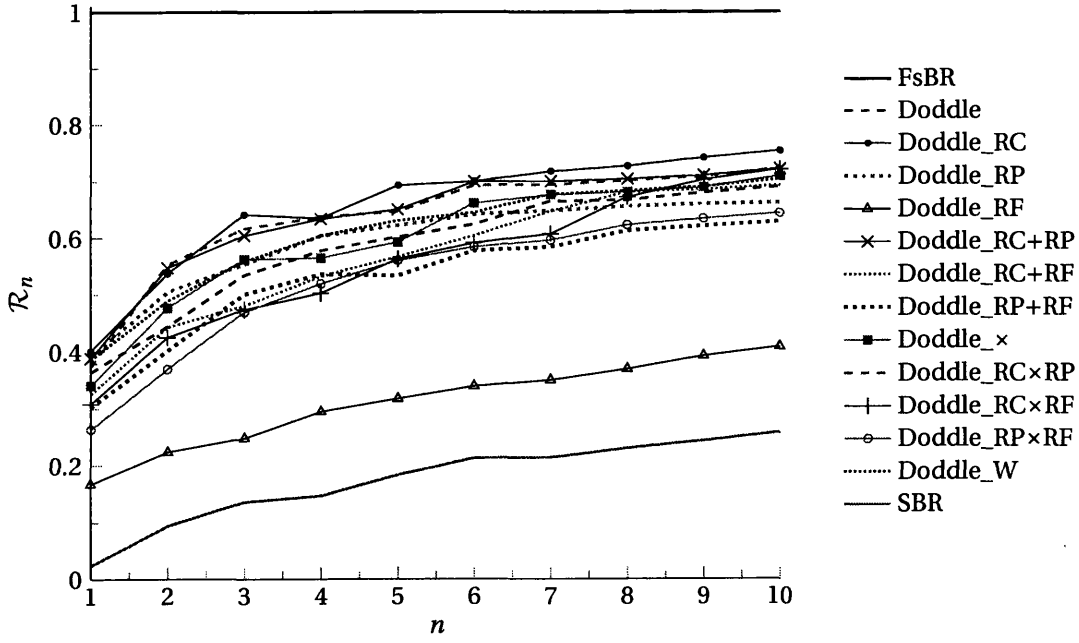
Figure 5.7 shows the results from the $\mathcal{R}_n$ measure. We first consider Figure 5.7a, which represents the performance achieved on the Title metadata. Here, all configurations of the algorithm perform similarly, with the exception of Doddle_RF, which is clearly below the other configurations. Overall, it appears that Doddle_RC and Doddle_RC+RP tend to represent the strongest performance.

The findings are much the same in Figure 5.7b, which shows the performance achieved on the Title and Description metadata. However, we observe that there is more variation between the algorithms, and overall the $\mathcal{R}_n$ scores are lower than those from the Title data.

We observe similar trends in Figures 5.8a and 5.8b, which present the performance according the the $\hat{\mathcal{R}}_n$ measure. Again, Doddle_RC and Doddle_RC+RP show the strongest

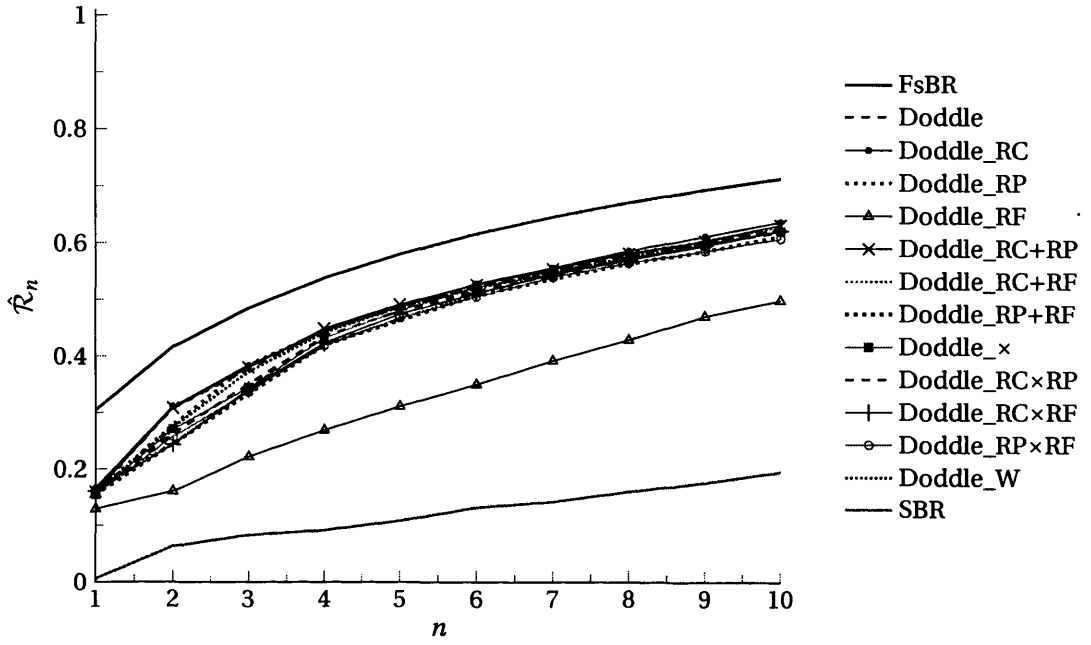


(a) Title data.

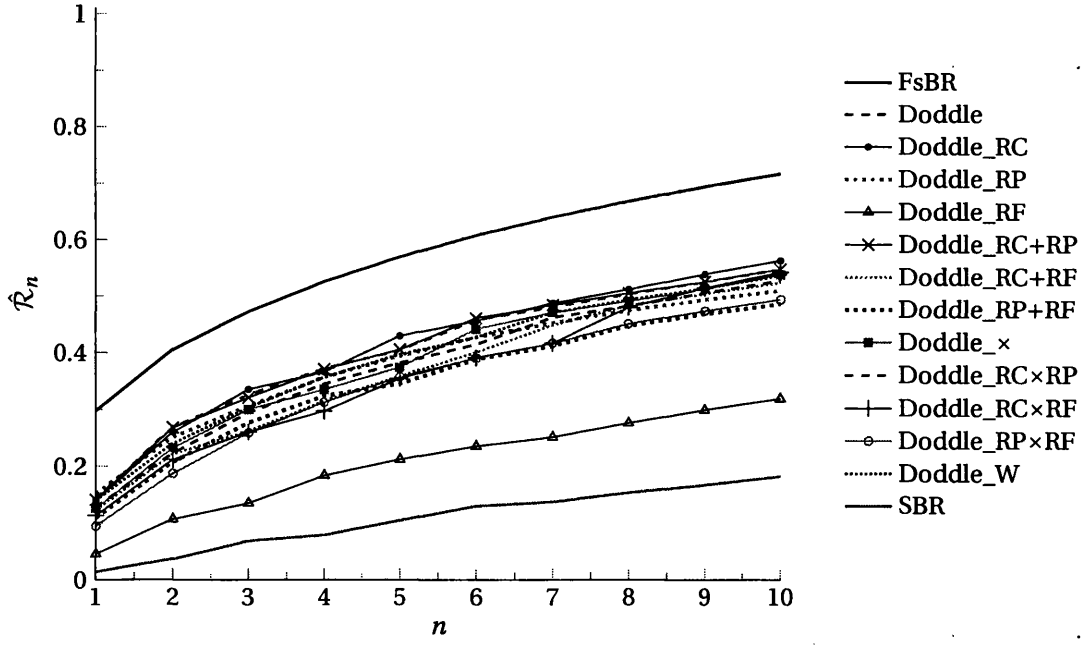


(b) Title and Description data.

Figure 5.7: The $\mathcal{R}_n$ values (up to $n = 10$) achieved by the various configurations of the Doddle algorithm, on the RTD set.

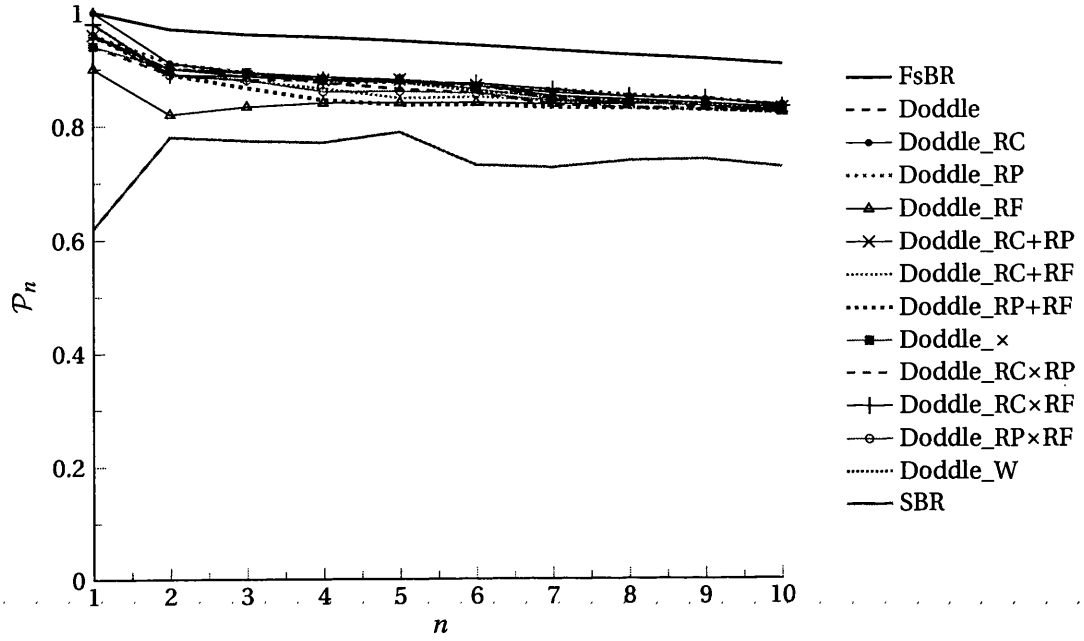


(a) Title data.

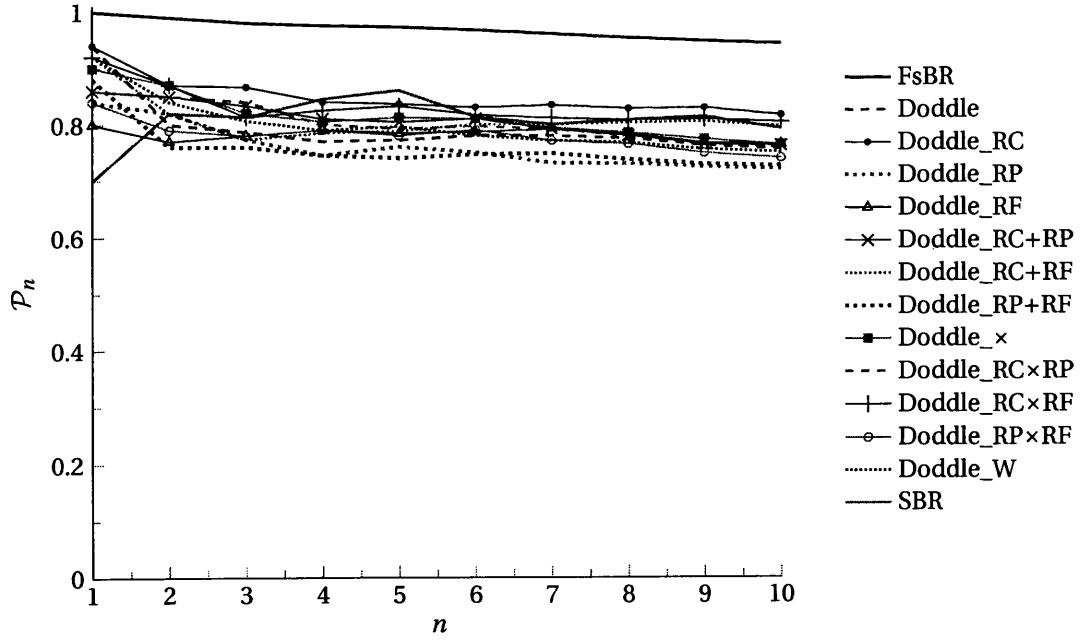


(b) Title and Description data.

Figure 5.8: The $\hat{\mathcal{R}}_n$ values (up to $n = 10$) achieved by the various configurations of the Doddle algorithm, on the RTD set.



(a) Title data.



(b) Title and Description data.

Figure 5.9: The $\mathcal{P}_n$ values (up to $n = 10$) achieved by the various configurations of the Doddle algorithm, on the RTD set.

performance, with Doddle_RF scoring poorly. Scores over Title and Description data are lower, and more varied.

In Figure 5.9 we present the scores achieved on $\mathcal{P}_n$, which measures how well an algorithm selects collections that have some value to the query. As we see in Figure 5.9a, on the Title metadata, Doddle_RC is the only algorithm to achieve the optimal score of 1, at the first rank position: meaning that the first collection always has some relevance to the query. Aside from this, the performance of the various configurations is generally close, with Doddle_RF making the weakest start.

However, in Figure 5.9b we see the performances of the algorithms on the Title and Description metadata. Here, the algorithm performances are more varied, and are lower than those achieved on the Title data alone. Indeed, the scores are only roughly on-par with those of the SBR lower bound baseline.

Top Rank Performance

In Table 5.12 we present the average precision within the top five collections (Precision@5), and the number of queries (out of 50) for which the first ranked collection is correct (Correct@1). The scores here are lower than those on the ITD set, however this is not unexpected: the RTD set is much larger, and as there are more collections to choose between, ranking is more difficult.

Looking at the Precision@5 scores on the Title data, we see that the Doddle_RP and Doddle_RC+RP configurations score highest, however the majority of the algorithms score very closely. The exception of this is Doddle_RF, which scores much lower than the other

Table 5.12: The Precision@5 and Correct@1 scores achieved by the various configurations of the Doddle algorithm, on the RTD set.

	Precision@5		**Correct@1**			
Algorithms	Titles	Titles & Desc.	Titles		Titles & Desc.	
Doddle	0.53	0.38	13	*(26%)*	11	*(22%)*
Doddle_RC	0.56	0.43	14	*(28%)*	11	*(22%)*
Doddle_RP	0.57	0.46	13	*(26%)*	13	*(26%)*
Doddle_RF	0.26	0.21	7	*(14%)*	2	*(4%)*
Doddle_RC+RP	0.57	0.44	13	*(26%)*	13	*(26%)*
Doddle_RC+RF	0.51	0.34	14	*(28%)*	8	*(16%)*
Doddle_RP+RF	0.50	0.36	12	*(24%)*	9	*(18%)*
Doddle_×	0.55	0.38	12	*(24%)*	9	*(18%)*
Doddle_RC×RP	0.56	0.44	13	*(26%)*	11	*(22%)*
Doddle_RC×RF	0.53	0.34	13	*(26%)*	8	*(16%)*
Doddle_RP×RF	0.52	0.38	12	*(24%)*	7	*(14%)*
Doddle_W	0.56	0.41	13	*(26%)*	13	*(26%)*
SBR	*0.07*	*0.09*	*0*	*(0%)*	*1*	*(2%)*

algorithms. On the Title and Description data, the scores achieved by the algorithms are lower and more varied; Doddle_RP again scores best, while Doddle_RF is worst.

A similar picture is seen according to the Correct@1 measure: on the Title metadata, the performance of the algorithms are fairly similar, with Doddle_RF performing notably worse. On Title and Description data, there is more variation between the algorithms, but Doddle_RF still performs poorly. We note that the highest number of queries for which any algorithm correctly predicts the top ranked collection, is 14 (28%) on the Title data, and 13 (26%) on the Title and Description data: around half the number achieved on the ITD set. While a larger data set is likely to be more difficult, we would like to see improvement on these scores.

5.6 Summary

The development of a search service for finding domain-specific collections warrants an optimal algorithm for ranking collections according to a user's query; our work is motivated towards identifying such an algorithm. As we have discussed in Chapter 3 of this thesis, it is desirable that in this domain, highly ranked collections should contain a large number of relevant documents, with these relevant documents amounting to a high proportion of the whole collection.

There are a variety of existing algorithms that have been developed for the task of ranking collections (collection selection), or that may be applied to the task (query performance predictors); we discussed these in Chapter 2. We will later examine the suitability of these algorithms, for the collection suggestion task. However, in the interest of finding the most effective algorithm to use in a collection suggestion search service, we have developed our own algorithm.

In this chapter we have presented and tested the initial specification of the Doddle algorithm, designed specifically for the collection suggestion task. The algorithm is based upon Zobel's criteria for highly ranked collections, which strongly represent the collection suggestion objectives. In addition, to ensure we have a competitive implementation, we have investigated the performances of the individual components of the algorithm, and combined these in alternative ways, to create new algorithm variants.

Our evaluation has used a variety of performance measures, and our two test data sets comprising real repository data. Across the different test environments, the performance of Doddle and its components and configurations was encouraging. Indeed, no statistically significant differences in performance (between any configuration) were found.

Despite this, some patterns and observations have emerged. Most notably, the Doddle_RF configuration, which uses only the document-level 'relative frequency' metric, often performed worse than the other configurations. This suggests that Zobel's fourth criterion, regarding average frequency of query terms within documents, is not particularly useful for ranking collections.

Some algorithm configurations were observed to perform marginally better than others (but not statistically significantly so), depending on the performance measure. For example, the Doddle_RC, Doddle_RP, Doddle_RC+RP, Doddle_RC+RF and Doddle_RP+RF

configurations all scored 'best', and some stage of the evaluation. As such, we test these configurations, in addition to the original algorithm specification, alongside existing algorithms, in Chapters 6 and 7. While we omit the other configurations (Doddle_RF, Doddle_×, Doddle_RC×RP, Doddle_RC×RF, Doddle_RP×RF and Doddle_W) from further discussion, complete performance scores for these can be found in Appendix D.

Finally, we observed in this portion of our evaluation, that the algorithms experienced a drop in performance when tested on Title and Description metadata, compared to that exhibited on Title metadata alone. To ensure this is not a property of our algorithm, we investigate this further in the following chapters, with existing algorithms.

Chapter 6

Evaluation of Algorithms: Scenarios and TREC

In the previous chapter, we presented our own algorithm for ranking collections for the collection suggestion task. We measured the performance of the algorithm over two test data sets, comprising data from real open access repositories, and using a variety of performance measures. In addition, we tested the performance of the individual components of the algorithm, and alternative configurations of those components. Some configurations of the algorithm showed strong and promising results, most notably: Doddle_RC, Doddle_RP, Doddle_RC+RP, Doddle_RC+RF and Doddle_RP+RF (see Chapter 5 for details of the formulation of these algorithms), in addition to the original Doddle algorithm.

However, we are interested in identifying an optimal algorithm to rank collections (with respect to a user's query) in a collection suggestion search service. As such, in this chapter we begin to conduct an extensive evaluation of the suitability and performance of a selection of existing algorithms, with respect to the collection suggestion task. As discussed at the end of Chapter 2, we test algorithms that were found to be effective in their original domains, and those whose underlying theories relate to the objectives of collection suggestion: bGlOSS, CORI, Zobel's four lexicon inspection algorithms, CVV, DFPROP, Distribution of Informative Amount (1), SCS, AvICTF and NSCQ. We compare these algorithms to the strongest configurations of our own algorithm, in order to find the best algorithm for the job.

Our evaluation follows the methodology discussed in Chapter 3: we first examine algorithm performance using a set of scenario-based tests (see Section 3.4 in Chapter 3). This enables us to quickly identify algorithms that may be suitable for collection suggestion; we present our findings of this stage of our evaluation in Section 6.1.

For the second stage of our evaluation, we use the baseline testing technique (see Section 3.2 of Chapter 3) to closely examine the performance of those algorithms found by the scenario-based tests to be potentially suitable for collection suggestion. In this chapter we present and discuss (in Sections 6.2 to 6.7, with a summary in Section 6.8) baseline testing results conducted on the six TREC-based test data sets specified in Chapter 4. Further

experiments, using the open access repository data sets, follow in Chapter 7.

6.1 Scenario-Based Test Results

In this section, we begin to examine the suitability of a selection of existing algorithms (from the collection selection and query performance prediction domains) for the collection suggestion task, using the scenario-based testing strategy from Chapter 3. Recall that the scenarios act as a 'health check': algorithms that fail to produce correct collection rankings for the scenarios are unlikely to be suited to collection suggestion, as they do not rank collections in accordance with its ideals.

We present the results of the scenario tests in Table 6.1, where a tick (✓) represents that the algorithm has produced the correct collection ranking for a particular scenario, and a cross (✗) represents that the algorithm has produced an incorrect collection ranking. We discuss these results, and reflect on situations on which the algorithms fail.

From Table 6.1, we see that bGlOSS and CORI are the two most consistent algorithms, as they produce correct collection rankings for all scenarios. As such, these algorithms may be suited to collection suggestion, and should be tested further.

The group of lexicon inspection algorithms perform well overall: Inner Product, Skew and Highest-available Similarity produce correct collection rankings on six of the scenarios, failing only on the more difficult S_7, which represents polysemy. However, the Cosine Measure is an exception, only succeeding in producing correct rankings on S_5 and S_6.

Table 6.1: The performance of existing algorithms, over scenario-based tests. For comparison, the results of the SBR baseline are also given.

	Scenarios							
Algorithms	S_1	S_2	S_3	S_4	S_5	S_6	S_7	
bGlOSS	✓	✓	✓	✓	✓	✓	✓	(7 correct)
CORI	✓	✓	✓	✓	✓	✓	✓	(7 correct)
Cosine Measure	✗	✗	✗	✗	✓	✓	✗	(2 correct)
Inner Product	✓	✓	✓	✓	✓	✓	✗	(6 correct)
Skew	✓	✓	✓	✓	✓	✓	✗	(6 correct)
Highest-available Similarity	✓	✓	✓	✓	✓	✓	✗	(6 correct)
CVV	✓	✓	✓	✓	✓	✗	✗	(5 correct)
DFPROP	✓	✓	✓	✓	✓	✓	✗	(6 correct)
Distribution of Informative Amount	✗	✗	✗	✗	✗	✗	✗	(0 correct)
SCS	✗	✗	✗	✗	✓	✓	✗	(2 correct)
AvICTF	✗	✗	✗	✗	✓	✓	✗	(2 correct)
NSCQ	✓	✓	✗	✓	✓	✓	✗	(5 correct)
SBR	✓	✓	✗	✓	✓	✓	✗	*(5 correct)*

The Cosine Measure operates by calculating the similarity between the query and each collection. For those scenarios on which it fails, the algorithm ranks the C_C collection first, which contains only one occurrence of each query term; the Cosine Measure sees this as being the most similar to the query. For the scenarios for which the Cosine Measure produces a correct ranking, the best collection (C_A) is the only one that contains all query terms, making it most similar to the query.

Similar to the Cosine Measure, the NSCQ query performance prediction algorithm also calculates the similarity between the query and collections, with the addition of normalisation by query length. Although this algorithm fairs better than the Cosine Measure, it still fails on two scenarios: S_3 and S_7, the same scenarios on which the SBR lower bound baseline fails. Due to the poor performances of the Cosine Measure and NSCQ at this stage, we will omit them from further evaluation.

The CVV and DFPROP algorithms both utilise only document frequency statistics to produce collection rankings. However, their performances in the scenario tests differ slightly. DFPROP performs well, only failing on S_7. The CVV algorithm fairs slightly worse, failing to produce a correct collection ranking on S_6 and S_7.

The weakest group of algorithms in the scenario tests are He and Ounis's query performance predictors [28]: Distribution of Informative Amount, AvICTF and SCS. While AvICTF and SCS produce correct collection rankings on two scenarios (S_5 and S_6), Distribution of Informative Amount fails on all scenarios.

We note particularly that Distribution of Informative Amount fails on S_4, which uses a single-term query. This failure is due to the specification of the algorithm: Distribution of Informative Amount calculates the standard deviation of the inverse document frequency of the query terms. As such, when there is only one query term, each collection will receive a score of zero. Since this algorithm cannot process single-term queries (in addition to its failure on the other scenarios), it is clearly unsuitable for collection suggestion.

The overall failure of this group of query performance prediction algorithms can be attributed to their intended application. Query performance predictors are used to measure the quality of search results at a single search engine, and rely on the discriminatory power of query terms. Query terms that are common in a collection have little discriminatory power, as they do not help the search engine choose between documents. Conversely, query terms that are rare in a collection make particular documents stand out as being useful.

As such, in these algorithms, query terms occurring in a small number of documents within a collection are given a larger weight within the overall collection score. This results in the algorithms producing collection rankings that are contrary to the objectives of collection suggestion, where the intention is to identify collections about the query, and therefore where the query terms are common.

In the remainder of this chapter, we provide closer evaluations of the algorithms (using the six TREC-based test data sets, discussed in Chapter 4) found to perform well in the scenario-based tests: bGlOSS, CORI, Inner Product, Skew, Highest-available Similarity and DFPROP. We also consider the performance of our own collection suggestion algorithm, Doddle, and its most consistent alternative configurations (see Chapter 5).

We will omit from further evaluation and discussion those algorithms that have not exceeded the performance of the SBR lower bound baseline, in the scenario-based tests. How-

ever, full performance scores for omitted algorithms are provided in Appendix D for reference.

6.2 SYM-236

The SYM-236 test data set, as described in Chapter 4, comprises collections formulated from documents in discs 1-3 of the TREC corpus. The documents are grouped by original source and publication date, to give 236 collections of varying size.

Using this TREC-based test data set, we examine the effectiveness of a selection of existing (collection selection) algorithms, and the configurations of our own collection suggestion algorithm that we found to be most consistent in Chapter 5. We present the findings from each type of performance measure (rank correlation, recall and precision analogues, and top rank performance) individually, in the remainder of this section.

Rank Correlation

We first consider the effectiveness of the algorithms under test, according to Spearman rank correlation. Correlation scores between an algorithm-produced collection ranking and both the FsBR and SBR baselines are given in Table 6.2.

Looking first at the correlations with the FsBR baseline, we see from the table that all algorithms show a statistically significant ($p < 0.05$) correlation to FsBR, over both short and long queries.

Over the short queries, the correlation scores achieved by the collection selection algorithms are very close: while CORI scores highest, there is no significant difference between those algorithms. The configurations of the Doddle algorithm however, perform significantly worse than the collection selection algorithms. The exception to this is Doddle_RC, which is only significantly worse than the best scoring algorithm (CORI).

Over the long queries, we notice the correlation scores achieved by the algorithms tends to be higher than on short queries, though generally not significantly so. The exception here is bGlOSS, which does perform significantly worse on long queries than it does on short. This is likely due to how the algorithm works: using only document frequencies, it aims to estimate the number of documents likely to contain all query terms. The longer the query, the less likely it is that a document contains all query terms.

Overall, the trends observed on the long queries are similar to those on the short queries: the collection selection algorithms perform similarly (aside from bGlOSS), and are significantly better than all but the Doddle_RC configuration of the Doddle algorithm. Doddle_RC performs significantly better than bGlOSS, Doddle_RP and Doddle_RP+RF. Complete results showing whether differences between correlation scores are significant are given in Tables D.63 and D.64 in Appendix D.

All of the algorithms show significant correlation to the SBR baseline; the correlation scores of the collection selection algorithms (bGlOSS excluded) are notably high. The best performing configuration of Doddle, Doddle_RC, has the highest correlation with SBR, out of the configurations tested here. These results support our suggestion in Chapter 4 that the

Table 6.2: The average Spearman rank correlations (over 100 test queries and the SYM-236 test data set) for selected existing algorithms and configurations of Doddle, comparing against both the FsBR and SBR baselines. Statistically significant correlations are given in bold.

	FsBR		**SBR**	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.58**	**0.35**	**0.67**	**0.33**
CORI	**0.65**	**0.65**	**0.85**	**0.85**
Inner Product	**0.63**	**0.65**	**0.89**	**0.89**
Skew	**0.61**	**0.62**	**0.89**	**0.89**
Highest-available Similarity	**0.64**	**0.66**	**0.91**	**0.92**
DFPROP	**0.61**	**0.62**	**0.89**	**0.91**
Doddle	**0.39**	**0.46**	**0.33**	**0.41**
Doddle_RC	**0.53**	**0.57**	**0.56**	**0.59**
Doddle_RP	**0.32**	**0.40**	**0.20**	**0.24**
Doddle_RC+RP	**0.44**	**0.50**	**0.36**	**0.42**
Doddle_RC+RF	**0.43**	**0.49**	**0.44**	**0.50**
Doddle_RP+RF	**0.28**	**0.36**	**0.20**	**0.30**
SBR	***0.53***	***0.53***	—	—

Table 6.3: The average Blest and Da Costa weighted rank correlations (over 100 test queries and the SYM-236 test data set) for selected existing algorithms and configurations of Doddle, comparing estimate rankings to FsBR. Statistically significant correlations are given in bold.

	Blest		**Da Costa**	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.63**	**0.53**	**0.66**	**0.63**
CORI	**0.65**	**0.63**	**0.70**	**0.70**
Inner Product	**0.63**	**0.63**	**0.68**	**0.70**
Skew	**0.61**	**0.60**	**0.66**	**0.67**
Highest-available Similarity	**0.65**	**0.65**	**0.69**	**0.71**
DFPROP	**0.61**	**0.60**	**0.66**	**0.67**
Doddle	**0.37**	**0.43**	**0.44**	**0.52**
Doddle_RC	**0.52**	**0.55**	**0.58**	**0.63**
Doddle_RP	**0.31**	**0.37**	**0.39**	**0.46**
Doddle_RC+RP	**0.42**	**0.48**	**0.49**	**0.56**
Doddle_RC+RF	**0.41**	**0.45**	**0.48**	**0.54**
Doddle_RP+RF	**0.25**	**0.32**	**0.34**	**0.42**
SBR	***0.53***	***0.53***	***0.58***	***0.58***

data set favours a size-based ranking: the larger collections tend to match the most queries, and contain many of the relevant documents.

The findings from the Spearman rank correlation scores are generally supported by the correlation scores according to the Blest and Da Costa weighted rank correlation measures, given in Table 6.3. All algorithms show significant correlation with the FsBR optimal ranking, with CORI and Highest-available Similarity scoring highest. The best configuration of Doddle is Doddle_RC, but it does not quite match the performance of the existing collection ranking algorithms.

Recall and Precision Analogues

The three performance measures analogous to recall and precision ($\mathcal{R}_n$, $\hat{\mathcal{R}}_n$ and $\mathcal{P}_n$) look at how well the algorithms select the best collections, and how useful the selected collections are, at each rank position.

The results achieved by the algorithms according to the $\mathcal{R}_n$ measure are given in Figure 6.1. We plot only scores up to rank position ten; scores achieved at selected other rank positions are given in tables in Appendix D.

Figure 6.1a shows the scores from executing the short test queries. Here, CORI performs best overall, followed by bGlOSS. The Skew, Inner Product, Highest-available Similarity and DFPROP algorithms are close, and perform similarly. However, the scores achieved are much lower than the desired optimal.

The performances of the various configurations of the Doddle algorithm are disappointing: Doddle_RC is the strongest configuration, but it lags behind the existing algorithms. The other configurations of Doddle tend to do little better than the SBR lower bound.

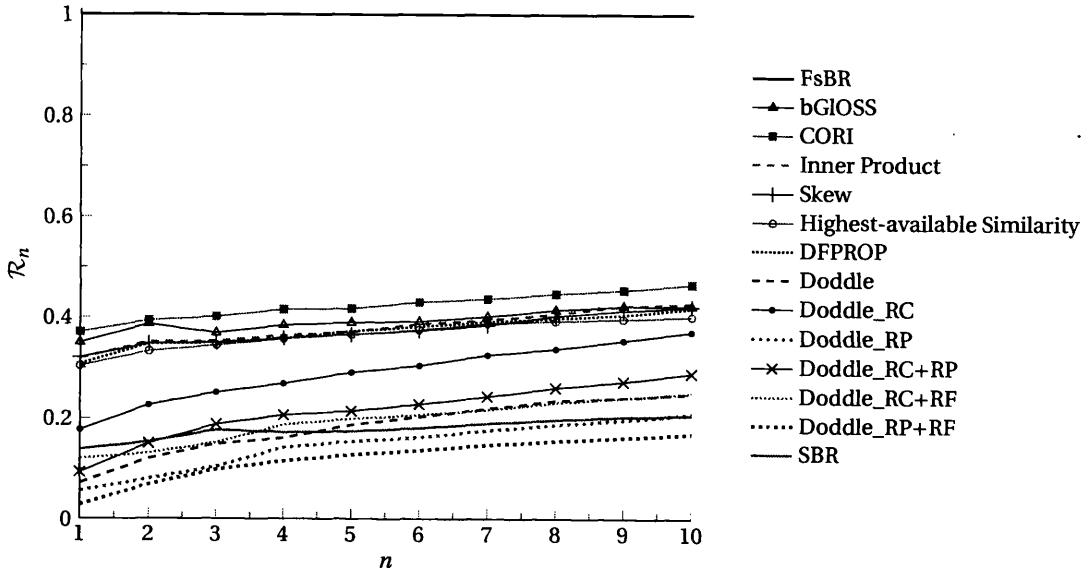
The scores achieved on the long queries, shown in Figure 6.1b, tend to be slightly higher than those from the short queries. CORI still shows strong performance, but with Inner Product and Highest-available Similarity being more competitive; indeed, those two algorithms do start stronger than CORI.

The configurations of Doddle perform better on the long queries, staying above the performance of SBR. The strongest configuration of Doddle is again Doddle_RC. Despite a slow start, it does become more competitive as we move down the rank positions.

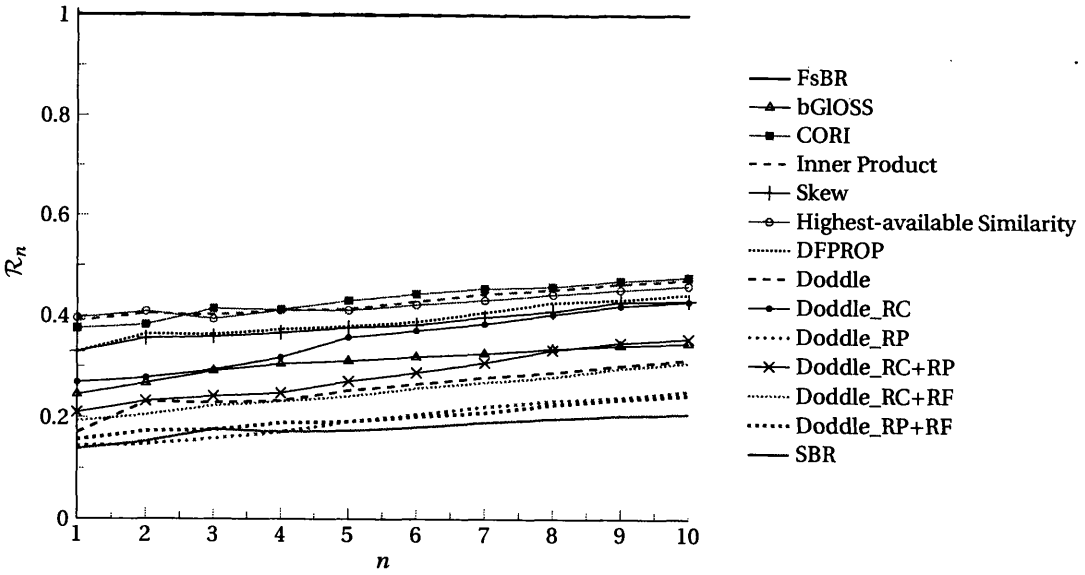
Figure 6.2 shows the $\hat{\mathcal{R}}_n$ performance measure scores, for both short and long queries. The performances of all the algorithms is very close, with CORI showing the strongest performance overall. The configurations of the Doddle algorithm appear to perform slightly worse than the collection selection algorithms, but are better than the SBR lower bound. The Doddle_RC algorithm is the best configuration of Doddle, and is competitive.

However, on the $\hat{\mathcal{R}}_n$ measure, the performance curves are much lower than the desired optimal. This suggests that the algorithms are not accurately selecting the most suitable collections at the given rank positions.

The scores achieved by the algorithms according to the $\mathcal{P}_n$ measure are given in Figure 6.3; $\mathcal{P}_n$ indicates the proportion of collections at a given rank position (averaged over all queries) that have some relevance to the query. As such, an algorithm could score highly here, but it does not necessarily mean that the collections chosen are the best available at that rank position.

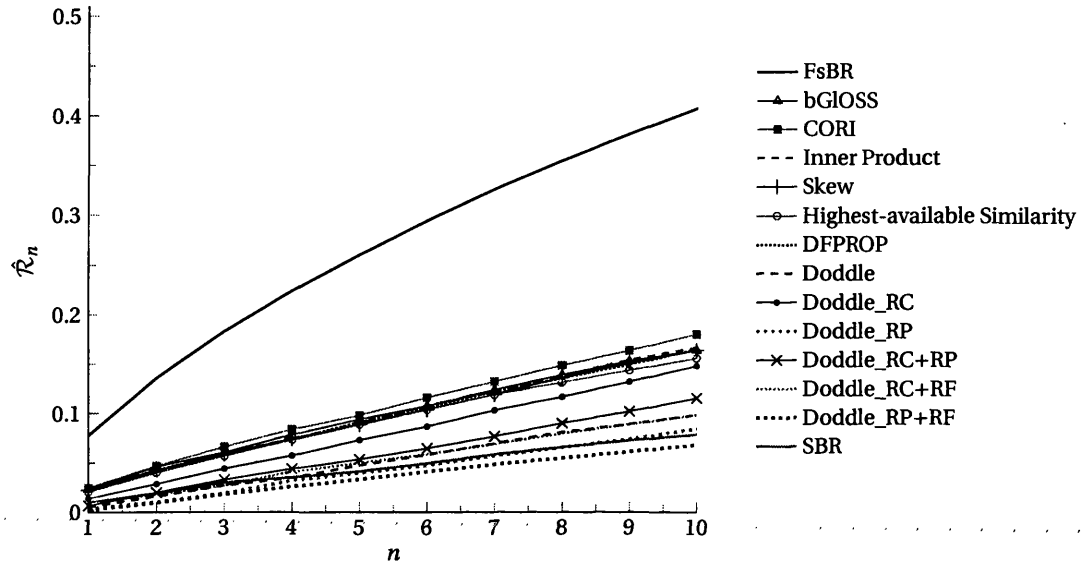


(a) Short queries.

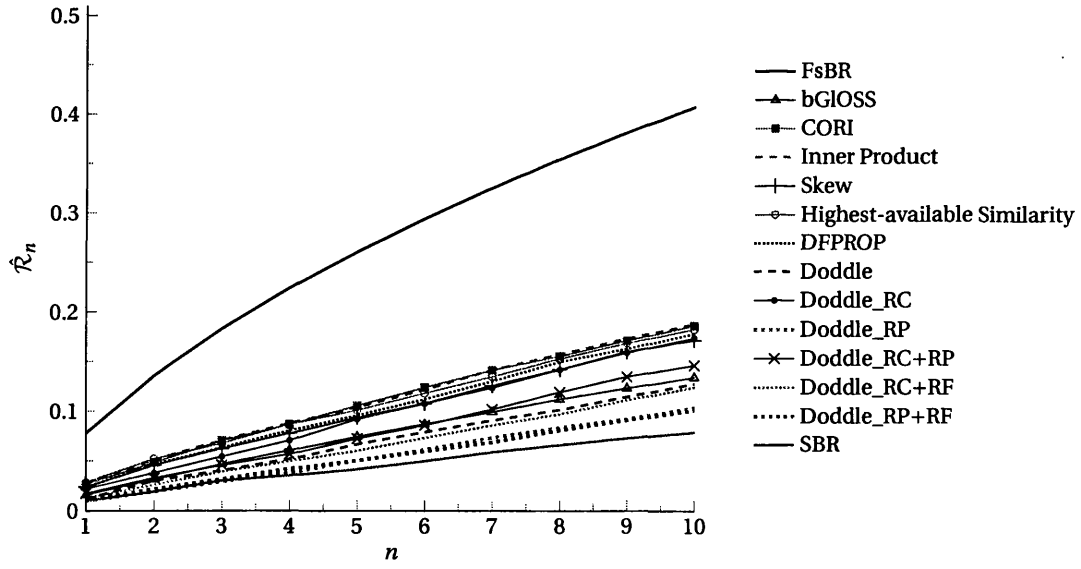


(b) Long queries.

Figure 6.1: The $\mathcal{R}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the SYM-236 test data set.



(a) Short queries.



(b) Long queries.

Figure 6.2: The $\hat{\mathcal{R}}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the SYM-236 test data set.

In Figure 6.3a we see the results from the execution of the short test queries. No algorithm achieves the best score of 1 at the first rank position. As such, on some occasions, the first ranked collection has no relevance to the query, according to the optimal ranking.

Overall, CORI scores best on the short queries, but the performances of all the collection selection algorithms are fairly close, and above that of the SBR lower bound. The performances of the various configurations of the Doddle algorithm are much poorer than those of the existing algorithms however, and fall below SBR. Of the Doddle configurations, Doddle_RC does best.

The $\mathcal{P}_n$ results from the execution of the long test queries are given in Figure 6.3b. Again, no algorithm achieves the best score of 1 at the first rank position. The best performers are CORI, Inner Product and Highest-available Similarity, which are very closely matched. The bGlOSS algorithm sees a large drop in performance, below the SBR lower bound. The configurations of the Doddle algorithm are again below SBR, with Doddle_RC the strongest of these.

Over these three performance measures, we have found CORI to be the most consistent algorithm, with Inner Product and Highest-available Similarity also performing well. Following the form shown on the rank correlation measures, the configurations of the Doddle algorithm have lagged behind the collection selection algorithms. Of these, Doddle_RC has been the strongest.

In the following section, we look closely at the performances of the algorithms within the top rank positions: correctly choosing the best collections in these positions is highly important for collection suggestion.

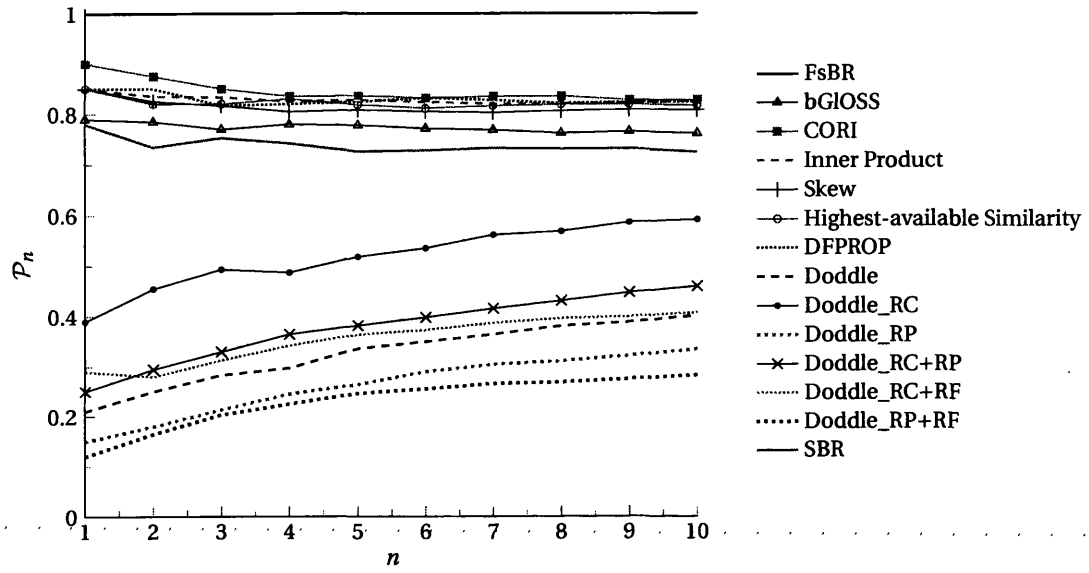
Top Rank Performance

In this section, we consider the performance of the selected collection selection algorithms and configurations of the Doddle algorithm, in terms of our Precision@5 and Correct@1 performance measures. These measures allow us to scrutinise algorithm performance within the top rank positions, where it is particularly important that algorithms choose the best collections.

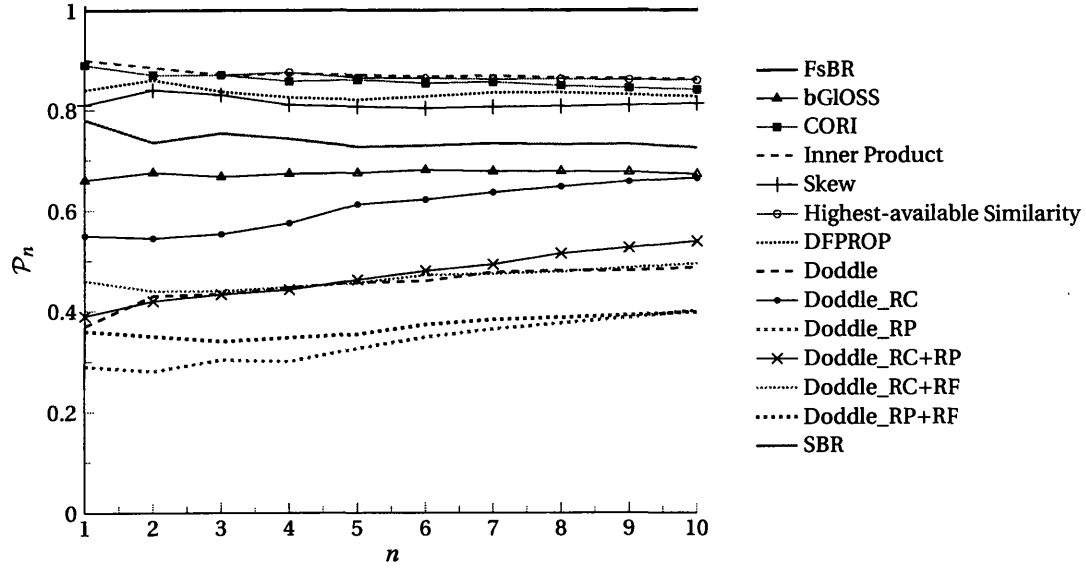
The scores achieved by the algorithms on these two measures are given in Table 6.4. Looking first at the Precision@5 scores, we see the scores are low for all algorithms. On the short queries, bGlOSS and CORI score highest. The scores of many of the configurations of the Doddle algorithm are lower than those of the collection selection algorithms. However, the best configuration (Doddle_RC) is competitive, and matches the likes of Inner Product and Highest-available Similarity.

On the long queries, the Precision@5 scores are slightly higher, but still poor overall. CORI scores highest, but Inner Product, Highest-available Similarity and Doddle_RC are also strong performers.

The scores achieved by the algorithms are also poor according to the Correct@1 measure, which shows the number of queries for which the algorithm correctly identified the first ranked collection. On short queries, CORI scores best, correctly identifying the top collection on only nine queries. Performance of the other collection selection algorithms is



(a) Short queries.



(b) Long queries.

Figure 6.3: The $\mathcal{P}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the SYM-236 test data set.

Table 6.4: The Precision@5 and Correct@1 scores achieved by selected existing algorithms and configurations of Doddle, on the SYM-236 test data set.

	Precision@5		**Correct@1**			
Algorithms	Q_s	Q_l	Q_s		Q_l	
bGlOSS	0.18	0.11	8	*(8%)*	2	*(2%)*
CORI	0.17	0.18	9	*(9%)*	12	*(12%)*
Inner Product	0.14	0.17	8	*(8%)*	13	*(13%)*
Skew	0.14	0.16	8	*(8%)*	9	*(9%)*
Highest-available Similarity	0.14	0.17	6	*(6%)*	15	*(15%)*
DFPROP	0.15	0.16	8	*(8%)*	9	*(9%)*
Doddle	0.10	0.12	2	*(2%)*	6	*(6%)*
Doddle_RC	0.14	0.17	7	*(7%)*	12	*(12%)*
Doddle_RP	0.09	0.10	1	*(1%)*	6	*(6%)*
Doddle_RC+RP	0.11	0.13	3	*(3%)*	11	*(11%)*
Doddle_RC+RF	0.10	0.12	3	*(3%)*	5	*(5%)*
Doddle_RP+RF	0.07	0.10	0	*(0%)*	5	*(5%)*
SBR	*0.03*	*0.03*	*2*	*(2%)*	*2*	*(2%)*

also similar, and Doddle_RC is not far behind, with seven correct first collections. The other configurations of Doddle perform poorly however.

Again, on long queries, there is an improvement in performance all-round. On this occasion, Highest-available Similarity scores best, with Inner Product, CORI and Doddle_RC close behind.

It is evident from these results that the SYM-236 data set is a challenging one. This may be due to the variety of content within individual collections, and therefore several collections appear to be very similar. This is an artefact of using the TREC document corpus to create synthetic collections.

There is much room for improvement in the top rank performance of these algorithms: the scores are much lower than we would like to see. A maximum of correctly identifying the top collection on only 15% of queries is not adequate.

We continue our evaluation using TREC-based test data in the following section.

6.3 UDC-236

Like the previous test data set, UDC-236 is also constructed from discs 1-3 of the TREC document corpus. Here, collections are grouped by original source and publication date into collections containing roughly the same number of documents. The spread of relevant documents within the collections is fairly even, however the collections may appear very similar in content (see Chapter 4).

We present the results achieved by the selected algorithms on this test data set, according to our choice of performance measures, in the following sections.

Rank Correlation

In Table 6.5, we present the Spearman rank correlation scores achieved by the tested algorithms. The scores show correlation between both the FsBR and SBR baselines, averaged over the set of 100 test queries.

We first look at the degree of correlation between the algorithm-produced collection ranking and the optimal FsBR collection ranking. All algorithms have a statistically significant correlation ($p < 0.05$) to FsBR, on both short and long queries. However, evaluating the Size-Based Ranking against FsBR does not give a significant correlation. This is due to the collections in this test data set being roughly equal in size, and thus ordering collections by size is ineffective.

On the short queries, CORI has the highest correlation score, while on long queries, Highest-available Similarity scores best. The configurations of the Doddle algorithm tend to score lower than the collection selection algorithms. However the differences are not statistically significant. Doddle_RC and Doddle_RC+RP are the strongest configurations of the Doddle algorithm here, and are fairly competitive with the collection selection algorithms.

As in the previous test data set (SYM-236) correlation scores over long queries are slightly higher than those over short queries, though the difference is not statistically significant. However, bGlOSS is an exception, which again performs significantly worse on long queries.

Looking at the SBR columns of Table 6.5, we see that none of the algorithms have a significant positive correlation to SBR; this is expected, due to the size attribute of this test data set.

In Table 6.6, we provide the average Blest and Da Costa weighted rank correlation scores for the algorithms, evaluated against the FsBR baseline. Here, the performances of the algorithms vary based on the performance measure and query length. For example, bGlOSS, Inner Product, CORI and Highest-available Similarity all have the highest correlation scores in a particular column.

Based on the Blest and Da Costa measures, the configurations of the Doddle algorithm are fairly competitive overall; several of the configurations such as Doddle_RC, Doddle_RP and Doddle_RC+RP score highly.

In the following section we further examine the performances of the selected algorithms with the recall and precision analogous measures.

Recall and Precision Analogues

In this section we discuss the effectiveness of the algorithms under test, in terms of the three performance measures analogous to precision and recall. For each of the measures, results are presented in the form of graphs, showing average scores, up to rank position ten. For reference, tables presenting scores at additional selected rank positions are given in Appendix D.

Table 6.5: The average Spearman rank correlations (over 100 test queries and the UDC-236 test data set) for selected existing algorithms and configurations of Doddle, comparing against both the FsBR and SBR baselines. Statistically significant correlations are given in bold.

	FsBR		**SBR**	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.43**	**0.24**	−0.16	−0.07
CORI	**0.48**	**0.47**	−0.30	−0.20
Inner Product	**0.46**	**0.51**	−0.21	−0.23
Skew	**0.42**	**0.45**	−0.15	−0.17
Highest-available Similarity	**0.47**	**0.53**	−0.23	−0.28
DFPROP	**0.40**	**0.44**	−0.13	−0.17
Doddle	**0.39**	**0.43**	−0.04	−0.07
Doddle_RC	**0.42**	**0.48**	−0.12	−0.16
Doddle_RP	**0.41**	**0.45**	−0.15	−0.18
Doddle_RC+RP	**0.42**	**0.47**	−0.13	−0.17
Doddle_RC+RF	**0.36**	**0.42**	0.04	0.01
Doddle_RP+RF	**0.36**	**0.41**	−0.02	−0.04
SBR	*−0.21*	*−0.21*	—	—

Table 6.6: The average Blest and Da Costa weighted rank correlations (over 100 test queries and the UDC-236 test data set) for selected existing algorithms and configurations of Doddle, comparing estimate rankings to FsBR. Statistically significant correlations are given in bold.

	Blest		**Da Costa**	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.54**	**0.47**	**0.50**	**0.52**
CORI	**0.52**	**0.50**	**0.52**	**0.51**
Inner Product	**0.52**	**0.57**	**0.50**	**0.55**
Skew	**0.49**	**0.52**	**0.46**	**0.49**
Highest-available Similarity	**0.46**	**0.53**	**0.50**	**0.57**
DFPROP	**0.47**	**0.51**	**0.44**	**0.48**
Doddle	**0.45**	**0.50**	**0.43**	**0.47**
Doddle_RC	**0.46**	**0.52**	**0.46**	**0.52**
Doddle_RP	**0.48**	**0.52**	**0.45**	**0.49**
Doddle_RC+RP	**0.48**	**0.52**	**0.46**	**0.51**
Doddle_RC+RF	**0.42**	**0.48**	**0.40**	**0.45**
Doddle_RP+RF	**0.44**	**0.48**	**0.40**	**0.44**
SBR	*−0.17*	*−0.17*	*−0.12*	*−0.12*

The performances of the algorithms according to the $\mathcal{R}_n$ measure, which indicates how well the best available collections are chosen, are shown in Figure 6.4. We first consider the algorithm performances over short queries, given in Figure 6.4a. Here, the Doddle_RC formulation of the Doddle algorithm starts highest, and stays strong. The CORI, Highest-available Similarity and Doddle_RC+RP algorithms also perform well.

Considering the performance of the algorithms on long queries (Figure 6.4b), we see that the scores are slightly higher than those achieved on the short queries. Here, Highest-available Similarity and Inner Product stand out as the strongest algorithms. However, the best configurations of Doddle, Doddle_RC and Doddle_RC+RP, are competitive and tend to match the performance of CORI.

We note here that although the algorithm performances stay above the performance of the SBR lower bound baseline, they are very distant from the optimal performance. As such, there is evidently much room for improvement.

Figure 6.5 shows the $\hat{\mathcal{R}}_n$ scores for the algorithms under test; Figure 6.5a gives the scores achieved on short queries, and Figure 6.5b gives the scores on long queries. In both graphs, there is little to choose between the algorithms.

On the short queries, Doddle_RC and Doddle_RC+RP just have the edge over the collection selection algorithms and other formulations of Doddle. These two algorithms are also strong on the long queries; however on this occasion, Inner Product and Highest-available Similarity appear slightly better overall.

As with the $\mathcal{R}_n$ measure, the $\hat{\mathcal{R}}_n$ scores for the algorithms are much lower than the desired optimal, supporting the view that the algorithms are not satisfactorily choosing the most suitable collections at a given rank position.

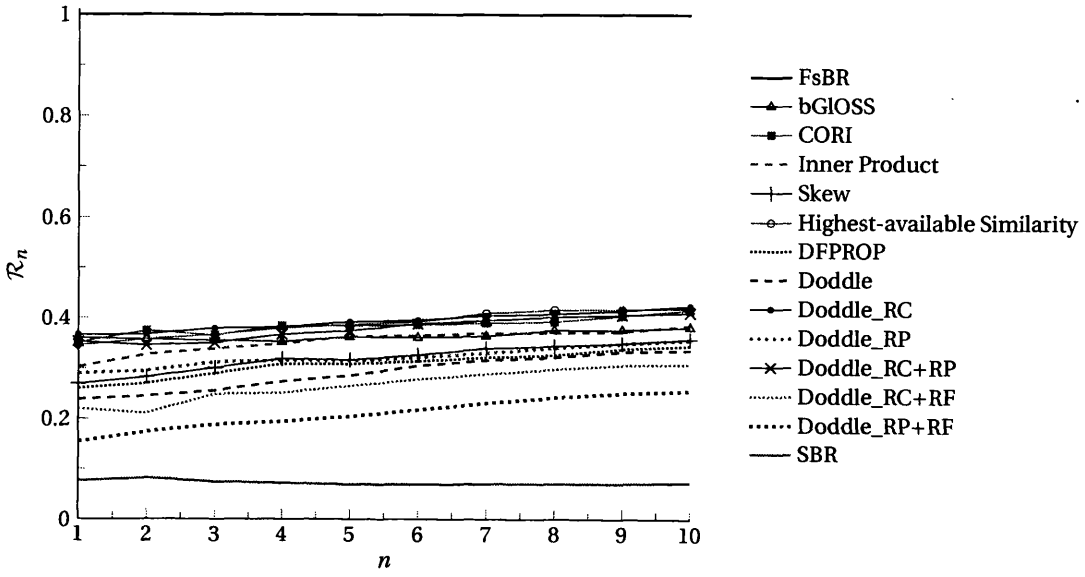
Finally in this section, we examine the performances of the algorithms with respect to the $\mathcal{P}_n$ measure. The graphs in Figure 6.6 show the $\mathcal{P}_n$ scores achieved by the algorithms. In Figure 6.6a we see the results from executing the short test queries. Note that, just as on the SYM-236 test data set in the previous section, no algorithm achieves the optimal score of 1 at the first rank position. As such, on some queries, the algorithms choose as their first ranked collection, a collection that FsBR has deemed to have no relevance to the query. The same is true of executing the long queries, shown in Figure 6.6b.

On short queries, all algorithms perform better than a size-based collection ranking. However, the stand-out algorithms are Highest-available Similarity, CORI and Doddle_RC, which tend to score higher than the other algorithms.

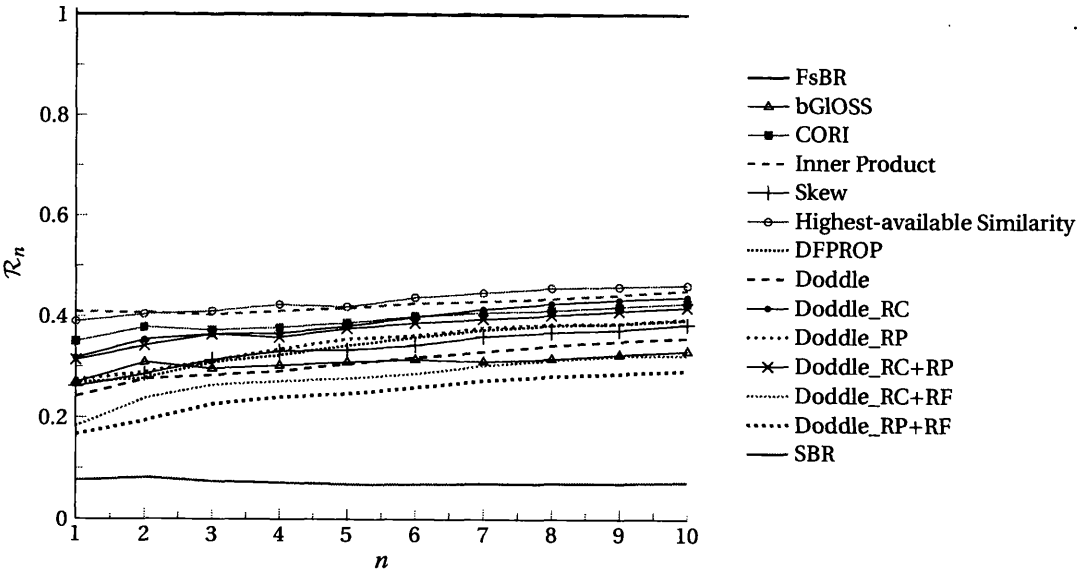
On long queries, performance scores tend to be higher overall, and Highest-available Similarity and CORI are still the stronger algorithms. The best configuration of Doddle is Doddle_RC, which starts a little slower than the top two algorithms. However, as we move down the rank positions, it does catch up (see Appendix D).

Over these three performance measures, we have found Highest-available Similarity, CORI, Inner Product and Doddle_RC to be the stronger algorithms. However, their performances are still distant from the desired optimal.

In the follow section we focus on performance within the top rank positions, where the correct ordering of collections takes priority.

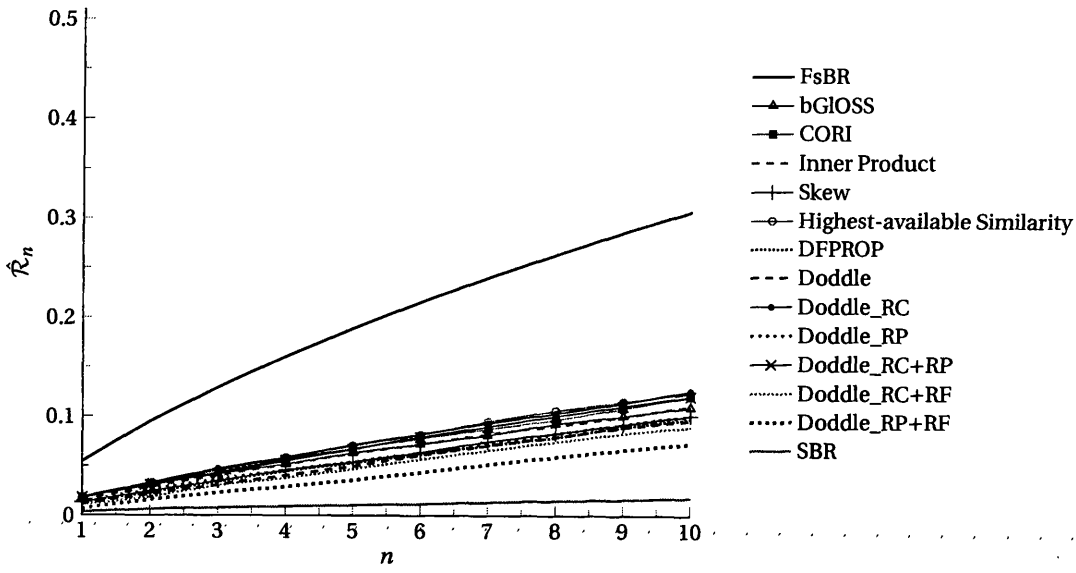


(a) Short queries.

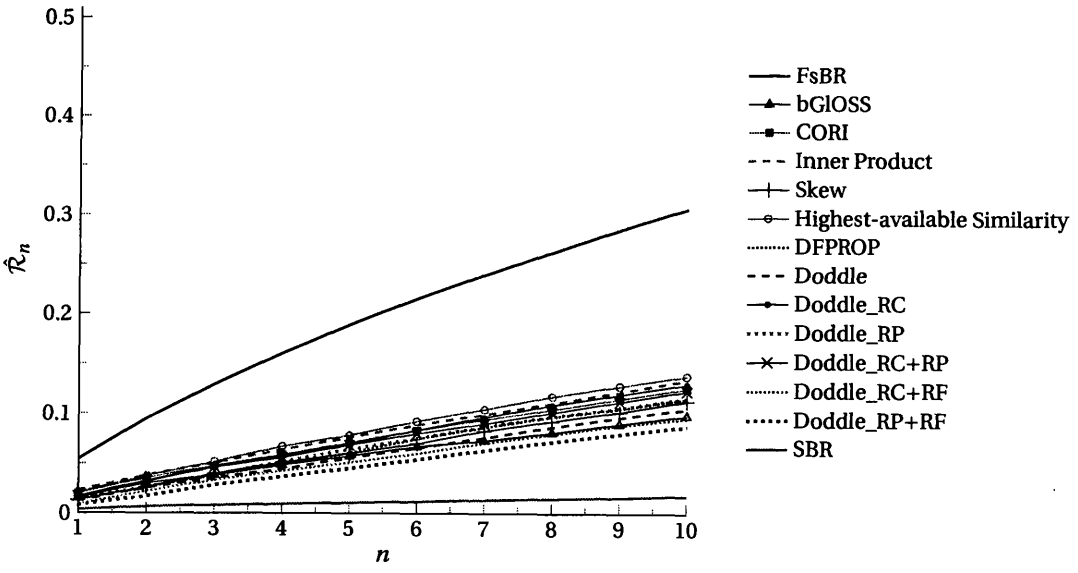


(b) Long queries.

Figure 6.4: The $\mathcal{R}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the UDC-236 test data set.

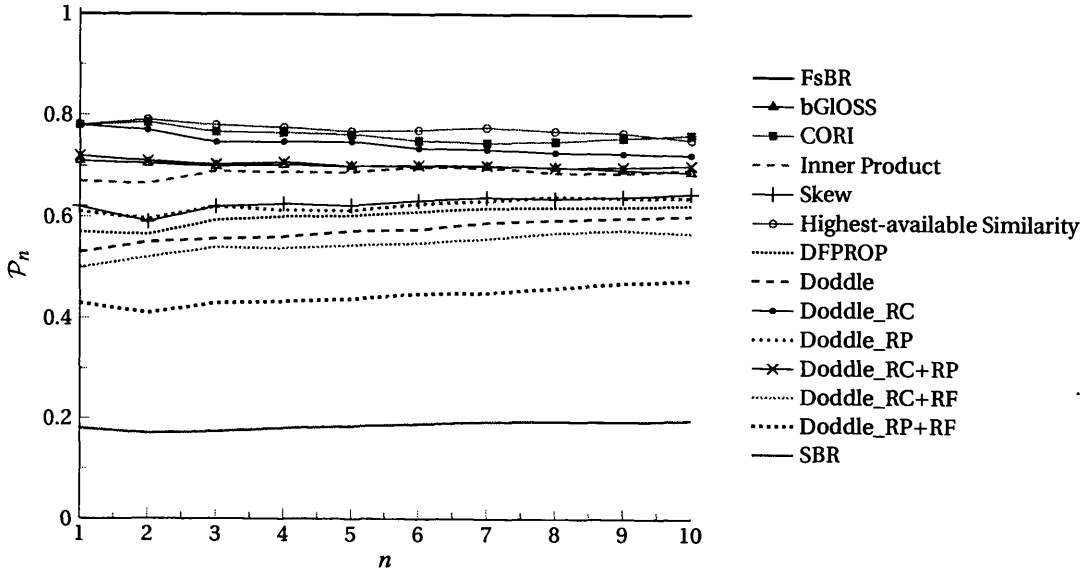


(a) Short queries.

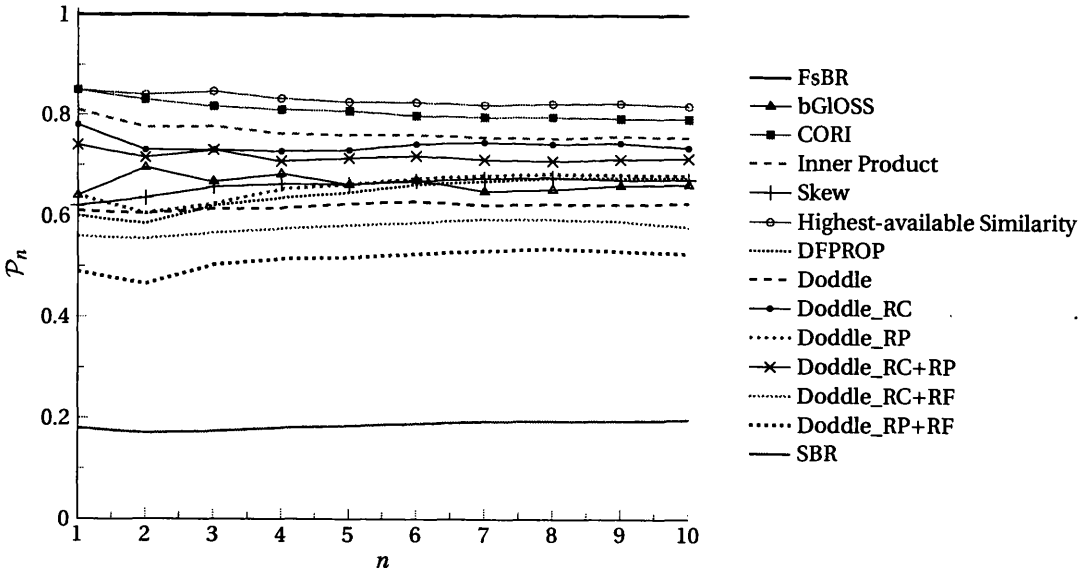


(b) Long queries.

Figure 6.5: The $\hat{\mathcal{R}}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the UDC-236 test data set.



(a) Short queries.



(b) Long queries.

Figure 6.6: The $\mathcal{P}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the UDC-236 test data set.

Top Rank Performance

In Table 6.7 we present the average Precision@5 and Correct@1 scores for the algorithms we are testing.

We first consider the Precision@5 scores, showing the proportion of collections within the top five rank positions of an algorithm-produced ranking, that should be there according to the optimal ranking. On short queries, the Precision@5 scores of the collection selection algorithms are very similar. However, Doddle_RC and Doddle_RC+RP score highest.

On long queries, the Doddle_RC and Doddle_RC+RP algorithms are still the best formulations of Doddle. However, Inner Product and Highest-available Similarity score highest overall.

We note however that we consider the Precision@5 scores, for all algorithms, to be low. For the algorithms to be effective, and consistently recommend suitable collections to the user, we would expect much higher precision within the top rank positions.

The Correct@1 scores are also very low. On short queries, bGlOSS, Doddle_RC and Doddle_RC+RP are the highest scorers, but only select the correct collection at the first rank position on 15% of the queries. On long queries, Inner Product manages to correctly identify the top collection on 16% of the queries. However, here the performance of the Doddle configurations drops from what it was on short queries.

Table 6.7: The Precision@5 and Correct@1 scores achieved by selected existing algorithms and configurations of Doddle, on the UDC-236 test data set.

	Precision@5		**Correct@1**			
Algorithms	Q_s	Q_l	Q_s		Q_l	
bGlOSS	0.15	0.09	15	*(15%)*	7	*(7%)*
CORI	0.16	0.15	10	*(10%)*	10	*(10%)*
Inner Product	0.16	0.17	11	*(11%)*	16	*(16%)*
Skew	0.14	0.13	10	*(10%)*	7	*(7%)*
Highest-available Similarity	0.16	0.17	11	*(11%)*	13	*(13%)*
DFPROP	0.13	0.14	10	*(10%)*	8	*(8%)*
Doddle	0.12	0.12	11	*(11%)*	5	*(5%)*
Doddle_RC	0.17	0.16	15	*(15%)*	8	*(8%)*
Doddle_RP	0.14	0.14	11	*(11%)*	8	*(8%)*
Doddle_RC+RP	0.17	0.15	15	*(15%)*	8	*(8%)*
Doddle_RC+RF	0.12	0.10	9	*(9%)*	2	*(2%)*
Doddle_RP+RF	0.09	0.09	4	*(4%)*	2	*(2%)*
SBR	*0.02*	*0.02*	*3*	*(3%)*	*3*	*(3%)*

6.4 UBC-100

The UBC-100 test data set contains 100 collections of roughly equal data size, with each collection containing documents from one source. This results in collections containing various numbers of documents, and collections that are larger (in terms of number of documents) than the SYM-236 and UDC-236 data sets.

As discussed in Chapter 4, in the UBC-100 data set, some of the larger collections have only a small share of the relevant documents. As such, an algorithm that tends towards a size-based ranking may perform poorly here.

We present the algorithm results over this test data set, according to each of our performance measures, in the following sections.

Rank Correlation

The Spearman rank correlation scores for the algorithms under test are given in Table 6.8; the 'FsBR' column shows the degree of correlation with the optimal baseline, while the 'SBR' column shows the degree of correlation with the lower bound baseline.

On both short and long queries, all algorithms show statistically significant positive correlation ($p < 0.05$) to the FsBR baseline, as does SBR. Two collection selection algorithms, CORI and Inner Product, show the highest correlation with FsBR, and are significantly better than Doddle_RC+RF and Doddle_RP+RF. The strongest configuration of Doddle is Dod-

Table 6.8: The average Spearman rank correlations (over 100 test queries and the UBC-100 test data set) for selected existing algorithms and configurations of Doddle, comparing against both the FsBR and SBR baselines. Statistically significant correlations are given in bold.

	FsBR		**SBR**	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.56**	**0.35**	0.12	0.06
CORI	**0.62**	**0.61**	**0.39**	**0.32**
Inner Product	**0.61**	**0.65**	**0.44**	**0.41**
Skew	**0.56**	**0.59**	**0.47**	**0.46**
Highest-available Similarity	**0.47**	**0.52**	**0.65**	**0.66**
DFPROP	**0.55**	**0.59**	**0.50**	**0.51**
Doddle	**0.42**	**0.47**	−0.15	−0.16
Doddle_RC	**0.52**	**0.59**	0.16	0.15
Doddle_RP	**0.45**	**0.51**	−0.17	−0.18
Doddle_RC+RP	**0.50**	**0.56**	−0.02	−0.04
Doddle_RC+RF	**0.38**	**0.45**	−0.12	−0.14
Doddle_RP+RF	**0.34**	**0.40**	−0.30	−0.30
SBR	***0.18***	***0.18***	—	—

dle_RC; it does not score as highly as CORI and Inner Product, but it is not significantly poorer.

As with the SYM-236 and UDC-236 data sets, correlation scores with FsBR tend to be slightly higher on long queries than on short queries (bGlOSS excluded). However, the differences in scores are not statistically significant.

Looking at the 'SBR' column of Table 6.8, we see that most of the collection selection algorithms are significantly correlated to SBR, while the configurations of Doddle are not. Since the collection selection algorithms tend to match the optimal FsBR baseline a little closer than the Doddle configurations, it is likely there is some bias towards larger collections within the test data set. Although the largest collections contain few relevant documents, the collections with the most relevant documents and matching the most queries tend to be larger than those with less relevant documents and matching less queries.

The Blest and Da Costa weighted rank correlation scores (to FsBR), shown in Table 6.9, tell a similar story to the Spearman correlation scores. That is, all algorithms exhibit a significant correlation to FsBR, with CORI and Inner Product having the best scores. Doddle_RC is the strongest configuration of Doddle, but in general it scores lower than the collection selection algorithms.

In the following section we consider the performances of the algorithms in terms of the recall and precision based measures.

Table 6.9: The average Blest and Da Costa weighted rank correlations (over 100 test queries and the UBC-100 test data set) for selected existing algorithms and configurations of Doddle, comparing estimate rankings to FsBR. Statistically significant correlations are given in bold.

	Blest		**Da Costa**	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.53**	**0.25**	**0.60**	**0.55**
CORI	**0.59**	**0.59**	**0.63**	**0.63**
Inner Product	**0.58**	**0.63**	**0.62**	**0.67**
Skew	**0.54**	**0.56**	**0.58**	**0.61**
Highest-available Similarity	**0.44**	**0.49**	**0.46**	**0.52**
DFPROP	**0.53**	**0.56**	**0.57**	**0.60**
Doddle	**0.39**	**0.43**	**0.44**	**0.49**
Doddle_RC	**0.50**	**0.56**	**0.54**	**0.61**
Doddle_RP	**0.42**	**0.46**	**0.48**	**0.53**
Doddle_RC+RP	**0.47**	**0.52**	**0.52**	**0.58**
Doddle_RC+RF	**0.36**	**0.41**	**0.40**	**0.46**
Doddle_RP+RF	**0.31**	**0.36**	**0.35**	**0.41**
SBR	***0.17***	***0.17***	*0.16*	*0.16*

Recall and Precision Analogues

In this section we discuss the performances of the algorithms, according to the recall and precision analogues. We focus exclusively on performance within the top ten rank positions: correctly identifying the most suitable collections for the user's query is important; as such, performance outside the first few collections is irrelevant. For reference however, we provide scores achieved at selected other rank positions, for these measures, in Appendix D.

We first consider the algorithm performances according to the $\mathcal{R}_n$ measure, which indicates how closely an algorithm selects the best available collection, at a given rank position. Scores achieved by the selected algorithms on this measure are given in Figure 6.7.

The performances of the algorithms vary depending on whether we are utilising short or long test queries. For example, on short queries (see Figure 6.7a) bGlOSS performs best overall; Inner Product starts well, and remains competitive. The strongest configuration of Doddle is Doddle_RC, which competes well with CORI and DFPROP.

On long queries however (see Figure 6.7b), Inner Product shows a much stronger performance, and tends to lead the other algorithms. Doddle_RC and Doddle_RC+RP score lower than Inner Product initially, but become very competitive as we move down the rank positions.

We note that all algorithms show much stronger performance than the Size-Based Ranking, which performs particularly poorly on this data set and performance measure.

We next look at the algorithm scores for the $\hat{\mathcal{R}}_n$ measure, which shows how much of the merit associated with the collections has been accumulated at each rank position: an alternative measure of whether the best collections are being chosen by the algorithms. Figure 6.8 shows the scores achieved by the algorithms for this measure.

We notice that for both short and long queries, the algorithm performances are fairly close. However, the graphs tell a similar story as those for the $\mathcal{R}_n$ results. That is, on short queries, bGlOSS performs best overall, and Doddle_RC is competitive with algorithms such as CORI, but slightly behind bGlOSS. On long queries however, the performance of bGlOSS drops, and Doddle_RC, Doddle_RC+RP and Inner Product come to the fore. SBR is again very poor.

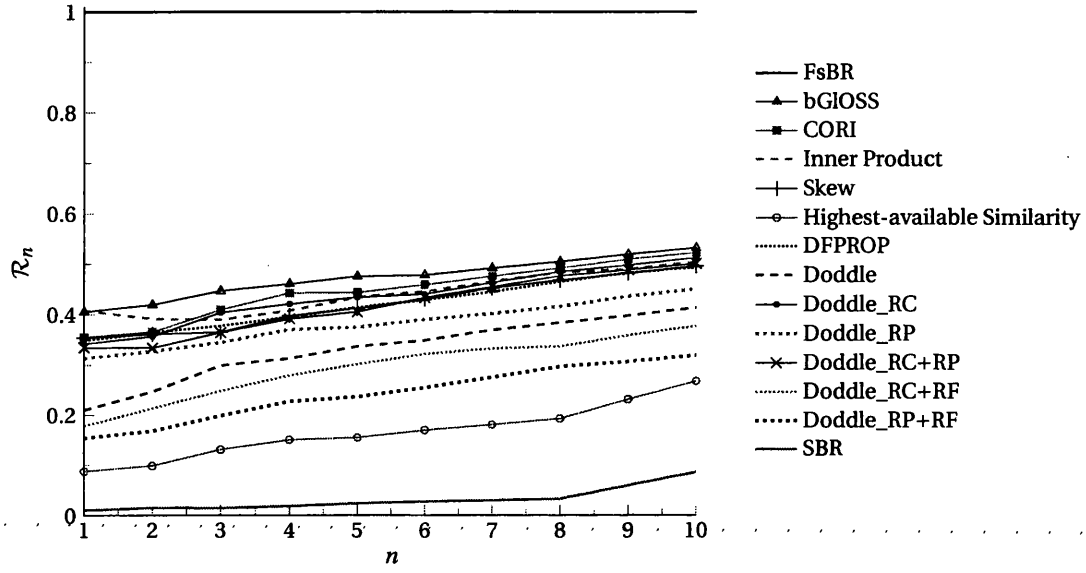
Finally, the algorithm performances according to the $\mathcal{P}_n$ performance measure are given in the graphs in Figure 6.9. This measure is analogous to precision, and indicates the proportion of collections at each rank position that have relevance to the query (averaged over all test queries), according to the optimal ranking.

Here, at the first rank position, an optimal $\mathcal{P}_n$ score is 1; however, none of the algorithms achieved this, suggesting they sometimes rank first a collection the optimal ranking has deemed to have no relevance.

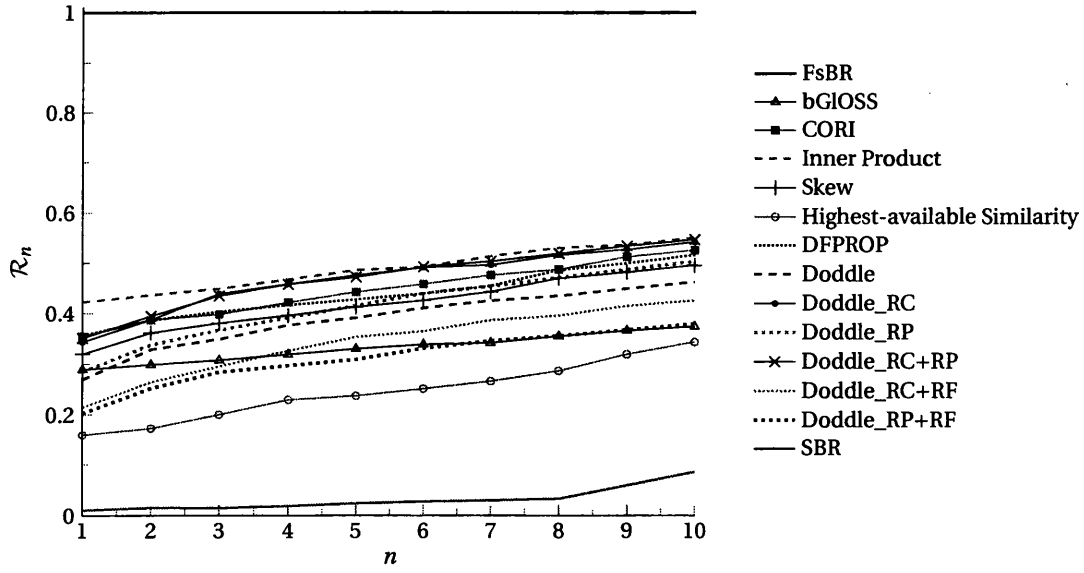
The performances of the algorithms again vary: some perform better on short queries than long queries. On the short queries, Inner Product and CORI score highest, and are fairly evenly matched, followed by DFPROP. The best configuration of Doddle is Doddle_RC, but it tends to perform worse than the best collection selection algorithms.

On long queries, Inner Product is clearly the stronger algorithm. Below Inner Product are CORI, DFPROP, Skew and Doddle_RC, which all perform similarly.

We look more closely at performance in the top rank positions, in the following sections.

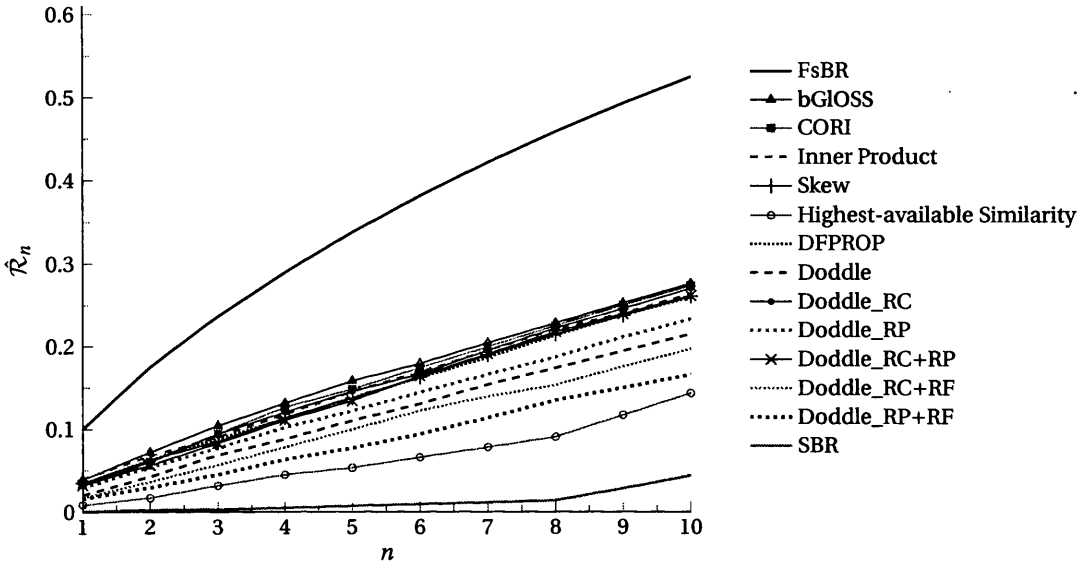


(a) Short queries.

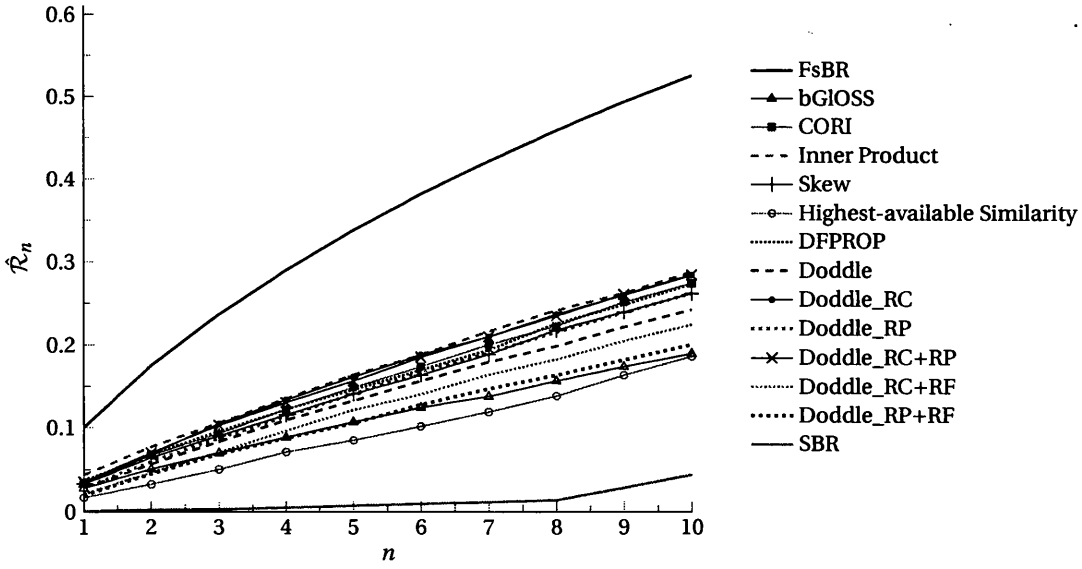


(b) Long queries.

Figure 6.7: The $\mathcal{R}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the UBC-100 test data set.

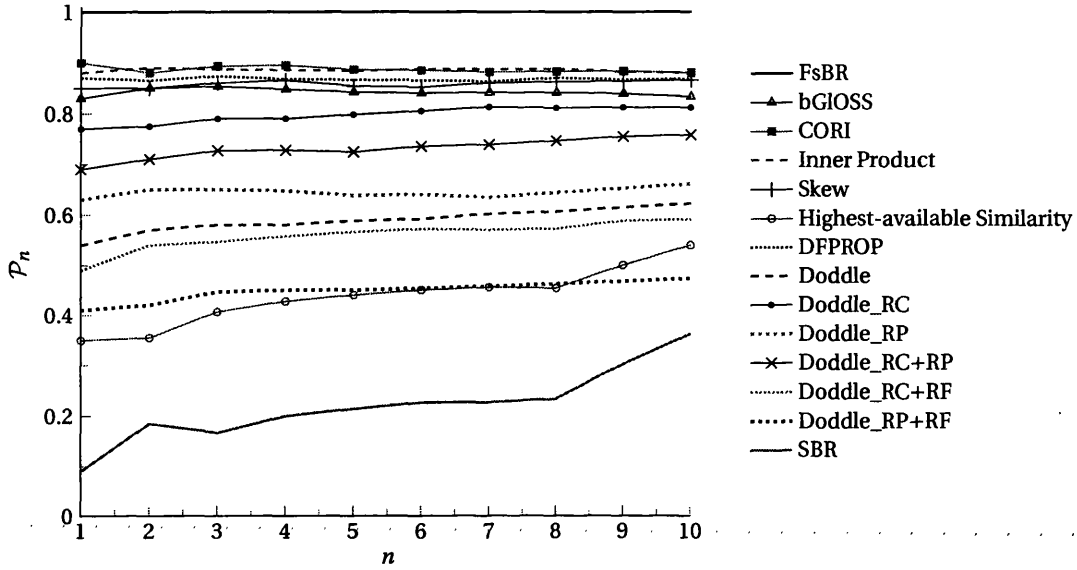


(a) Short queries.

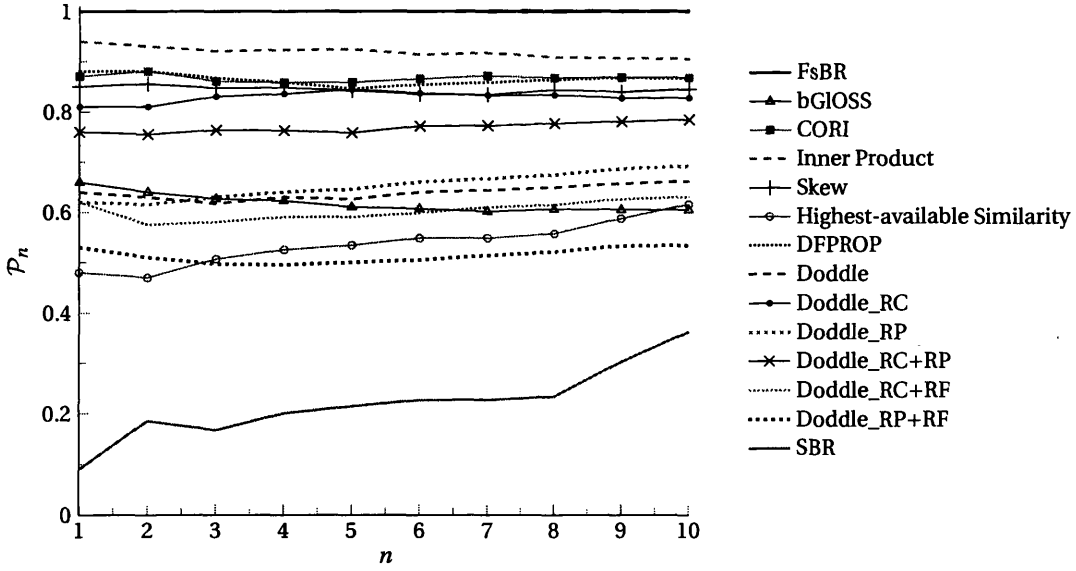


(b) Long queries.

Figure 6.8: The $\hat{\mathcal{R}}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the UBC-100 test data set.



(a) Short queries.



(b) Long queries.

Figure 6.9: The $\mathcal{P}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the UBC-100 test data set.

Top Rank Performance

The scores achieved by the tested algorithms, on the Precision@5 and Correct@1 performance measures, are given in Table 6.10. We first consider the Precision@5 scores.

Precision@5 indicates the proportion of collections within the top five rank positions, that should be present, according to the optimal ranking (averaged over all test queries). On short queries, bGlOSS scores highest, with Doddle_RC being the most competitive of the Doddle configurations. However, on long queries the configurations of Doddle perform particularly well: Doddle_RC+RP scores highest, followed by Doddle_RC and Inner Product.

The Correct@1 measure shows the number and percentage of queries for which the algorithms correctly identified the first-ranked collection. On short queries, bGlOSS again scores best. However it predicts the top collection on only 15% of the queries. Inner Product and Doddle_RP follow, with 13% and 12% respectively. On the long test queries, Inner Product has the highest score (14%); the most competitive configurations of Doddle are Doddle_RC and Doddle_RC+RP, with 10%.

We observe here that the performance of Highest-available Similarity has been particularly poor on this test data set, often showing some of the lowest scores for the performance measures. In contrast, it was one of the stronger algorithms on the previous two data sets (SYM-236 and UDC-236).

Table 6.10: The Precision@5 and Correct@1 scores achieved by selected existing algorithms and configurations of Doddle, on the UBC-100 test data set.

	Precision@5		**Correct@1**			
Algorithms	Q_s	Q_l	Q_s		Q_l	
bGlOSS	0.26	0.16	15	*(15%)*	7	*(7%)*
CORI	0.23	0.21	11	*(11%)*	8	*(8%)*
Inner Product	0.21	0.25	13	*(13%)*	14	*(14%)*
Skew	0.19	0.20	9	*(9%)*	7	*(7%)*
Highest-available Similarity	0.07	0.13	2	*(2%)*	5	*(5%)*
DFPROP	0.19	0.22	9	*(9%)*	10	*(10%)*
Doddle	0.18	0.22	5	*(5%)*	6	*(6%)*
Doddle_RC	0.23	0.26	8	*(8%)*	10	*(10%)*
Doddle_RP	0.19	0.24	12	*(12%)*	8	*(8%)*
Doddle_RC+RP	0.21	0.27	10	*(10%)*	10	*(10%)*
Doddle_RC+RF	0.16	0.20	3	*(3%)*	3	*(3%)*
Doddle_RP+RF	0.12	0.17	4	*(4%)*	2	*(2%)*
SBR	*0.00*	*0.00*	0	*(0%)*	0	*(0%)*

6.5 2LDB-60COL

The 2LDB-60COL test data set is formulated from the UDC-100 data set, by combining a selection of collections to form two very large collections. As such, as discussed in Chapter 4, the data set contains many small collections of similar size, and the two large collections. The distribution of relevant documents is such that a size-based ranking of collections may be an adequate ranking strategy here.

We discuss algorithm performances over this data set in the following sections.

Rank Correlation

In this section we discuss the performances of the algorithms, in terms of the rank correlation measures. These measures indicate the similarity between a baseline collection ranking, and a collection ranking produced by an algorithm.

We first look at similarity scores according to the Spearman rank correlation coefficient, given in Table 6.11. The 'FsBR' column gives algorithm correlation with the optimal baseline, while the 'SBR' column gives algorithm correlation with the lower bound baseline.

From Table 6.11, we see that all algorithms tested (including SBR) have significant positive correlation ($p < 0.05$) with FsBR, on both short and long queries. The algorithms showing the strongest correlation with FsBR are CORI and Inner Product. The best configuration

Table 6.11: The average Spearman rank correlations (over 100 test queries and the 2LDB-60COL test data set) for selected existing algorithms and configurations of Doddle, comparing against both the FsBR and SBR baselines. Statistically significant correlations are given in bold.

	FsBR		**SBR**	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.56**	**0.37**	**0.25**	0.15
CORI	**0.63**	**0.63**	**0.48**	**0.43**
Inner Product	**0.61**	**0.65**	**0.55**	**0.53**
Skew	**0.57**	**0.60**	**0.59**	**0.58**
Highest-available Similarity	**0.51**	**0.56**	**0.67**	**0.68**
DFPROP	**0.56**	**0.59**	**0.61**	**0.62**
Doddle	**0.43**	**0.49**	−0.09	−0.08
Doddle_RC	**0.54**	**0.60**	**0.21**	**0.22**
Doddle_RP	**0.45**	**0.51**	−0.13	−0.13
Doddle_RC+RP	**0.51**	**0.57**	0.04	0.03
Doddle_RC+RF	**0.40**	**0.47**	−0.05	−0.04
Doddle_RP+RF	**0.34**	**0.41**	−0.23	−0.21
SBR	***0.24***	***0.24***	—	—

Table 6.12: The average Blest and Da Costa weighted rank correlations (over 100 test queries and the 2LDB-60COL test data set) for selected existing algorithms and configurations of Doddle, comparing estimate rankings to FsBR. Statistically significant correlations are given in bold.

	Blest		**Da Costa**	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.50**	**0.22**	**0.60**	**0.55**
CORI	**0.58**	**0.58**	**0.64**	**0.64**
Inner Product	**0.56**	**0.61**	**0.61**	**0.66**
Skew	**0.53**	**0.56**	**0.58**	**0.60**
Highest-available Similarity	**0.47**	**0.52**	**0.50**	**0.55**
DFPROP	**0.52**	**0.55**	**0.57**	**0.60**
Doddle	**0.38**	**0.43**	**0.45**	**0.51**
Doddle_RC	**0.50**	**0.55**	**0.56**	**0.62**
Doddle_RP	**0.41**	**0.45**	**0.47**	**0.53**
Doddle_RC+RP	**0.47**	**0.52**	**0.53**	**0.59**
Doddle_RC+RF	**0.36**	**0.41**	**0.42**	**0.48**
Doddle_RP+RF	**0.30**	**0.35**	**0.36**	**0.42**
SBR	***0.25***	***0.25***	***0.22***	***0.22***

of Doddle is Doddle_RC; although its correlation with FsBR is lower than the top collection selection algorithms, there is no statistically significant difference.

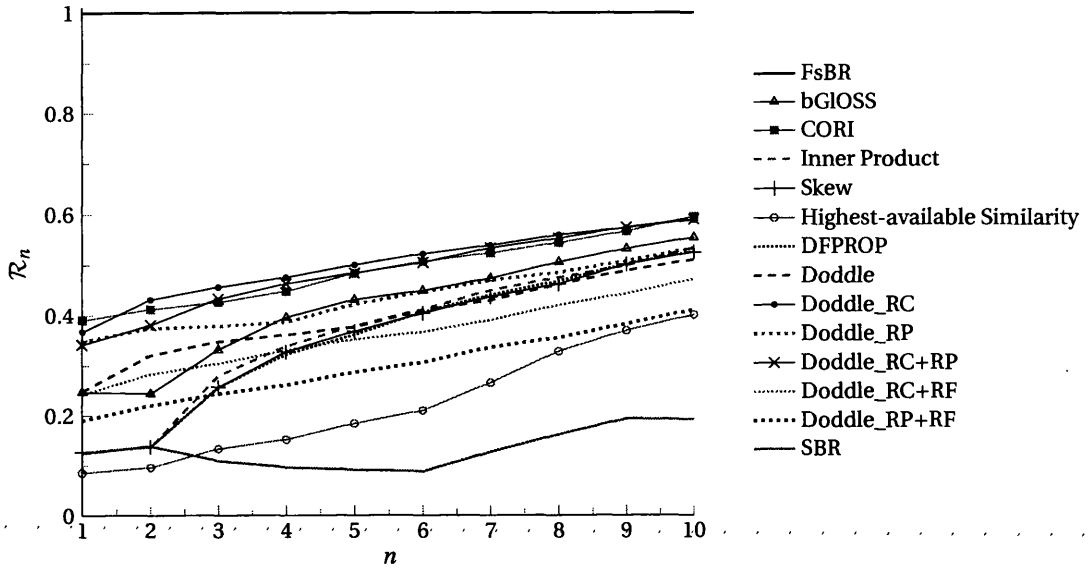
Looking at the 'SBR' column of Table 6.11, we see that the collection selection algorithms also show significant correlation to a size-based ranking. This is also true of Doddle_RC, the only configuration of Doddle to do so. However, its correlation with SBR is lower than those of the collection selection algorithms. The correlations with the Size-Based Ranking support the view that the data set has a bias towards the larger collections.

The observations made on the Spearman rank correlation results are supported by the Blest and Da Costa weighted rank correlation scores, given in Table 6.12. All algorithms are again significantly correlated to FsBR, with CORI and Inner Product scoring highest. Once again, Doddle_RC is the strongest configuration of the Doddle algorithm, and is fairly competitive.

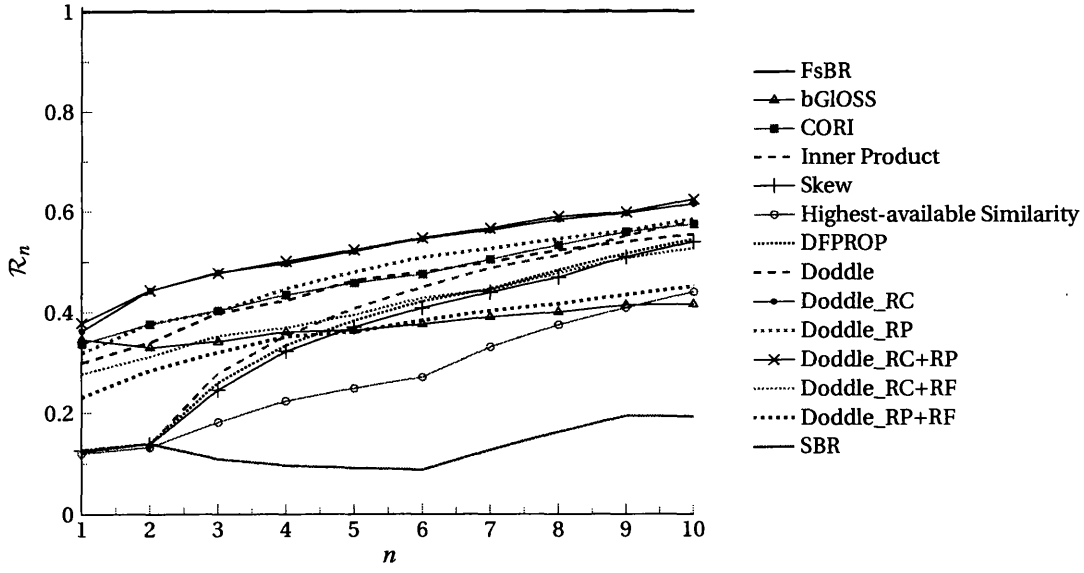
We investigate the performance of the algorithms on the 2LDB-60COL test data set further in the following section, with the recall and precision analogous measures.

Recall and Precision Analogues

In Figure 6.10, we present the $\mathcal{R}_n$ scores achieved within the first ten rank positions, by the algorithms under test. Recall that $\mathcal{R}_n$ indicates how well an algorithm selects the best available collection at a given rank position.



(a) Short queries.



(b) Long queries.

Figure 6.10: The $\mathcal{R}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the 2LDB-60COL test data set.

On the short test queries (see Figure 6.10a), CORI scores best at the first rank position; however, Doddle_RC and Doddle_RC+RP appear to be the strongest algorithms overall. These two algorithms are also best overall on the long test queries (see Figure 6.10b), and are noticeably stronger than CORI.

Notably, while Inner Product scored highly on the rank correlation measures, its performance according to the $\mathcal{R}_n$ measure is poor, although it does exceed the lower bound SBR baseline.

The $\hat{\mathcal{R}}_n$ measure is an alternative indicator of how accurately an algorithm selects the best collections. The results of this measure are given as graphs in Figure 6.11, and tell a similar story to the $\mathcal{R}_n$ graphs. On short queries, Doddle_RC, Doddle_RC+RP and CORI perform best, and are closely matched. However on long queries, the performance of CORI drops slightly, relative to that of the two best Doddle configurations. Inner Product is again sub-par, based on its previous rank correlation scores.

In Figure 6.12 we present the performances of the algorithms according to the $\mathcal{P}_n$ measure. $\mathcal{P}_n$ indicates whether the collections an algorithm has selected, at a given rank position, are relevant to the query, according to the optimal. The findings here differ from those of the $\mathcal{R}_n$ and $\hat{\mathcal{R}}_n$ measures.

Here, Inner Product performs best overall, followed by DFPROP and Skew; while CORI starts slower, it does match these two algorithms as we move down the rank positions. However, Doddle_RC and Doddle_RC+RP lag behind these.

On this data set, several algorithms (Inner Product, Skew, DFPROP and SBR) achieve the optimal score of 1, at the first rank position. This shows that the first ranked collection always has some relevance to the query (although it may not be the most relevant collection). We note that as SBR achieves this optimal score, this suggests that the largest collection always has relevance to the test queries. However, based on its poor performance according to $\mathcal{R}_n$ and $\hat{\mathcal{R}}_n$, the largest collection is rarely the best. Indeed, this is also true of Inner Product on this data set: while it chooses collections that have some relevance, they are not the most relevant.

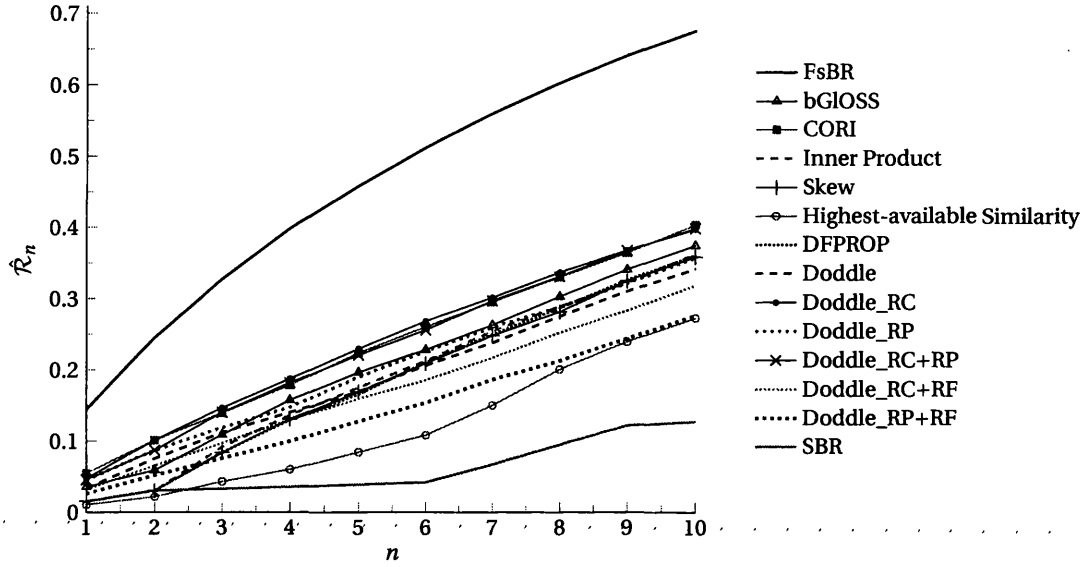
We investigate the performance of these algorithms further in the following section, with our Precision@5 and Correct@1 measures.

Top Rank Performance

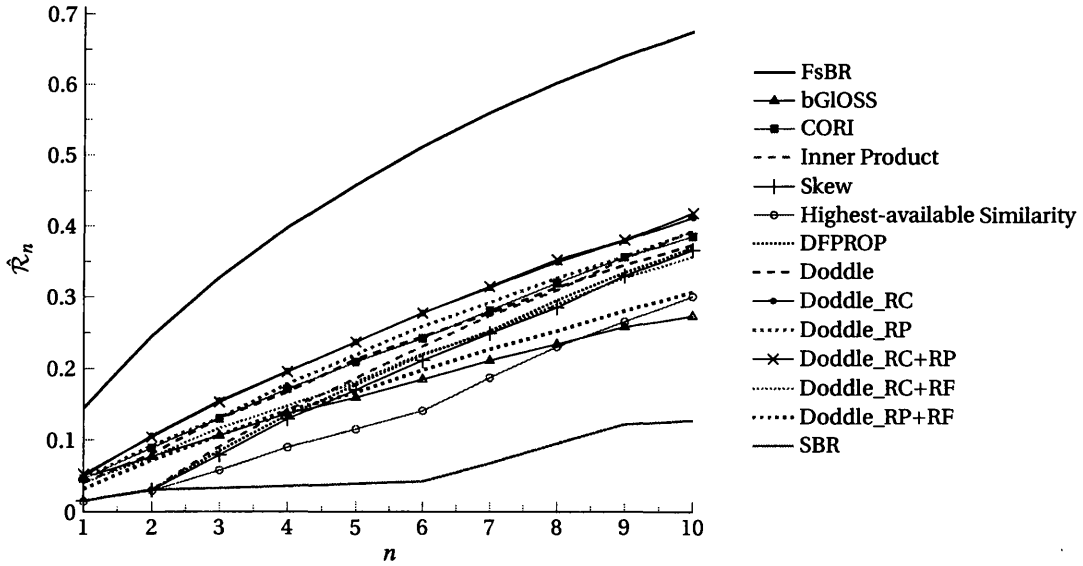
In this section, we consider the effectiveness of the algorithms at the top five, and first rank positions. That is, with our Precision@5 measure we investigate how accurately the algorithms identify collections that belong in the first five rank positions. In addition, Correct@1 checks how often the algorithms correctly identify the best collection. The scores achieved by the algorithms for these measures are given in Table 6.13.

Looking first at the Precision@5 scores, we see that the various Doddle configurations tend to score highest overall, with Doddle_RC and Doddle_RC+RP achieving the best scores. The strongest collection selection algorithm is CORI, however it does not quite match the two Doddle configurations.

On the Correct@1 measure, the collection selection algorithms perform poorly overall: many fail to identify the top ranked collection on even a single query. On short queries,

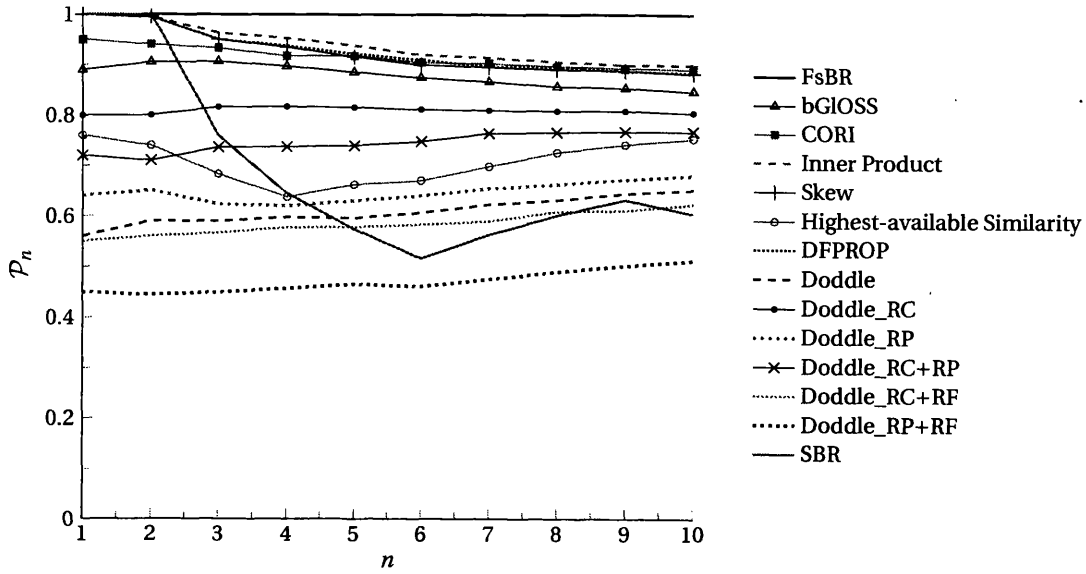


(a) Short queries.

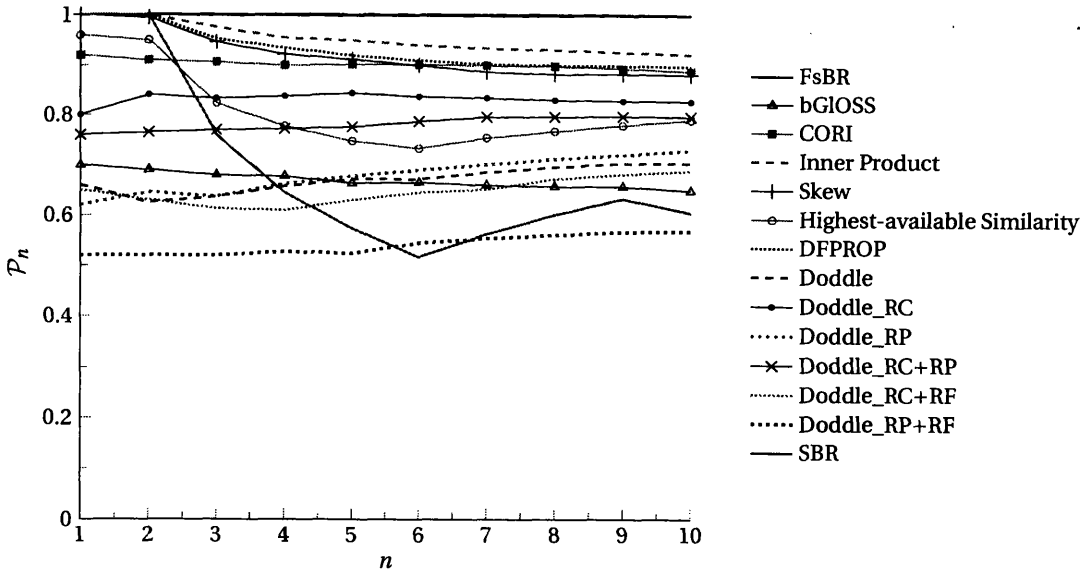


(b) Long queries.

Figure 6.11: The $\hat{\mathcal{R}}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the 2LDB-60COL test data set.



(a) Short queries.



(b) Long queries.

Figure 6.12: The $\mathcal{P}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the 2LDB-60COL test data set.

Table 6.13: The Precision@5 and Correct@1 scores achieved by selected existing algorithms and configurations of Doddle, on the 2LDB-60COL test data set.

	Precision@5		Correct@1			
Algorithms	Q_s	Q_l	Q_s		Q_l	
bGlOSS	0.23	0.21	10	*(10%)*	14	*(14%)*
CORI	0.27	0.26	11	*(11%)*	6	*(6%)*
Inner Product	0.18	0.22	0	*(0%)*	0	*(0%)*
Skew	0.18	0.19	0	*(0%)*	0	*(0%)*
Highest-available Similarity	0.08	0.12	0	*(0%)*	0	*(0%)*
DFPROP	0.17	0.20	0	*(0%)*	0	*(0%)*
Doddle	0.22	0.29	8	*(8%)*	9	*(9%)*
Doddle_RC	0.31	0.32	10	*(10%)*	11	*(11%)*
Doddle_RP	0.26	0.31	15	*(15%)*	12	*(12%)*
Doddle_RC+RP	0.30	0.33	14	*(14%)*	13	*(13%)*
Doddle_RC+RF	0.21	0.23	7	*(7%)*	8	*(8%)*
Doddle_RP+RF	0.16	0.22	6	*(6%)*	7	*(7%)*
SBR	*0.01*	*0.01*	*0*	*(0%)*	*0*	*(0%)*

the best collection selection algorithm is CORI, which correctly identifies the top collection on 11% of the test queries. However, Doddle_RP exceeds this, with 15%. On long queries, bGlOSS performs best, identifying the best collection on 14% of queries; Doddle_RC+RP follows with 13%.

As with the previous TREC-based test data sets, we consider the Precision@5 and Correct@1 scores to be fairly low: improvement is required to be effective in an operational environment.

6.6 AP-WSJ-60COL

In this section, we examine the effectiveness of the selected algorithms over the AP-WSJ-60COL test data set. Like 2LDB-60COL in the previous section, AP-WSJ-60COL is formulated from the collections in the UBC-100 data set. We again have two very large collections, however on this occasion the large collections comprise all documents from the original Associated Press and Wall Street Journal sources. As such, in this test data set, the large collections contain a significant share of the relevant documents. Therefore, the SBR lower bound baseline may perform fairly well on this data set.

In the following sections, we discuss the algorithm performances on this test data set using a variety of performance measures.

Rank Correlation

In this section we present the Spearman rank correlation, and Blest and Da Costa weighted rank correlation scores achieved by the algorithms on the AP-WSJ-60COL data set.

The Spearman rank correlation scores between the algorithm-produced collection rankings and both the FsBR optimal and SBR lower bound baselines are given in Table 6.14. All algorithms are significantly correlated to FsBR ($p < 0.05$), with CORI and Inner Product scoring highest overall. The strongest configuration of Doddle is Doddle_RC; although it achieves lower correlation scores than CORI and Inner Product, the differences are not statistically significant ($p > 0.05$).

We note that when comparing SBR to the optimal baseline, there is no significant correlation; this is contrary to our speculation at the start of this section, that SBR may perform well on this test data set. We reflect on this further as we consider the other performance measures.

Table 6.14 also shows algorithm correlation scores with the size-based ranking. The majority of the collection selection algorithms exhibit significant positive correlation with SBR, while Doddle_RC and the other Doddle configurations do not. This suggests there is some bias in the test data set towards the larger collections. In contrast, the SBR correlation scores of some configurations of Doddle indicate a bias against larger collections. While our collection suggestion objectives may encourage the selection of smaller collections over equally relevant larger collections, we do not wish to discriminate against the larger collections if they most appropriately address the user's query.

In Table 6.15 we provide the Blest and Da Costa weighted rank correlation scores, between the algorithm-produced rankings and the FsBR optimal ranking. These two measures emphasise the importance of correctly ranking the top collections.

In the 'Blest' column of the table, we see that all collection selection algorithms have significant correlation to FsBR, with CORI and Inner Product again scoring highest. Doddle_RC is again the best configuration of Doddle; however, some Doddle configurations are not significantly correlated to FsBR, again suggesting a bias against larger collections.

In the 'Da Costa' column however, all algorithms are significantly correlated to FsBR, and the Doddle configurations compete well with the collection selection algorithms.

On the Blest and Da Costa measures, SBR is significantly correlated to FsBR. This suggests that it may perform well within the top few rank positions, matching the optimal ranking. However, as we move down the rank positions, it becomes less effective and thus its overall performance is reduced.

In the following section we investigate the algorithms further, with the recall and precision analogous measures; these may provide additional insight into the results we have observed thus far.

Recall and Precision Analogues

In this section we consider the performances of the selected algorithms, according to the three performance measures analogous to recall and precision: $\mathcal{R}_n$, $\hat{\mathcal{R}}_n$ and $\mathcal{P}_n$. The scores

Table 6.14: The average Spearman rank correlations (over 100 test queries and the AP-WSJ-60COL test data set) for selected existing algorithms and configurations of Doddle, comparing against both the FsBR and SBR baselines. Statistically significant correlations are given in bold.

	FsBR		SBR	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.44**	**0.35**	−0.02	0.04
CORI	**0.50**	**0.53**	**0.43**	**0.32**
Inner Product	**0.48**	**0.54**	**0.42**	**0.37**
Skew	**0.41**	**0.46**	**0.46**	**0.45**
Highest-available Similarity	**0.36**	**0.38**	**0.80**	**0.81**
DFPROP	**0.39**	**0.45**	**0.48**	**0.49**
Doddle	**0.35**	**0.41**	−0.33	−0.36
Doddle_RC	**0.45**	**0.52**	0.12	0.09
Doddle_RP	**0.35**	**0.41**	−0.43	−0.46
Doddle_RC+RP	**0.42**	**0.48**	−0.19	−0.24
Doddle_RC+RF	**0.34**	**0.41**	−0.23	−0.25
Doddle_RP+RF	**0.25**	**0.33**	−0.53	−0.53
SBR	*0.13*	*0.13*	—	—

Table 6.15: The average Blest and Da Costa weighted rank correlations (over 100 test queries and the AP-WSJ-60COL test data set) for selected existing algorithms and configurations of Doddle, comparing estimate rankings to FsBR. Statistically significant correlations are given in bold.

	Blest		Da Costa	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.39**	**0.35**	**0.54**	**0.61**
CORI	**0.50**	**0.52**	**0.58**	**0.61**
Inner Product	**0.49**	**0.54**	**0.56**	**0.62**
Skew	**0.43**	**0.47**	**0.49**	**0.54**
Highest-available Similarity	**0.37**	**0.40**	**0.42**	**0.45**
DFPROP	**0.40**	**0.47**	**0.47**	**0.52**
Doddle	**0.21**	**0.24**	**0.45**	**0.50**
Doddle_RC	**0.36**	**0.39**	**0.54**	**0.60**
Doddle_RP	**0.23**	**0.24**	**0.44**	**0.50**
Doddle_RC+RP	**0.30**	**0.32**	**0.51**	**0.57**
Doddle_RC+RF	0.20	**0.24**	**0.43**	**0.49**
Doddle_RP+RF	0.11	0.16	**0.34**	**0.41**
SBR	***0.27***	***0.27***	***0.21***	***0.21***

achieved by the algorithms, within the first ten rank positions, are presented as graphs. For reference, scores at additional selected rank positions are provided in tables in Appendix D.

Figure 6.13 presents the scores achieved, according to the recall-analogous $\mathcal{R}_n$ measure, for both short and long test queries. On short queries, we see that bGlOSS scores highest at the first rank position. However, it is overtaken by CORI, which appears to be the most consistent overall. The strongest configurations of Doddle are Doddle_RC and Doddle_RC+RP, however they are a little behind CORI.

On the long test queries, bGlOSS is one of the weakest algorithms. Several algorithms perform similarly well: Inner Product, CORI, Doddle_RC and Doddle_RC+RP.

On both short and long queries, we observe that the SBR baseline scores highly within the first two rank positions, before its performance drops off. This suggests that the two largest collections are often highly relevant to the test queries (as is evidenced in Chapter 4).

The $\hat{\mathcal{R}}_n$ measure is an alternative recall-analogous measure; scores achieved by the algorithms according to $\hat{\mathcal{R}}_n$ are given as graphs in Figure 6.14. The observations on the short and long test queries are the same: CORI, Inner Product, Doddle_RC and Doddle_RC+RP all show similar performances, and tend to lead the other algorithms. As with $\mathcal{R}_n$, the SBR baseline starts strongly on $\hat{\mathcal{R}}_n$, matching the top performing algorithms. This provides further evidence that the top two collections are highly relevant to many of the test queries.

The scores achieved by the algorithms according to the precision-analogous $\mathcal{P}_n$ measure are given in Figure 6.15. Again, we observe similar trends on both sets of test queries. No algorithm achieves the optimal score of 1 at the first rank position. As such, although SBR again scores highly early on, the largest collections are not relevant to every query in this data set.

According to $\mathcal{P}_n$, CORI and Inner Product appear to be the most consistent algorithms, followed by Skew and DFPROP. The most consistent configurations of the Doddle algorithm are Doddle_RC and Doddle_RC+RP, however they do not score as highly as some of the collection selection algorithms. As such, at a given rank position, they are more likely to choose a collection that has no relevance to the user's query, according to the optimal ranking.

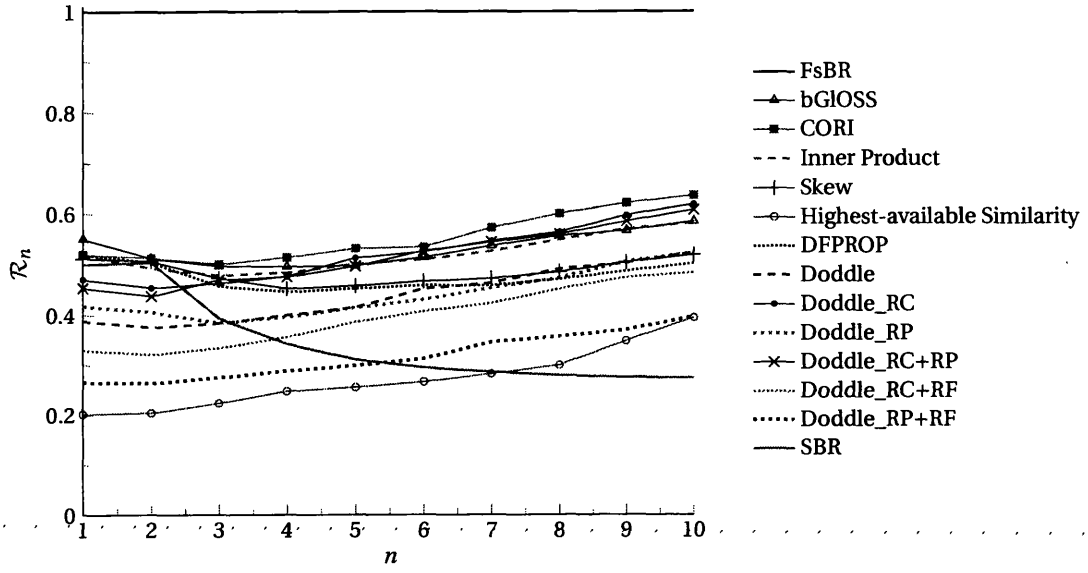
Additional discussion of the algorithm performances follows in the next section, where we look more closely at performance within the top few rank positions.

Top Rank Performance

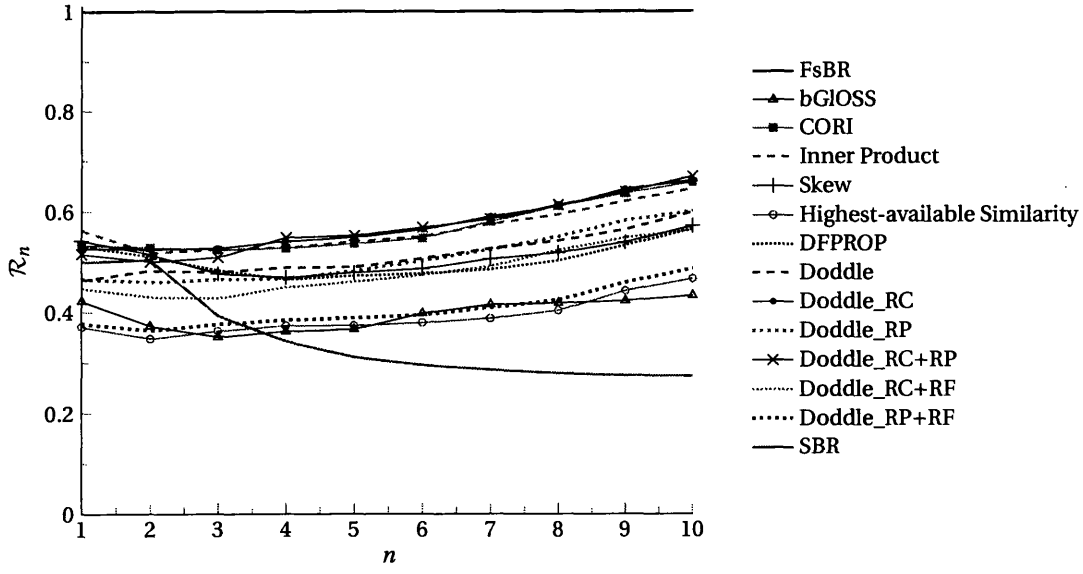
In Table 6.16 we provide the scores achieved by the algorithms according to our top rank performance measures: Precision@5 indicates how accurately an algorithm includes relevant collections within the top rank positions; while Correct@1 gives the number of queries for which an algorithm correctly identifies the top-ranked collection.

Our initial observation over these results is that the scores achieved by the algorithms are noticeably higher than those on the previous TREC-based data sets. It is possible that this data set is 'easier' than the others, due to its two large collections containing many of the documents relevant to each query.

Looking at the Precision@5 scores, we see similar trends to those on the previous performance measures discussed in this section: CORI and Inner Product are the strongest of the collection selection algorithms. On the short queries, Doddle_RC is the best configuration

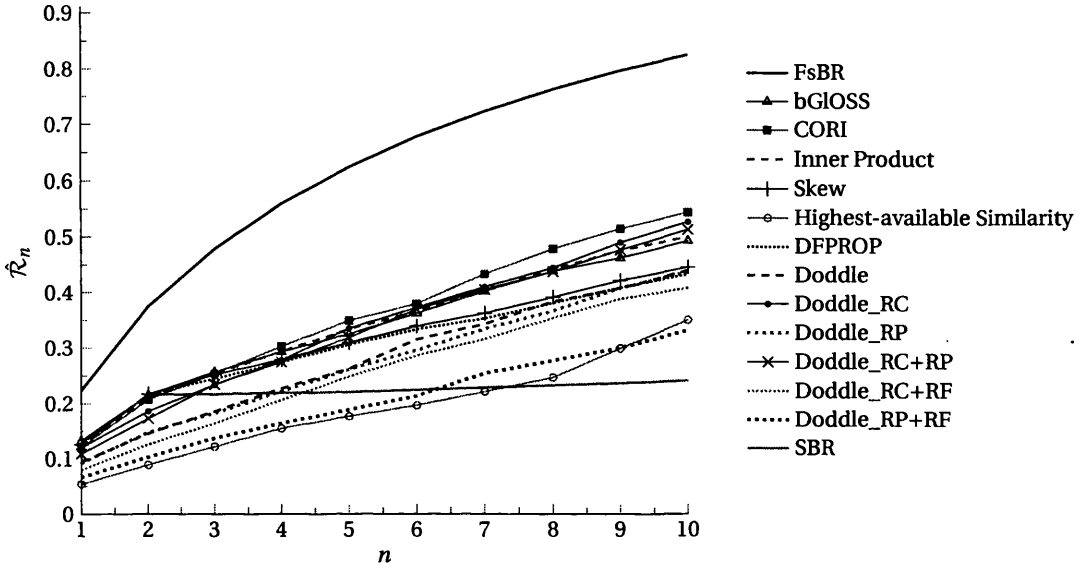


(a) Short queries.

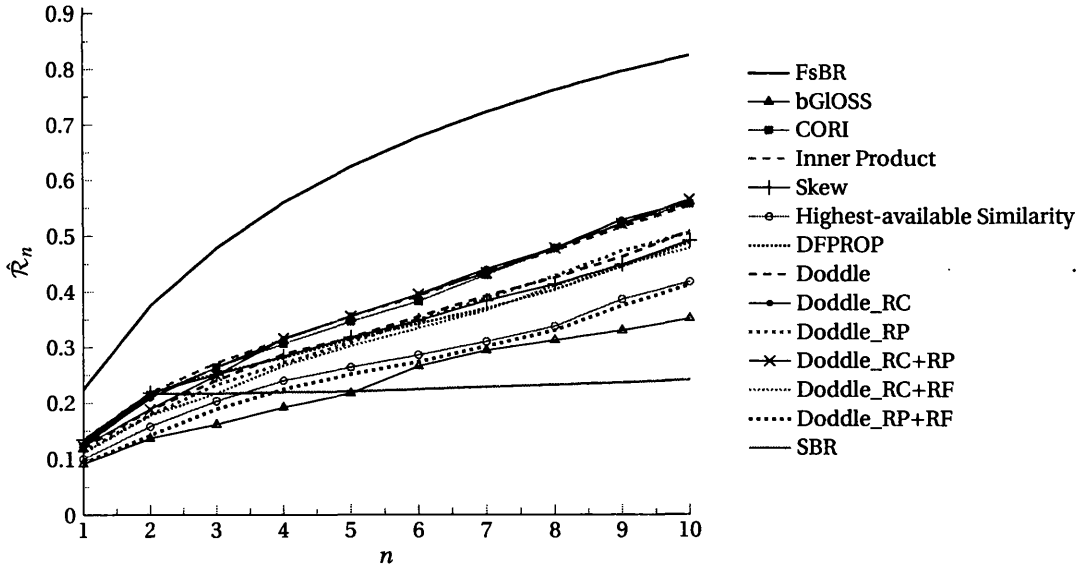


(b) Long queries.

Figure 6.13: The $\mathcal{R}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the AP-WSJ-60COL test data set.

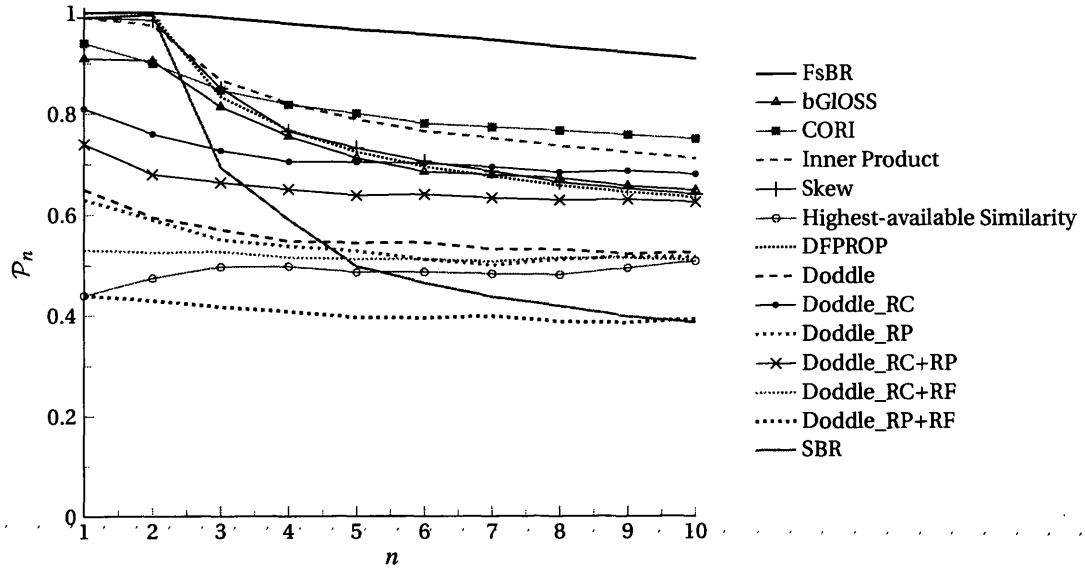


(a) Short queries.

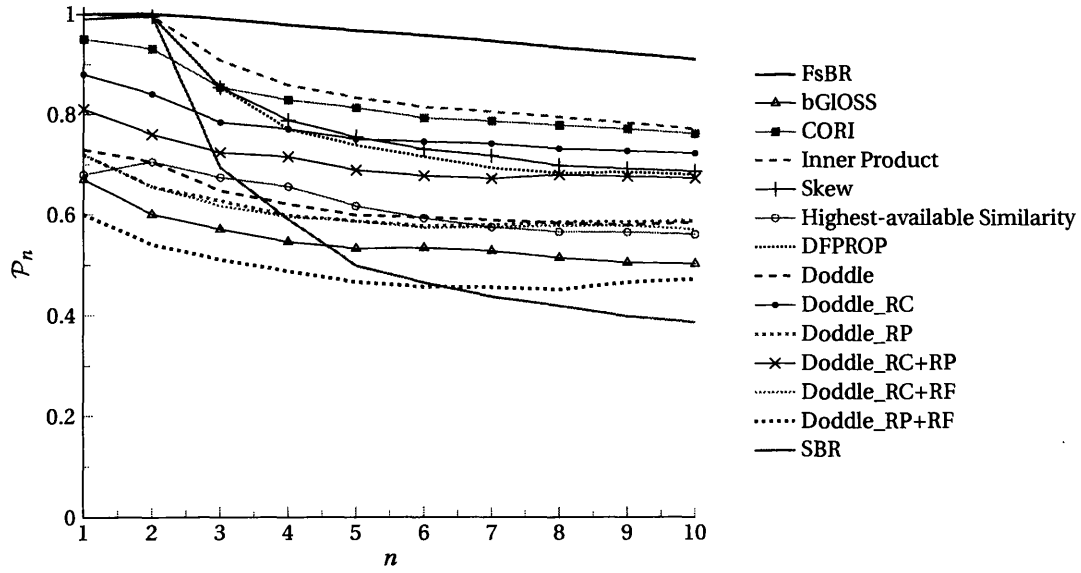


(b) Long queries.

Figure 6.14: The $\hat{\mathcal{R}}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the AP-WSJ-60COL test data set.



(a) Short queries.



(b) Long queries.

Figure 6.15: The $\mathcal{P}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the AP-WSJ-60COL test data set.

Table 6.16: The Precision@5 and Correct@1 scores achieved by selected existing algorithms and configurations of Doddle, on the AP-WSJ-60COL test data set.

	Precision@5		Correct@1			
Algorithms	Q_s	Q_l	Q_s		Q_l	
bGlOSS	0.35	0.22	25	*(25%)*	22	*(22%)*
CORI	0.36	0.36	22	*(22%)*	23	*(23%)*
Inner Product	0.34	0.36	21	*(21%)*	24	*(24%)*
Skew	0.32	0.33	20	*(20%)*	23	*(23%)*
Highest-available Similarity	0.18	0.25	9	*(9%)*	20	*(20%)*
DFPROP	0.31	0.32	19	*(19%)*	22	*(22%)*
Doddle	0.28	0.34	19	*(19%)*	25	*(25%)*
Doddle_RC	0.34	0.38	25	*(25%)*	28	*(28%)*
Doddle_RP	0.28	0.34	22	*(22%)*	23	*(23%)*
Doddle_RC+RP	0.33	0.39	26	*(26%)*	25	*(25%)*
Doddle_RC+RF	0.25	0.32	17	*(17%)*	25	*(25%)*
Doddle_RP+RF	0.21	0.26	13	*(13%)*	19	*(19%)*
SBR	*0.22*	*0.22*	*19*	*(19%)*	*19*	*(19%)*

of Doddle; however, Doddle_RC+RP narrowly scores higher than Doddle_RC on the long queries, also beating CORI and Inner Product.

On the Correct@1 measure, the algorithms are fairly evenly matched. Here, the Doddle configurations perform best, with Doddle_RC+RP and Doddle_RC the highest scorers on short and long queries respectively.

In the following section we discuss the performances of the algorithms on the final TREC-based test data set; following this we provide a discussion and summary of our findings.

6.7 FR-DOE-81COL

The final TREC-based test data set we use to evaluate the selected algorithms is FR-DOE-81COL. Formulated from the collections in the UBC-100 data set, FR-DOE-81COL contains two very large collections, comprising all the documents from the original Federal Register and Department of Energy sources.

In contrast to AP-WSJ-60COL, the large collections in this data set contain few of the relevant documents. As such the SBR baseline, and any algorithm that mimics it, is unlikely to perform well on this data set.

Using a range of performance measures, we present and discuss the performances of the algorithms on the FR-DOE-81COL data set in the following sections.

Rank Correlation

In this section we consider the performances of the algorithms in terms of three rank correlation coefficients, which measure the similarity between two rankings of objects; collections in this case. Specifically, Spearman rank correlation is a standard correlation measure. We also use two weighted rank correlation measures, which place an importance on similarity within the top rank positions.

In Table 6.17 we present the Spearman rank correlation scores between the algorithm-produced collection rankings, and both the optimal baseline, and a size-based ranking. We first consider the correlation with the optimal FsBR baseline.

As we have seen on several of the TREC-based test data sets thus far, CORI and Inner Product show the highest correlation with FSBR, with Doddle_RC the strongest of the Doddle configurations.

On both short and long queries, all algorithms have statistically significant correlation ($p < 0.05$) with FsBR, and there are no statistically significant differences between the algorithms. However, SBR is an exception: it is not significantly correlated to FsBR, and is significantly poorer than the other algorithms tested.

Comparing the similarity of algorithm-produced collection rankings and the size-based ranking indicates whether the algorithms tend to favour large collections. Table 6.17 also shows the correlation scores between the algorithms and SBR. The collection selection algorithms are significantly correlated to SBR, as is Doddle_RC (though its score is lower than

Table 6.17: The average Spearman rank correlations (over 100 test queries and the FR-DOE-81COL test data set) for selected existing algorithms and configurations of Doddle, comparing against both the FsBR and SBR baselines. Statistically significant correlations are given in bold.

	FsBR		**SBR**	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.57**	**0.45**	**0.25**	0.12
CORI	**0.63**	**0.61**	**0.31**	**0.29**
Inner Product	**0.59**	**0.63**	**0.45**	**0.43**
Skew	**0.55**	**0.57**	**0.48**	**0.47**
Highest-available Similarity	**0.51**	**0.57**	**0.42**	**0.44**
DFPROP	**0.53**	**0.56**	**0.51**	**0.51**
Doddle	**0.49**	**0.54**	0.05	0.05
Doddle_RC	**0.54**	**0.61**	**0.20**	**0.19**
Doddle_RP	**0.49**	**0.54**	0.00	−0.01
Doddle_RC+RP	**0.53**	**0.58**	0.09	0.08
Doddle_RC+RF	**0.46**	**0.53**	0.09	0.08
Doddle_RP+RF	**0.44**	**0.49**	−0.01	−0.01
SBR	*0.13*	*0.13*	—	—

Table 6.18: The average Blest and Da Costa weighted rank correlations (over 100 test queries and the FR-DOE-81COL test data set) for selected existing algorithms and configurations of Doddle, comparing estimate rankings to FsBR. Statistically significant correlations are given in bold.

	Blest		Da Costa	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.52**	**0.20**	**0.60**	**0.56**
CORI	**0.57**	**0.57**	**0.63**	**0.61**
Inner Product	**0.55**	**0.58**	**0.59**	**0.63**
Skew	**0.51**	**0.54**	**0.55**	**0.57**
Highest-available Similarity	**0.45**	**0.51**	**0.50**	**0.56**
DFPROP	**0.50**	**0.53**	**0.53**	**0.56**
Doddle	**0.47**	**0.51**	**0.50**	**0.54**
Doddle_RC	**0.51**	**0.58**	**0.55**	**0.62**
Doddle_RP	**0.47**	**0.51**	**0.50**	**0.55**
Doddle_RC+RP	**0.51**	**0.55**	**0.54**	**0.59**
Doddle_RC+RF	**0.45**	**0.50**	**0.47**	**0.53**
Doddle_RP+RF	**0.43**	**0.47**	**0.45**	**0.49**
SBR	*0.11*	*0.11*	*0.11*	*0.11*

other algorithms).

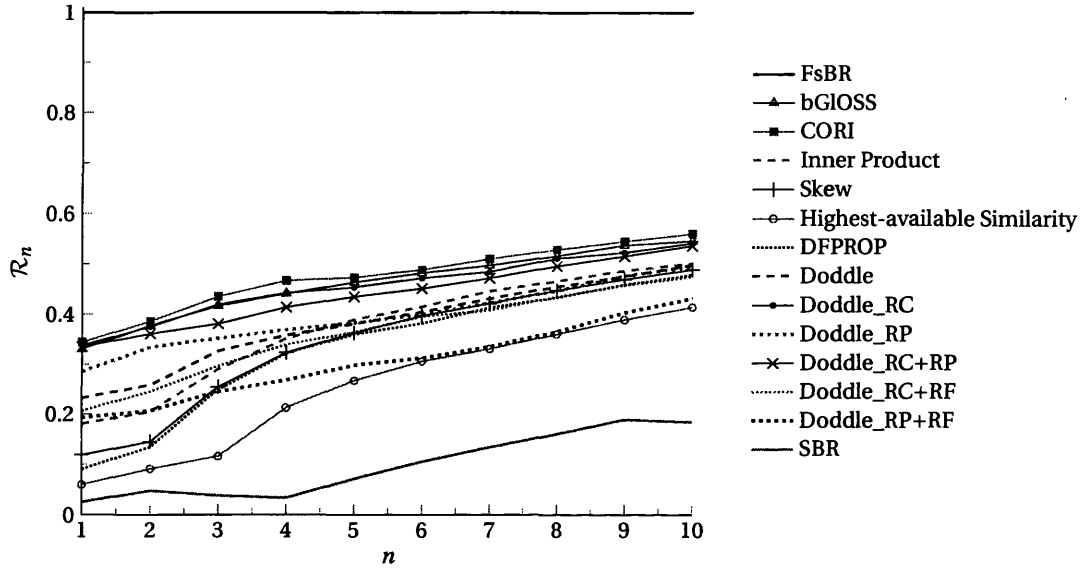
From Table 6.18, giving the Blest and Da Costa weighted rank correlation scores, we can make similar observations about similarity with FsBR, as with the Spearman correlation scores. That is, CORI and Inner Product achieve the highest correlation scores overall. The strongest configuration of Doddle is Doddle_RC, with Doddle_RC+RP also scoring highly. We note that these two algorithms are competitive, particularly on long queries.

Recall and Precision Analogues

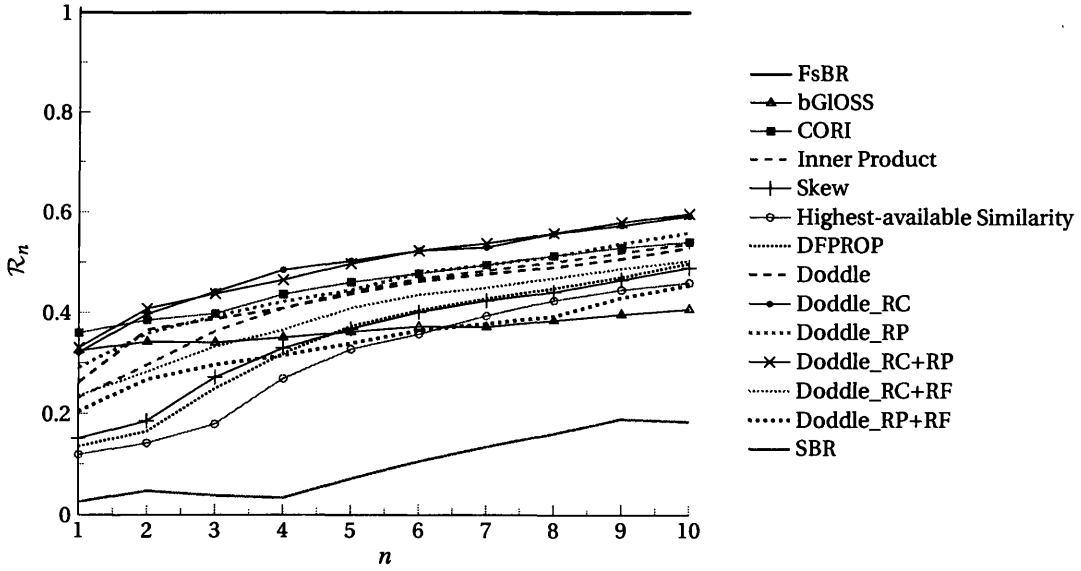
In this section we provide performance scores for the selected algorithms, on the measures analogous to recall and precision: $\mathcal{R}_n$, $\hat{\mathcal{R}}_n$ and $\mathcal{P}_n$. Scores for the first ten rank positions are given as graphs. For reference, scores achieved at additional rank positions are given in tables in Appendix D.

We first look at the $\mathcal{R}_n$ scores, given in Figure 6.16. $\mathcal{R}_n$ shows how well an algorithm selects the best collections at a given rank position. On the short test queries (Figure 6.16a), CORI is the most consistent overall. However, bGlOSS, Doddle_RC and Doddle_RC+RP also perform well, and are competitive with CORI.

On the long test queries (Figure 6.16b) the results are slightly different. Here, CORI starts with the highest score at the first rank position. However, Doddle_RC and Doddle_RC+RP score highest at the second rank position, and remain highest overall. In contrast to the short queries, bGlOSS performs poorly on long queries.

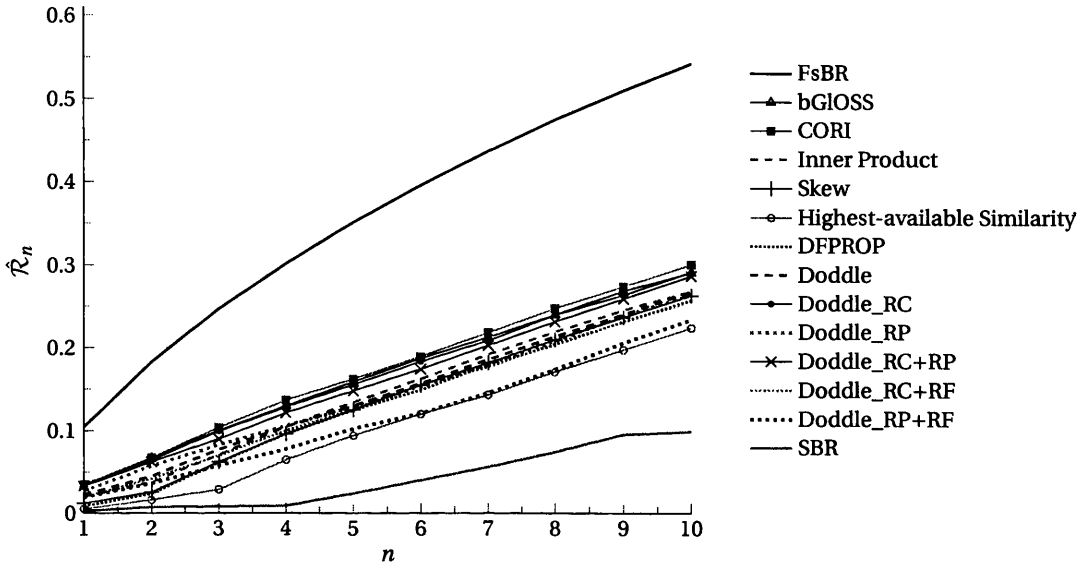


(a) Short queries.

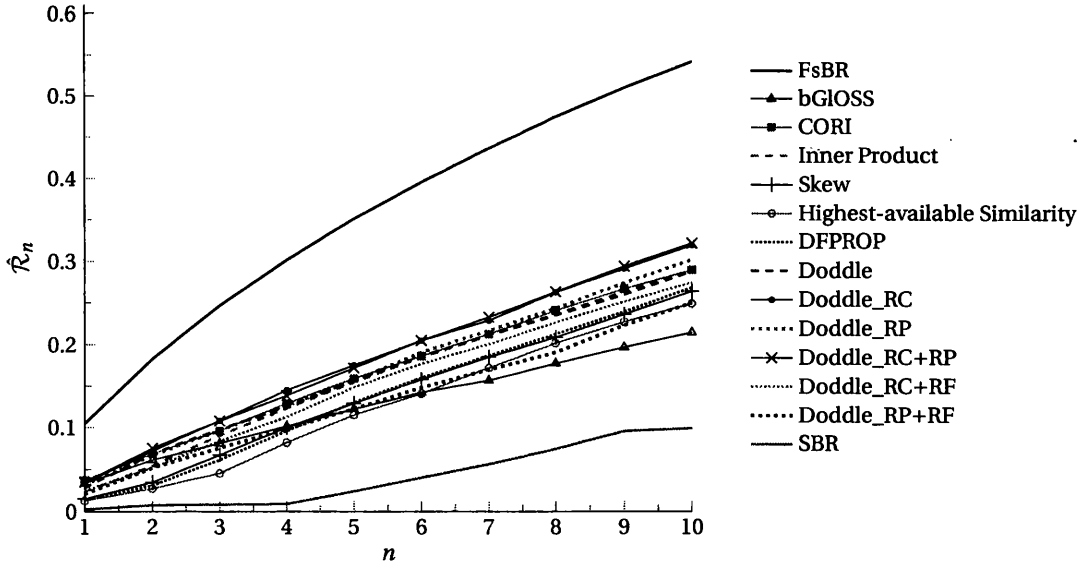


(b) Long queries.

Figure 6.16: The $\mathcal{R}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the FR-DOE-81COL test data set.

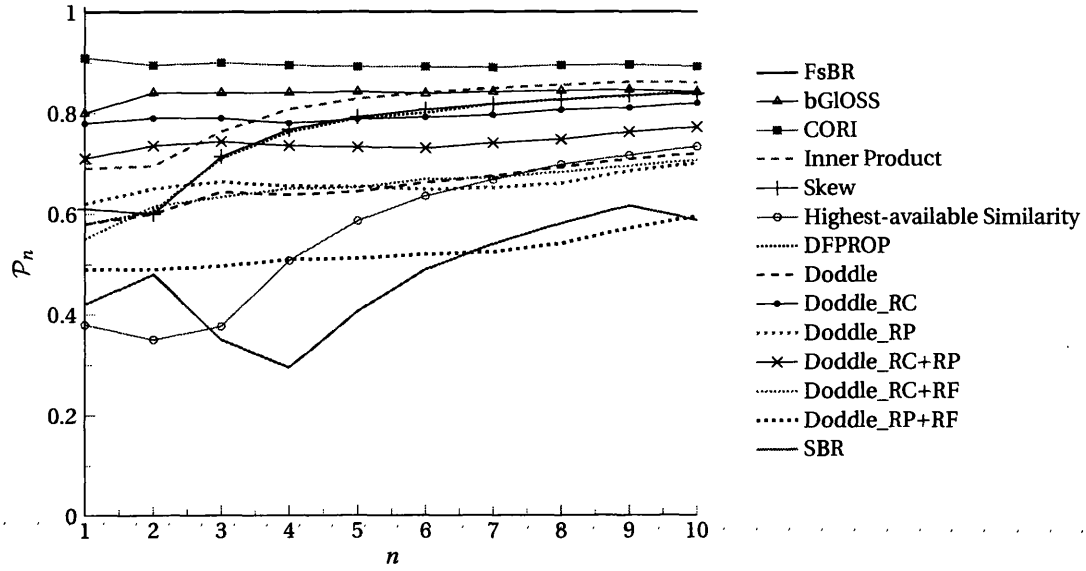


(a) Short queries.

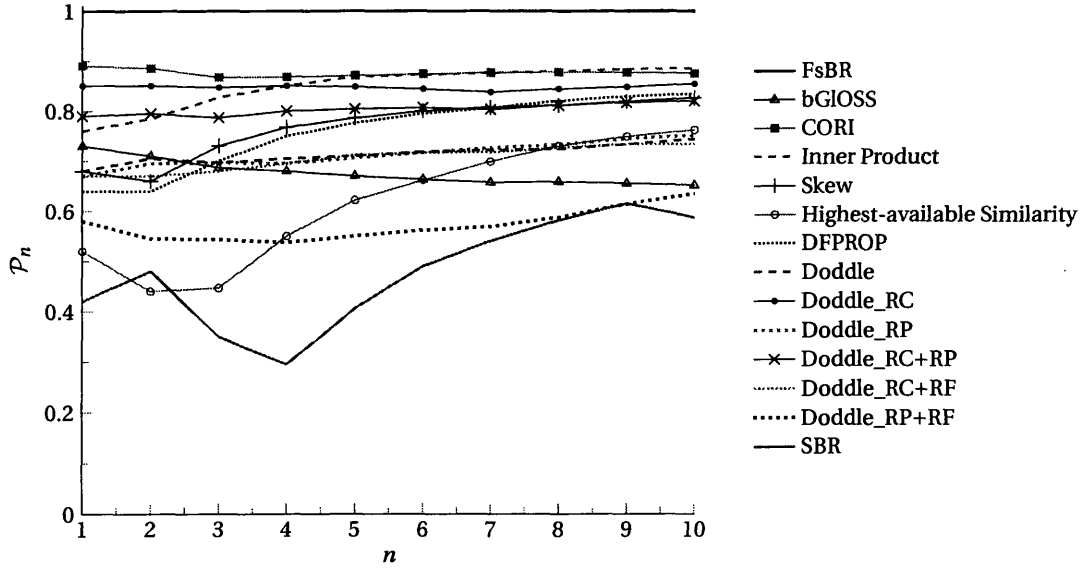


(b) Long queries.

Figure 6.17: The $\hat{\mathcal{R}}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the FR-DOE-81COL test data set.



(a) Short queries.



(b) Long queries.

Figure 6.18: The $\mathcal{P}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the FR-DOE-81COL test data set.

The $\hat{\mathcal{R}}_n$ measure is an alternative indicator of how well an algorithm chooses the best collection; the results are given as graphs in Figure 6.17. The graphs here support the findings from $\mathcal{R}_n$: on short queries CORI scores best overall, with bGlOSS, Doddle_RC and Doddle_RC+RP also competitive. On long queries, Doddle_RC and Doddle_RC+RP are the most consistent overall.

In Figure 6.18, we give the scores achieved by the algorithms on the $\mathcal{P}_n$ measure; this shows how well an algorithm chooses a collection with some relevance to the query, at a given rank position.

On the short test queries (see Figure 6.18a), CORI is clearly the strongest algorithm. However, it does not achieve the optimal score at the first rank position. As such, on some queries it ranks a collection top that the optimal ranking has deemed not to be relevant to the query. In collection suggestion, where we are directing the user to useful collections, this is undesirable.

The strongest configuration of the Doddle algorithm on short queries is Doddle_RC; however, it does lag behind CORI. On long queries, Doddle_RC is more competitive with CORI, which again appears to be the most consistent algorithm overall.

Top Rank Performance

In this section, we look more closely at the performance of the algorithms within the top few rank positions; here, it is important that the best possible collections are included, and ranked correctly.

In Table 6.19 we provide the Precision@5 and Correct@1 scores achieved by the algorithms. Recall that Precision@5 measures the proportion of collections within the top five rank positions of an algorithm-produced collection ranking, that should be there according to the optimal baseline. Correct@1 gives the number of queries for which the top ranked collection was exactly correct.

According the the Precision@5 measure, bGlOSS scores highest on short test queries, but this is very closely followed by CORI and Doddle_RC. On long test queries however, Doddle_RC and Doddle_RC+RP score highest.

The Correct@1 scores achieved by the algorithms seem generally low; this is a trend we have observed on the majority of the other TREC-based test data sets. On the short test queries in this data set, bGlOSS and Doddle_RP both score highest, predicting the top collection on 12% of the test queries. On long queries, bGlOSS, Doddle_RC, Doddle_RP and Doddle_RC+RP all exhibit the same performance, with scores of 9%.

6.8 Summary

A prerequisite for an effective search service for finding authoritative collections, is an optimal algorithm for ranking collections according to the user's query. In order to support a large-scale evaluation of potential algorithms for the task, in Chapter 3 we developed a comprehensive and rigorous evaluation methodology.

There are several aspects to our evaluation methodology. The scenario-based test component allows us to scrutinise algorithm performance on a set of specific test cases; their

Table 6.19: The Precision@5 and Correct@1 scores achieved by selected existing algorithms and configurations of Doddle, on the FR-DOE-81COL test data set.

	Precision@5		Correct@1			
Algorithms	Q_s	Q_l	Q_s		Q_l	
bGlOSS	0.25	0.18	12	*(12%)*	9	*(9%)*
CORI	0.24	0.22	10	*(10%)*	8	*(8%)*
Inner Product	0.19	0.23	8	*(8%)*	5	*(5%)*
Skew	0.17	0.19	4	*(4%)*	2	*(2%)*
Highest-available Similarity	0.13	0.18	0	*(0%)*	2	*(2%)*
DFPROP	0.16	0.20	2	*(2%)*	3	*(3%)*
Doddle	0.21	0.26	6	*(6%)*	7	*(7%)*
Doddle_RC	0.24	0.29	7	*(7%)*	9	*(9%)*
Doddle_RP	0.20	0.26	12	*(12%)*	9	*(9%)*
Doddle_RC+RP	0.23	0.29	10	*(10%)*	9	*(9%)*
Doddle_RC+RF	0.21	0.24	4	*(4%)*	6	*(6%)*
Doddle_RP+RF	0.16	0.20	6	*(6%)*	4	*(4%)*
SBR	*0.01*	*0.01*	*0*	*(0%)*	*0*	*(0%)*

simplicity means we can quickly make judgements on whether a particular algorithm is likely to be suitable for collection suggestion.

In contrast, the baseline testing component of our methodology takes a broader view of algorithm performance. We use a variety of performance measures to consider average performance, over sets of test queries. Each type of performance measure considers a different aspect of algorithm performance. The rank correlation coefficients measure the similarity between an optimal ranking of collections, and an algorithm-produced collection ranking. Within this class of performance measure, the two weighted rank correlation coefficients place an emphasis on similarity within the top rank positions. This reflects the objectives of collection suggestion, where we want to recommend the best collections to the user.

The recall and precision analogous measures respectively look at how well an algorithm chooses the best collection at a given rank position ($\mathcal{R}_n$ and $\hat{\mathcal{R}}_n$), and whether chosen collections are useful ($\mathcal{P}_n$). Note that an algorithm may do well at choosing a 'useful' collection at each rank position, but it may not select the best collection.

Perhaps the most important indicators of algorithm performance are Precision@5 and Correct@1. These focus specifically on precision within the top five rank positions, and whether the first ranked collection is correctly identified by an algorithm. As a collection suggestion search service would recommend collections for the user to visit, correctly identifying the best collection(s) is vital; correctly ordering the collections further down the ranking is somewhat irrelevant.

To ensure we test algorithms thoroughly during baseline testing, we utilise several test data sets: a range of controlled TREC-based data sets enable us to consider performance on data sets exhibiting specific attributes (different ranges of collection sizes, and distribu-

tion of relevant documents, for example). In addition, two data sets comprising data from real open access repositories, provide a test environment that is realistic of the intended operational setting.

Therefore, while previous evaluations of algorithms for ranking collections (in the related domain of collection selection) have used only one type of test data (TREC-based or realistic), we use both. This ensures that we can identify a strong and consistent algorithm, whose effectiveness will transfer to an operational environment.

In the previous chapter, we developed and tested our own collection ranking algorithm (Doddle) for collection suggestion. Using the scenario-based testing technique and baseline testing on real repository data sets, we found our algorithm to exhibit promising performance, as did several alternative configurations of the algorithm: Doddle_RC, Doddle_RP, Doddle_RC+RP, Doddle_RC+RF, and Doddle_RP+RF.

In this chapter, we have continued to work towards finding an optimal algorithm for collection suggestion. With scenario-based testing, we found that several collection selection algorithms may be suitable for collection suggestion: bGlOSS, CORI, Inner Product, Skew, Highest-available Similarity and DFPROP. Given this, the majority of this chapter evaluates these algorithms further, in addition to the most effective configurations of our Doddle algorithm, on the six TREC-based data sets. We summarise our findings as follows.

Our findings over the TREC-based data sets have been varied, with different algorithms appearing to be the most effective, depending on the data set, performance measure, and query set (short or long queries). This is not unexpected, as it is a trend that has emerged in previous evaluations in the collection selection domain (see Chapter 2).

Of the configurations of the Doddle algorithm, Doddle_RC and Doddle_RC+RP are the most competitive and consistent. These algorithms perform best when evaluated using the recall-analogous measures, $\mathcal{R}_n$ and $\hat{\mathcal{R}}_n$, which examine whether an algorithm chooses the best available collection at each rank position. For these measures, Doddle_RC and Doddle_RC+RP are the strongest algorithms on the UBC-100 data set, and its derivatives (2LDB-60COL, AP-WSJ-60COL and FR-DOE-81COL). These two algorithms are also often amongst the strongest algorithms according to the Precision@5 and Correct@1 measures.

However, a notable weakness of the Doddle configurations is their performance according to the $\mathcal{P}_n$ measure (showing whether an algorithm chooses a useful collection at a given rank position). Even the Doddle_RC and Doddle_RC+RP algorithms tend to lag behind several collection selection algorithms, and are prone to choosing collections that the optimal ranking has determined to have no relevance to a query.

The weakest configurations of Doddle tend to be Doddle_RC+RF and Doddle_RP+RF: those featuring the 'relative frequency' component that we found to be ineffective in our experiments in Chapter 5. As such, it is becoming evident that this component is detrimental to algorithm performance.

Of the selection of collection selection algorithms tested, CORI has been the most consistent throughout. While it frequently scores highest, across the various performance measures (particularly on the correlation coefficients), it is not always best. For example, Inner Product (which is also generally strong and consistent), Doddle_RC and Doddle_RC+RP have scored better on multiple occasions.

The Highest-available Similarity algorithm was strong across all performance measures on the SYM-236 and UDC-236 test data sets. However, it is one of the weakest algorithms on the other test data sets (all of which are derived from UBC-100).

The bGlOSS algorithm also has variable performance. Although it is often competitive on the set of short queries, it performs poorly on the set of long queries. bGlOSS aims to estimate the number of documents in each collection that contain all query terms. The longer the query, the less likely it is for documents to match all query terms; this explains its reduced effectiveness on long queries. However, the performance drop of bGlOSS on the long test queries is in contrast to the other algorithms, which tend to achieve slightly higher performance scores on long queries.

At this stage of our evaluation of algorithms, with respect to collection suggestion, CORI, Inner Product, Doddle_RC and Doddle_RC+RP are the strongest and most consistent algorithms. However, the performance scores achieved by all algorithms are generally lower than we might expect (and some distance from optimal performance), for them to effectively recommend suitable collections to the user. However, the performances shown by the collection selection algorithms are generally on par with those found in original evaluations of those algorithms. There is particular room for improvement on the Precision@5 and Correct@1 measures; correctly identifying the best few collections is especially important in collection suggestion.

In the following chapter we continue our large-scale evaluation of algorithms, for collection suggestion. We use a large data set built from open access repository data (see Chapter 4). As such, we conduct tests on an environment representative of an operational setting: realistic data, and collections of a range of sizes, some of varied subject matter, and others specialising on a specific topic.

Chapter 7

Evaluation of Algorithms: Open Access Repository Data

In the previous chapter we began our large-scale evaluation of the suitability and performance of a selection of algorithms (various configurations of our own algorithm, and several collection selection algorithms), for the collection suggestion task. We tested the algorithms in controlled settings (on data exhibiting particular attributes); first with scenario-based testing, followed by baseline testing using a set of six TREC-based data sets. These gave an insight into strengths and weakness of the algorithms. From these tests, we found four algorithms to be strong and consistent: CORI, Inner Product, Doddle_RC and Doddle_RC+RP.

In this chapter, we continue our evaluation of algorithms for collection suggestion. Here, we use a large open access repository test data set (presented in Chapter 4), which represents a realistic operational environment. With this, we can examine whether algorithm performances transfer to an operational setting. The combination of scenario-based testing, and baseline testing on both TREC-based and open access repository (realistic) data sets, provides a rigorous and thorough test environment for algorithms.

This chapter is organised as follows: in Section 7.1 we provide an overview of the experimental environment used here, with the results over this environment reported in Section 7.2. We discuss and summarise our findings in Section 7.3.

7.1 Overview

For this portion of our evaluation of algorithms for collection suggestion, we utilise our large-scale open access repository data set: RTD (Refined Test Data). As discussed previously in Chapter 4, the RTD set includes metadata (titles and descriptions) documents from 100 real digital repositories. As such, this data is realistic of the intended operational environment.

On this data set, we utilise two term indexes: one containing data from the Title metadata field alone, and one containing data from both the Title and Description metadata fields of documents. As mentioned in Chapter 4, this allows us to identify which metadata

is most useful for collection suggestion, by observing which produces the highest algorithm performance scores. Therefore, in this chapter we present results from both term indexes. For our experiments in this chapter, we use the large set of 200 queries (listed in Appendix B).

In the following section we present the performance results of a selection of algorithms from the collection selection domain, and several configurations of the Doddle algorithm (developed and tested in Chapter 5). Specifically, we test the following algorithms: bGlOSS, CORI, Inner Product, Skew, Highest-available Similarity, DFPROP, Doddle, Doddle_RC, Doddle_RP, Doddle_RC+RP, Doddle_RC+RF and Doddle_RP+RF. These are the same selection of algorithms as those tested in Sections 6.2 to 6.7 of the previous chapter. However, for reference, performance scores of all implemented algorithms (all Doddle configurations and the collection selection and query performance prediction algorithms used in the scenario-based testing in Chapter 6), over this test data set, are given in Appendix D.

7.2 RTD Results

In this section we present the performance scores achieved by the tested algorithms, on the RTD set. A variety of performance measures are used, including: rank correlation, recall and precision analogues, and measures focussing on performance within the top few rank positions. We discuss the performances with respect to each class of measure, in turn.

Rank Correlation

We first consider the performances of the algorithms under test, in terms of three rank correlation measures: Spearman, Blest and Da Costa. These measure the similarity between an algorithm-produced collection ordering, and a baseline (optimal or lower bound) collection ordering. While Spearman is a standard correlation measure, Blest and Da Costa are weighted rank correlation measures. That is, they place an emphasis on similarity within the top rank positions. Correctly identifying and ordering the top collections is particularly important for collection suggestion: recommending sub-par collections to the user will waste their time.

We first consider the algorithm performances according to Spearman; scores achieved by the algorithms are given in Table 7.1. The 'FsBR' column shows the similarity between the optimal FsBR baseline, while the 'SBR' column indicates algorithm similarity to a size-based ranking.

All algorithms tested here show a statistically significant ($p < 0.05$) correlation to the optimal baseline; including SBR. Indeed, there is little to choose between the majority of the algorithms. The exceptions are bGlOSS and SBR, which score significantly lower ($p < 0.05$) than the other algorithms, on both the Titles and Titles and Descriptions term indexes. The highest scoring algorithms are CORI and Doddle_RC. On this data set, there is significant difference between correlations scores achieved on the two term indexes.

In addition to being correlated to the optimal ranking, the majority of the algorithms are also significantly correlated to a size-based ranking. Since the algorithms perform well in relation to the optimal, this suggests there is some tendency within the data set for the

Table 7.1: The average Spearman rank correlations (over 200 test queries and the RTD test data set) for selected existing algorithms and configurations of Doddle, comparing against both the FsBR and SBR baselines. Statistically significant correlations are given in bold.

	FsBR		**SBR**	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
bGlOSS	**0.19**	**0.25**	0.14	**0.20**
CORI	**0.93**	**0.84**	**0.50**	**0.54**
Inner Product	**0.90**	**0.81**	**0.57**	**0.62**
Skew	**0.90**	**0.81**	**0.57**	**0.62**
Highest-available Similarity	**0.90**	**0.83**	**0.57**	**0.65**
DFPROP	**0.90**	**0.81**	**0.57**	**0.62**
Doddle	**0.92**	**0.76**	**0.40**	**0.35**
Doddle_RC	**0.93**	**0.80**	**0.39**	**0.38**
Doddle_RP	**0.92**	**0.73**	**0.39**	**0.29**
Doddle_RC+RP	**0.93**	**0.77**	**0.39**	**0.33**
Doddle_RC+RF	**0.92**	**0.78**	**0.41**	**0.41**
Doddle_RP+RF	**0.92**	**0.73**	**0.41**	**0.34**
SBR	***0.42***	***0.45***	—	—

Table 7.2: The average Blest and Da Costa weighted rank correlations (over 200 test queries and the RTD test data set) for selected existing algorithms and configurations of Doddle, comparing estimate rankings to FsBR. Statistically significant correlations are given in bold.

	Blest		**Da Costa**	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
bGlOSS	**0.17**	0.10	**0.66**	**0.61**
CORI	**0.91**	**0.80**	**0.92**	**0.84**
Inner Product	**0.88**	**0.77**	**0.88**	**0.81**
Skew	**0.88**	**0.77**	**0.88**	**0.81**
Highest-available Similarity	**0.87**	**0.78**	**0.88**	**0.82**
DFPROP	**0.88**	**0.77**	**0.88**	**0.80**
Doddle	**0.92**	**0.75**	**0.92**	**0.76**
Doddle_RC	**0.92**	**0.79**	**0.92**	**0.80**
Doddle_RP	**0.92**	**0.72**	**0.92**	**0.73**
Doddle_RC+RP	**0.92**	**0.76**	**0.92**	**0.77**
Doddle_RC+RF	**0.91**	**0.77**	**0.92**	**0.78**
Doddle_RP+RF	**0.91**	**0.71**	**0.91**	**0.72**
SBR	*0.02*	*0.14*	***0.49***	***0.49***

larger collections to contain larger shares of the relevant documents, or have some relevance to more queries.

Table 7.2 gives the Blest and Da Costa weighted rank correlation scores between the algorithm-produced collection rankings and the optimal rankings. The findings here are similar to those from Spearman. All algorithms are significantly correlated to FsBR, and the scores are generally close. The strongest algorithms are Doddle_RC and CORI; however, other configurations of Doddle, such as Doddle_RP, Doddle_RC+RP, and the original formulation of the algorithm, also do well.

We notice that correlation scores are much higher on this data set than those observed on the TREC test data sets, in Chapter 6. This may be because we are utilising real repositories, with many dealing with only a few topics. As such, it may be clearer which collections best match each query, than in the TREC data sets, where the collections may appear very similar, and deal with many topics. Alternatively, this may be an artefact of our technique for generating relevance judgements: as this is an automated approach, the techniques for choosing relevant documents use the same statistics as those used to rank collections. There is no human input or additional information, as is the case with the TREC data sets.

In the following section we consider the performances of the algorithms in terms of the recall and precision analogues. These consider how well the algorithms choose the best collections, and collections that have some value to the query.

Recall and Precision Analogues

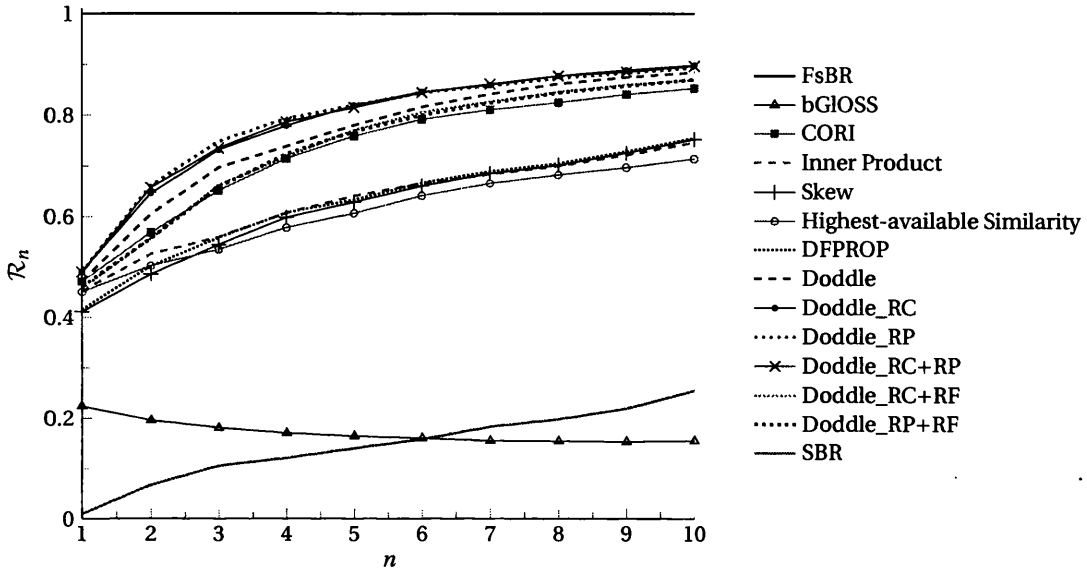
In this section, we consider the performances of the tested algorithms, with respect to the recall and precision analogous measures: $\mathcal{R}_n$, $\hat{\mathcal{R}}_n$ and $\mathcal{P}_n$. The scores achieved by the algorithms at each rank position, up to the tenth rank position, are given as graphs. For reference, scores achieved at additional selected rank positions are given in tables in Appendix D.

The $\mathcal{R}_n$ measure indicates how well the algorithms select the best collections, at a given rank position. Figure 7.1 shows the algorithm scores for this measure. Here, three configurations of the Doddle algorithm stand out as being superior over the collection selection algorithms: Doddle_RC, Doddle_RP and Doddle_RC+RP. As with the rank correlation scores in the previous section, the $\mathcal{R}_n$ scores tend to be higher when queries are executed over the Titles term index, than on the Titles and Descriptions term index.

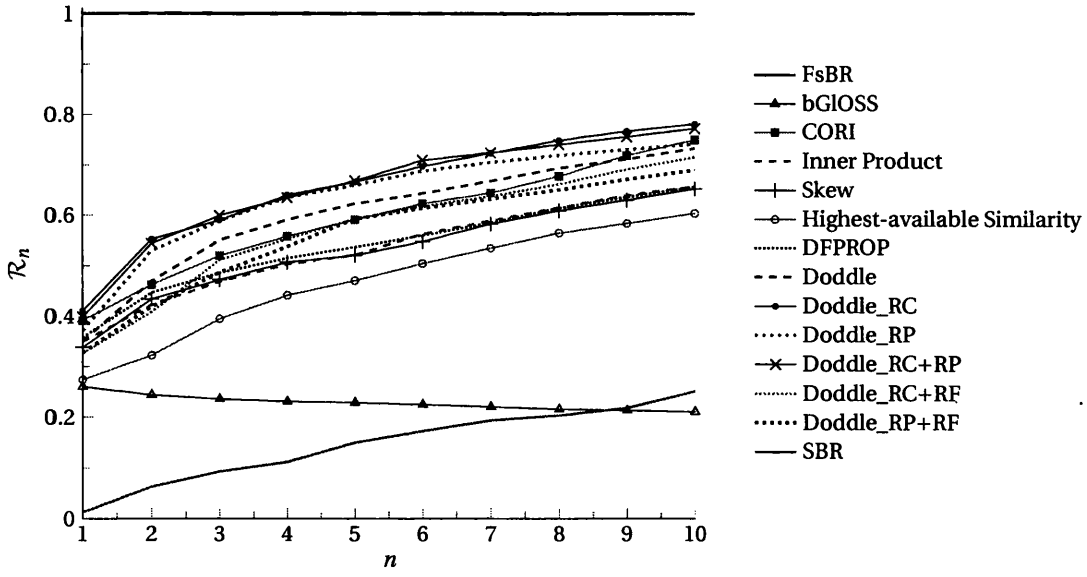
The $\hat{\mathcal{R}}_n$ measure shows the merit associated with collections gathered up to each rank position. As such, it is an alternative measure of how well the algorithms select the best collections. Figure 7.2 gives the scores achieved by the algorithms, according to this measure. The observations here are similar to those from $\mathcal{R}_n$. That is, Doddle_RC, Doddle_RP and Doddle_RC+RP tend to be the strongest algorithms overall. Again, the Titles term index yields the highest scores.

Finally, the $\mathcal{P}_n$ measure shows how well the algorithms select collections that have some relevance to the query (at a given rank position, while an algorithm may choose a collection that is relevant, it may not be the best collection available). The scores achieved according to $\mathcal{P}_n$ are given in Figure 7.3.

Here, the collection selection algorithms tend to perform better than the Doddle configurations; in particular, Highest-available Similarity, CORI and Inner Product appear to

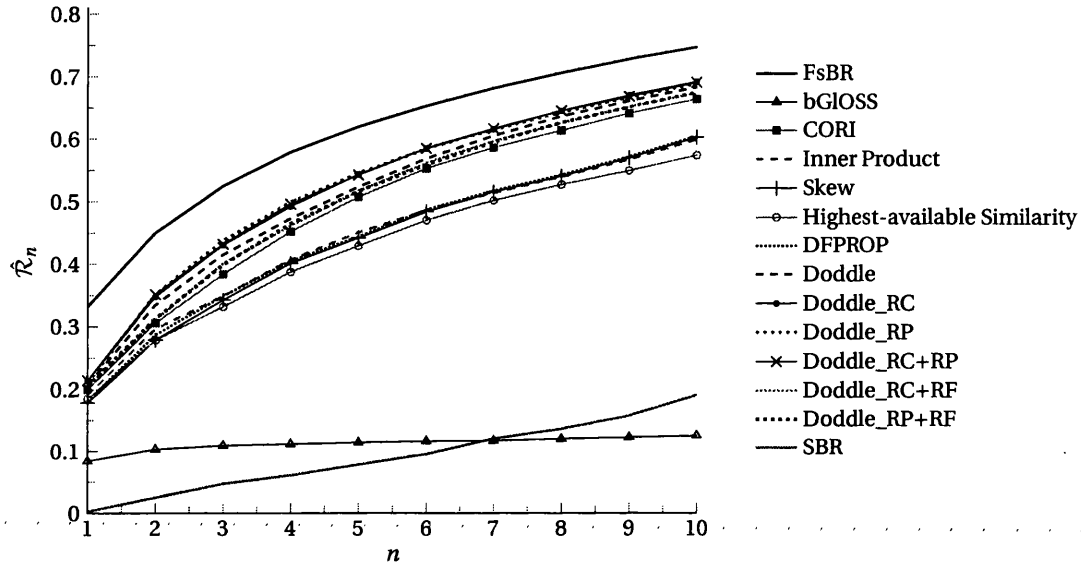


(a) Titles.

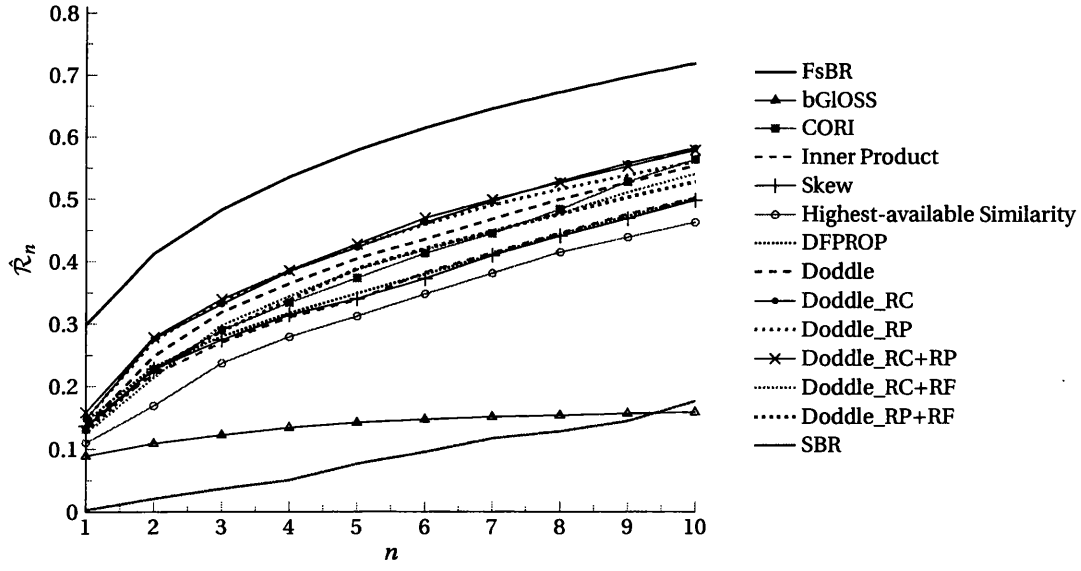


(b) Titles and Descriptions.

Figure 7.1: The $\mathcal{R}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the RTD test data set.

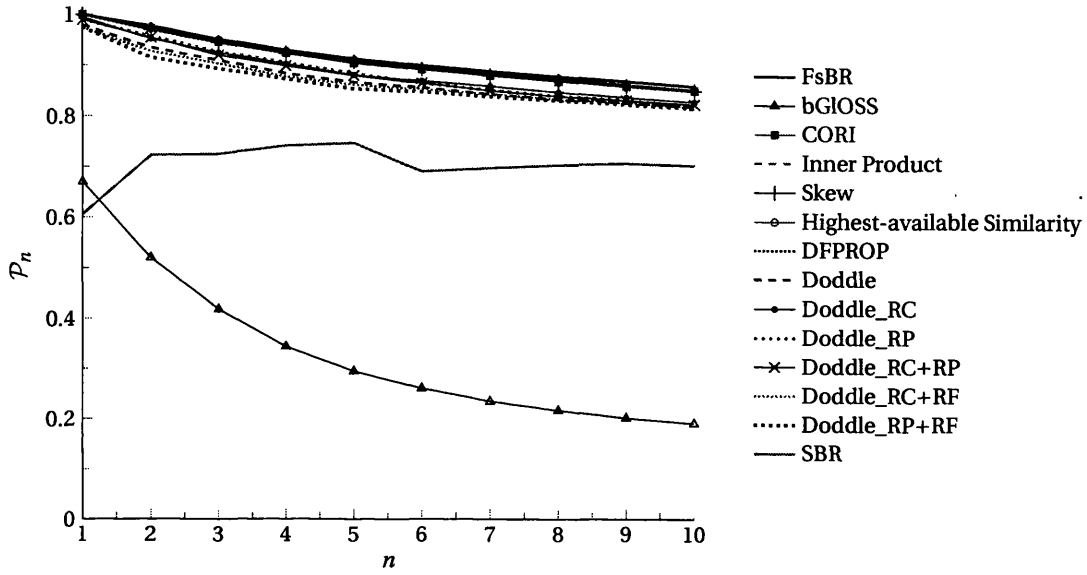


(a) Titles.

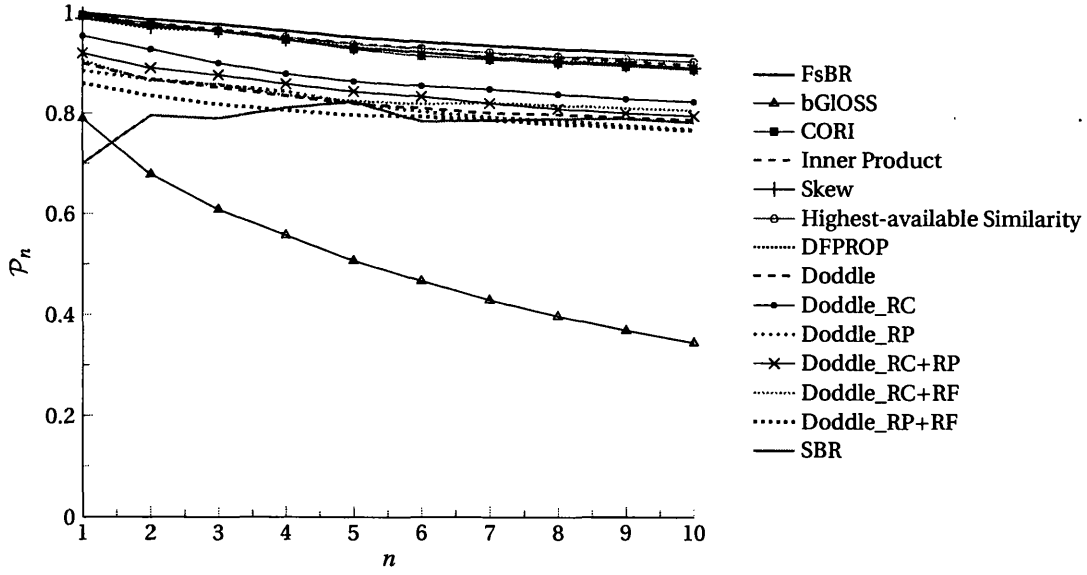


(b) Titles and Descriptions.

Figure 7.2: The $\hat{\mathcal{R}}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the RTD test data set.



(a) Titles.



(b) Titles and Descriptions.

Figure 7.3: The $\mathcal{P}_n$ values (up to $n = 10$) achieved by selected existing algorithms and configurations of Doddle, on the RTD test data set.

perform best overall. While algorithm performances are fairly close, when using the Titles and Descriptions term index, there are more pronounced differences between the collection selection algorithms and the configurations of Doddle. The strongest configuration of Doddle overall, on the $\mathcal{P}_n$ measure, is Doddle_RC.

Over all three of these performance measures, bGlOSS again performs very poorly, frequently scoring worse than the lower bound baseline, SBR.

In the following section we focus more closely on the performance of the algorithms within the top few rank positions.

Top Rank Performance

For the collection suggestion task, the goal is to recommend relevant collections to the user, so they may search and browse them themselves, and return to satisfy future related information needs. As such, correctly ranking the top few collections is of primary importance.

To evaluate algorithm performance within the top rank positions we utilise two measures: Precision@5 and Correct@1. As discussed in Chapter 3, Precision@5 shows the proportion of collections within the top five rank positions that should be present, according to the optimal ranking. Correct@1 gives the number and percentage of queries for which an algorithm correctly identifies the top ranked collection. Table 7.3 gives the scores achieved by the algorithms for these measures.

We first consider the Precision@5 scores; our first observation is that the Precision@5 scores achieved by the algorithms on this test data set are much higher than those on the TREC-based test data sets (see Chapter 6). The highest scoring algorithm overall is Dod-

Table 7.3: The Precision@5 and Correct@1 scores achieved by selected existing algorithms and configurations of Doddle, on the RTD test data set.

	Precision@5		**Correct@1**			
Algorithms	Titles	Titles & Desc.	Titles		Titles & Desc.	
bGlOSS	0.11	0.15	17	*(8.5%)*	19	*(9.5%)*
CORI	0.58	0.42	53	*(26.5%)*	39	*(19.5%)*
Inner Product	0.45	0.34	52	*(26.0%)*	34	*(17.0%)*
Skew	0.43	0.33	44	*(22.0%)*	35	*(17.5%)*
Highest-available Similarity	0.44	0.32	48	*(24.0%)*	27	*(13.5%)*
DFPROP	0.44	0.34	45	*(22.5%)*	40	*(20.0%)*
Doddle	0.61	0.43	59	*(29.5%)*	38	*(19.0%)*
Doddle_RC	0.64	0.48	61	*(30.5%)*	47	*(23.5%)*
Doddle_RP	0.66	0.49	68	*(34.0%)*	44	*(22.0%)*
Doddle_RC+RP	0.65	0.48	64	*(32.0%)*	47	*(23.5%)*
Doddle_RC+RF	0.59	0.41	55	*(27.5%)*	34	*(17.0%)*
Doddle_RP+RF	0.60	0.41	62	*(31.0%)*	35	*(17.5%)*
SBR	*0.07*	*0.07*	0	*(0.0%)*	*1*	*(0.5%)*

dle_RP; however all configurations of the Doddle algorithm tested here achieve scores that are superior to those of the collection selection algorithms.

Turning to the Correct@1 scores, we see that the configurations of Doddle again tend to score better than the collection selection algorithms: the Doddle_RC, Doddle_RP and Doddle_RC+RP performing best overall. We again note that these scores tend to be higher than those achieved over several of the TREC-based data sets; however there is still room for improvement.

Finally, we note that SBR scores poorly according to these two metrics. In particular, the highest Correct@1 score it achieves is using the Titles and Descriptions term index, correctly ranking the top collection on only one query. This suggests that on this test data set, the largest collection is rarely the most relevant.

7.3 Summary

In this chapter we have continued our work to identify an optimal algorithm to rank collections in a collection suggestion search service. While in the previous chapter we looked at algorithm performance in controlled settings (where different attributes of the data were varied), in this chapter we tested algorithms on a large-scale data set comprising metadata from real open access repositories. This data set is intended to be representative of the collection suggestion operational environment, and as such provides a real indication of algorithm performance.

Algorithm performances have varied on this data set, depending on which performance measure we are looking at. However, some trends have emerged.

Here, all configurations of the Doddle algorithm have shown strong and consistent performance across the performance measures. There is little difference between this group of algorithms, however Doddle_RC, Doddle_RP and Doddle_RC+RP often have the highest scores of all the algorithms tested. This shows that algorithms utilising simple calculations can be at least as, if not more so, effective for collection suggestion as more complex algorithms.

As we saw in the previous chapter with results on the TREC-based data sets, the Doddle algorithms do not perform as well according to the $\mathcal{P}_n$ metric as they do on the other measures. However, on this data set they do remain competitive.

The performances of Zobel's lexicon inspection algorithms (Inner Product, Skew and Highest-available Similarity) are mixed. On the rank correlation measures the three algorithms are amongst the strongest algorithms, on the Title and Description metadata. They also perform well according to the $\mathcal{P}_n$ metric. However, on $\mathcal{R}_n$, $\hat{\mathcal{R}}_n$ and the top rank performance measures, these algorithms are off the pace of the Doddle algorithms. Of these, Highest-available Similarity tends to be the weakest, as we saw in Chapter 6.

Of the other algorithms, CORI is again strong and consistent (as we saw in Chapter 6), however it is often beaten by the strongest formulations of Doddle (on $\mathcal{R}_n$, $\hat{\mathcal{R}}_n$, Precision@5 and Correct@1). DFPROP is consistent but unexceptional, and bGlOSS consistently performs very poorly on this data set. Indeed, it has fallen below the lower bound baseline performance of SBR on some measures (Spearman and $\mathcal{P}_n$).

In this chapter we have tested algorithms using two separate term indexes; one containing terms from the Title metadata field alone, and the other containing terms from both the Title and Description fields. This enables us to determine optimal metadata to represent collections.

On this data set we have found that most algorithms (with bGlOSS being the only exception) achieved significantly higher performance scores on the Titles term index. We speculate that the Title metadata provides keywords that precisely describe the contents of a collection, whereas the Description field contains some noise.

A final observation from this part of our evaluation of algorithms, is that the scores from the various performance measures are higher on the RTD set than those achieved on the TREC-based data sets. This may be a result of the presence of real collections, rather than artificial ones: individual collections in the TREC data sets are diverse in content, while the real repositories may have more defined coverage of a few specific topics. As such, choosing suitable collections for a query may be easier. However, this does not diminish the value of the RTD set: testing on realistic data is essential to ensure performance will transfer to a real operational setting.

In the following chapter we give a summary and discussion of our empirical findings from the last three chapters. In addition, we recapitulate the contributions of our work.

Chapter 8

Conclusions

Subject specialists and researchers often face the problem of identifying domain-specific authoritative collections: those directly about their topic of interest, to which they regularly return to satisfy related information needs, or monitor for new material. Discovery of such collections is often incidental, or relies on recommendations from domain experts. Existing search services, such as general purpose search engines and repository directories, offer poor support for this collection finding task.

Therefore, there is a clear need for a search service to assist users in finding authoritative collections, that can serve both their current and future information needs. However, there are several prerequisites for developing an effective search service of this kind; it is these prerequisites that we have addressed in this thesis, and summarise as follows.

At the heart of a search service for collections, we require an effective algorithm for ranking collections with respect to the user's query. Additional tasks and challenges then follow from this requirement. For example, to objectively evaluate the effectiveness (how accurately they rank collections) of potential algorithms for the task, we require appropriate performance measures and test data. In addition, it follows that we need tools to facilitate the management of test data, and execution of algorithm evaluation experiments.

In the remainder of this chapter, we first summarise our contributions with respect to the prerequisites for a search service for collections, and their associated challenges (Section 8.1). Following this, we discuss our empirical findings regarding suitable collection ranking algorithms (Section 8.2), and consider the limitations of our work (Section 8.3). Finally, we reflect on possibilities for future work in this domain (Section 8.4).

8.1 Contributions

As mentioned in the introduction of this chapter, there are several preconditions to developing a search service for authoritative collections. The contributions of our work address these preconditions; the eventual development of an effective collection search service requires fundamental research.

An effective search service for collections needs an optimal algorithm for ranking collections with respect to a user's query; other challenges and tasks emerge from this requirement. As discussed in Chapter 1 of this thesis, there is a large body of research relating to ranking resource collections according to their relevance to a user's query: namely, the domain of collection selection.

Collection selection is traditionally considered to be a sub-problem of federated search;[1] as such, it is a precursor to document retrieval. Collection selection algorithms aim to rank collections by their likelihood of containing relevant documents.

However, the focus of a search service specifically to recommend authoritative collections to a user, differs from collection selection in the context of federated search. Here, while it is important for a 'relevant' collection to contain a large number of documents, for it to be an authority we consider that these relevant documents should constitute a large proportion of the collection as a whole.

As such, in our work we essentially reformulate the collection selection problem: treating it as an independent search task, with the goal of recommending authoritative collections to the user. We refer to this task as *collection suggestion.*

Towards identifying an optimal algorithm for collection suggestion, a logical starting point is to examine whether existing algorithms can be effectively applied to the task. As such, a significant contribution of this work is a large scale evaluation of the suitability and performance of collection selection algorithms, when applied to collection suggestion.

In addition, we also consider the suitability of another class of algorithm: query performance predictors. These algorithms were originally intended to estimate the quality of search results at a single search engine, such that action (for example, query term expansion) can be taken for queries predicted to perform poorly.

Given that we want to evaluate algorithms with respect to collection suggestion, we require a rigorous evaluation methodology that meets the expectations of information retrieval research. Again, a logical starting point is to draw from existing work where possible. As such, we use established evaluation techniques, from the related domain of collection selection, as a basis for our evaluation methodology. However, we amend and extend these techniques where necessary, to develop a methodology that represents the specific objectives of collection suggestion.

Therefore, as in Chapter 1, we specify our contributions towards collection suggestion as follows:

- Evaluate the effectiveness of collection selection and query performance prediction algorithms, for collection suggestion;
- Specify and evaluate a new algorithm for collection suggestion; and
- Develop a robust methodology, apparatus and test data for evaluating algorithms for collection suggestion.

[1] Recall, in federated search, the user's query is forwarded to several resources, and the returned results are merged into one results list.

We have addressed these contributions and their respective challenges in detail throughout this thesis. We summarise our work in the remainder of this section: we focus first on our methodological process, prior to discussing algorithms and performance findings.

Evaluation Methodology

As mentioned previously, the collection suggestion task requires a suitable algorithm for ranking collections according to the user's query. However, how do we assess what is the best algorithm for the job? We require a methodology to evaluate the effectiveness (how accurately collections are ranked) of potential algorithms. We do not undertake a subjective, user-based evaluation at this stage, as we first require an effective and mature system before we can evaluate effect on user performance.

Our evaluation strategy for collection suggestion is based on traditional information retrieval evaluations of the Cranfield format (see Section 2.2.2 of Chapter 2). Such evaluations are replicable and, ideally, test algorithms in a realistic environment.

Several of the techniques we have employed to evaluate algorithms for collection suggestion are drawn from those used in collection selection evaluations. However, we make reasoned adjustments and additions where appropriate, to suit the requirements of the collection suggestion task.

As we discussed in Chapter 2, collection selection evaluations are traditionally based around testing against baseline rankings. Here, collection rankings produced by an algorithm are compared to baseline collection rankings. Two baselines commonly used are Relevance-Based Ranking (RBR), where collections are ranked in descending order of the number of relevant documents they actually contain; and Size-Based Ranking (SBR), where collections are ranked in descending order of the number of documents they contain. RBR is considered an optimal baseline, whereas SBR is a lower bound baseline.

For our collection suggestion evaluation, we follow the baseline approach; however, we specify an alternative optimal baseline, that reflects the specific goals of collection suggestion. As discussed in Chapter 3, our optimal baseline, F-score Based Ranking (FsBR), takes into account the quantity of relevant documents in a collection, and the proportion of the total collection size these relevant documents comprise.

Algorithm evaluations in the related domain of collection selection have used a variety of performance measures. For example, Spearman rank correlation, and measures analogous to recall and precision. We employ these measures in our collection suggestion evaluation. However, we also use weighted rank correlation coefficients, and introduce two new measures (Precision@5 and Correct@1) which focus on performance within the top rank positions. These additional performance measures reflect the importance within collection suggestion to accurately identify and rank the best collections. As we are recommending collections for the user to visit themselves, identifying the best collection is vital: recommending sub-par collections wastes the user's time, and may cause frustration. We discussed our selection of performance measures and their merits in Section 3.3 of Chapter 3.

The baseline-testing approach is at the heart of our evaluation strategy. However, as an additional element to our evaluation, we have also developed and used a scenario-based approach. As described in Section 3.4 of Chapter 3, this is intended as a first step in an al-

gorithm evaluation. It allows us to quickly determine whether an algorithm may be suitable for the collection suggestion task, prior to executing the more rigorous baseline-based testing. In each scenario, we model a hypothetical situation, involving only a small number of collections. This enables us to easily reason about the optimal ordering of the collections, according to the specific objectives of collection suggestion. As is evident in Section 8.2, the findings from these tests correlate with findings from baseline-testing.

To summarise, towards a rigorous methodology for evaluating algorithms with respect to collection suggestion, we have made the following four contributions: first, we have formulated a two-tiered evaluation strategy. Second, within this evaluation strategy, as an initial evaluation step, we have developed a set of scenario tests, to identify potentially suitable algorithms.

The second, and main, stage of our evaluation methodology is based around baseline testing from collection selection evaluations. For this, we have developed a new optimal baseline (our third contribution), which reflects the objectives of collection suggestion. Finally, we have applied and developed additional performance measures (for baseline testing), which reflect the objectives of collection suggestion.

In the following section we discuss the test data used in baseline testing, and how this contributes to collection suggestion research.

Test Data

Appropriate test data is a key factor in any information retrieval evaluation, as it enables us to justify the quality of results, and helps to understand the research problem. As collection suggestion falls into the category of information retrieval, we should therefore follow its standards, and ensure our test data meets the expectations of information retrieval research.

In the context of collection suggestion, test data comprises collections of documents, queries and relevance judgements. The latter are necessary to generate optimal collection rankings, against which we evaluate algorithm-produced collection rankings.

As discussed in Chapter 2, in the related domain of collection selection, from which we draw inspiration, it is common to utilise test data formulated from the TREC document corpus (for example, by grouping documents by source and publication date). Such data sets offer a controlled environment, in which algorithm performance can be examined under various conditions (such as varying collection sizes and distribution of relevant documents). As such, for part of the evaluation of algorithms for collection suggestion, we utilise these established test data sets.

However, as also discussed in Chapter 2, there are drawbacks of the TREC-based data sets; in particular, they are not representative of a realistic operational environment, which is likely to contain many collections of varying size, with diverse topic coverage. The synthetic TREC-based document collections however, are heterogeneous in content (each covering several subjects), and the documents within ultimately stem from only a small number of sources: Wall Street Journal, Federal Register, Associated Press, Department of Energy abstracts, Computer Select discs, San Jose Mercury News and U.S. Patents.

Therefore, to ensure a rigorous and realistic test environment, we have developed our own test data set: comprising metadata from real open access repositories.

Developing a test data set from real repository data presents challenges: we require test queries and document relevance judgements. In Chapter 4 we discussed our methodical approach to solving these challenges; we summarise our solutions as follows.

To produce test queries for the data, we select 'seed terms' from the term index; these are then submitted to the Yahoo! Related Suggestions service, which returns query suggestions based on real user queries. We take a sample of these queries to form a set of test queries.

Generating relevance judgements for a test data set is a laborious task. For TREC, human assessors are used to judge the relevance of documents. However, this task is made easier by generating a pool of documents to assess, from the retrieval results of participating systems (see Section 2.2.4 in Chapter 2). As such, the human assessors do not need to examine every document.

We draw on this practice to generate surrogate relevance judgements for documents within our realistic test data set, using an entirely programmatic approach (assessing even a pool of documents for such a large data set is impractical within the scope of this work). For each test query, we use two document ranking algorithms (Lucene and BM25) to rank all the documents within the collections. The documents deemed to be relevant to the query are those that both algorithms agree are relevant (see Section 4.2.4 in Chapter 4 for further details).

To summarise, for our collection suggestion algorithm evaluation we use a combination of synthetic and realistic test data collections. This provides both a controlled test environment in which we can understand how the algorithms work, and a test environment that is realistic of the intended operational setting.

This strategy differs from evaluations in the related domain of collection selection, in which test data sets are often of one type or another. Indeed, utilising realistic test data is often overlooked in algorithm evaluations. This is due in part to the challenges associated with producing queries, and generating document relevance judgements for these. In our work, we have developed programmatic solutions to these issues, which reduce the workload associated with compiling a data set from real data.

Specification of a Collection Suggestion Algorithm

The work we have undertaken in this thesis is driven by the objective of identifying an optimal algorithm for the collection suggestion task. As such, while we have investigated the applicability of algorithms from other domains to this task, we have also developed a new algorithm. Our collection ranking algorithm is geared specifically towards collection suggestion. Therefore with this, we aimed to achieve stronger performance scores than those of the algorithms from relevant domains.

The initial specification of our algorithm, as documented in Chapter 5, is based upon Zobel's criteria for highly ranked collections [66], which embody the objectives of collection suggestion. To rank collections, our algorithm considers: the *proportion* of documents in a collection that contain the query terms; the *commonness* of query terms in the collections; and the *frequency* of query terms within documents in a collection. Each component of our algorithm is a mathematical representation of a single criterion.

As we will discuss in Section 8.2, we evaluate the performance of our algorithm, in addition to its individual component parts, and alternative combinations of those components. Therefore, by evaluating this algorithm, we are essentially systematically testing Zobel's criteria; despite specifying the criteria, Zobel did not explicitly formulate an algorithm from these himself.

Evaluation of Algorithms for Collection Suggestion

As mentioned in the previous section, we are working towards finding an optimal algorithm for collection suggestion. Therefore, we use the evaluation methodology and test data sets we have specified to conduct a large-scale evaluation of the suitability and performance of algorithms, with respect to collection suggestion.

In addition to developing our own algorithm, a logical avenue of enquiry is to examine whether any existing algorithms, from other domains, may be effectively applied to the task. As such, we also test a variety of performance measures from other domains: collection selection and query performance prediction.

While this empirical work is a contribution to collection suggestion (in terms of identifying an appropriate algorithm, and learning what algorithm attributes are successful for collection suggestion), its value reaches wider: evaluation of such a large range of algorithms of this type, over both synthetic and realistic test data sets, has not previously been undertaken.

The results of our evaluation are presented in Chapters 6 and 7, however we summarise and discuss our findings in Section 8.2.

8.2 Discussion of Empirical Results

In order to identify an optimal algorithm for the collection suggestion task, we have conducted a large-scale evaluation of a variety of algorithms, on several test environments. The results of this evaluation are presented in Chapters 6 and 7 of this document. Of the algorithms tested, some were borrowed from other domains, to examine their applicability to this problem. Others were developed specifically for the collection suggestion task, as discussed in Chapter 5.

In this section we provide a discussion of the results given in Chapters 6 and 7. We consider each class of algorithm separately, before summarising our findings.

8.2.1 Doddle Algorithm Configurations

As mentioned above, and in Section 8.1, one contribution of our work was the development of a collection ranking algorithm (Doddle) designed specifically for collection suggestion. In Chapter 5 we presented the initial specification of our algorithm and discussed its derivation. In addition, we conducted an initial evaluation (on the realistic test data sets) of the algorithm, its individual components and alternative configurations of the components.

There are several observations that we can make about the performance of the variants of the Doddle algorithm. Over all test data sets (TREC-based and realistic), two configura-

tions of the algorithm tended to score better than the others: Doddle_RC and Doddle_RC+RP. This applies across all performance measures.

In our initial evaluation of the Doddle algorithms, we found Doddle_RF consistently performed poorly. Recall that Doddle_RF uses only the 'Relative Frequency' metric, which looks at the average occurrences of query terms per document.

Due to its poor performance at that early stage, we omitted the algorithm from further evaluation against other algorithms. However, complete performance scores for this algorithm are given in Appendix D. They support our earlier findings, and indicate that the 'Relative Frequency' component is ineffective for collection suggestion. Indeed, the component appears to be detrimental to the performance of algorithm configurations in which it features; such configurations tend to exhibit lower performance scores across the measures and data sets. Consequently, this result suggests that Zobel's fourth criterion ("There is likely to be documents in the collection in which the term is relatively frequent") is not relevant to the collection suggestion setting.

The performances of the Doddle algorithms (relative to other algorithms tested) vary between test data sets and performance measures. For example, on the RTD set (comprising real repository data), the strongest configurations of the Doddle algorithm (Doddle_RC and Doddle_RC+RP) are either the best algorithms (for example, according to the $\mathcal{R}_n$, $\hat{\mathcal{R}}_n$ and top rank performance measures), or are competitive with the top algorithms.

The results on the TREC-based data sets are a little more variable. On the SYM-236 and UDC-236 data sets, the Doddle algorithms perform quite poorly compared to the other algorithms tested. However, on the data sets derived from the UBC-100 data set, the Doddle_RC and Doddle_RC+RP algorithms are highly competitive, and often score best according the $\mathcal{R}_n$, $\hat{\mathcal{R}}_n$ and top rank performance measures.

Across the data sets, we notice that the best configurations of Doddle often score highly on the $\mathcal{R}_n$, $\hat{\mathcal{R}}_n$ and top rank performance measures. This suggests that the algorithms are relatively good at identifying the most suitable collections within the top rank positions. However, the Doddle configurations often suffer on the $\mathcal{P}_n$ measure, which examines whether collections chosen at a given rank position have *some* relevance to the query (not necessarily the best collection). As such, the Doddle algorithms may be prone to choosing collections that have no relevance to a query.

Finally, on the three rank correlation measures (Spearman, Blest and Da Costa), despite producing lower scores, Doddle_RC and Doddle_RC+RP are no less effective than any other algorithm (across the data sets): there are no statistically significant differences between their scores (SYM-236 is the only exception to this), and those of whichever algorithm scores highest.

8.2.2 Collection Selection Algorithms

In addition to evaluating the performance of our algorithm, in this thesis we also test the suitability of a variety of existing algorithms, for collection suggestion. Many of these algorithms are from the domain of collection selection.

In Chapter 6, we began our evaluation of collection selection algorithms by conducting scenario-based testing. From this, we quickly identified that two of these algorithms (Co-

sine Measure and CVV) were unlikely to be suited to collection suggestion. We omitted these from further evaluation. However, the complete performance results listings in Appendix D confirm our findings; these algorithms frequently have amongst the lowest performance scores, some of which are statistically significantly worse than the best performing algorithms.

Of all the collection selection algorithms tested, CORI was shown to be the most consistent, and was frequently one of the highest scoring algorithms, in particular according to the rank correlation and $\mathcal{P}_n$ measures. However, while CORI tended to perform best across the TREC-based data sets, on the RTD set it was often second to Doddle_RC.

Three of Zobel's lexicon inspection-based algorithms (Inner Product, Skew and Highest-available Similarity) were shown to have variable performance across the data sets. The Highest-available Similarity algorithm was one of the stronger algorithms, across all performance measures, on the TREC-based SYM-236 and UDC-236 data sets. However, in contrast, on the data sets derived from the UBC-100 data set, it was one of the weakest algorithms. The algorithm also performs poorly on the RTD set.

The strongest and most consistent of Zobel's algorithms is Inner Product, which is often amongst the highest scoring algorithms, on all data sets and performance measures. However, the best performances of Zobel's algorithms (Inner Product in particular) are generally seen on the rank correlation measures, and $\mathcal{P}_n$. This suggests they are generally good at choosing collections with some relevance, but the best collections are not necessarily identified at the correct rank positions.

Of the remaining collection selection algorithms, DFPROP is unspectacular; it does not have any noticeably weak performance scores. However, equally, it does not shine through as a particularly strong algorithm.

In contrast, bGlOSS has mixed performance scores. On the TREC-based data sets, it performs competitively on short queries. However, due to the nature of the algorithm it suffers significantly on long queries. In addition, bGlOSS performs poorly overall on the realistic RTD set.

To summarise, we found that the collection selection algorithms tended to perform better on the synthetic TREC-based test data and the $\mathcal{P}_n$ and rank correlation measures, than they did on the realistic test data, and other performance measures. The strongest of these algorithms was CORI, which had performance scores comparable with Doddle_RC.

8.2.3 Query Performance Predictors

An additional class of algorithm we tested for suitability for collection suggestion was query performance predictors, specifically: Distribution of Informative Amount, SCS, AvICTF and NSCQ (see Chapter 2 for details of these algorithms). However, as a result of scenario-based testing in Section 6.1 in Chapter 6, we found these algorithms to be unsuitable very early on.

These algorithms tended to favour collections in which query terms were rare, and thus have more discriminatory power than common terms (this is useful for distinguishing between documents, in traditional document ranking). However, in collection suggestion we want query terms to be common in the highly ranked collections. As such, query performance predictors are counter to the objectives of collection suggestion.

Perusal of the complete performance scores for all implemented algorithms, in Appendix D, confirms the findings from the scenario-based tests, as these achieve the lowest performance scores overall.

8.2.4 Summary

Over the course of our evaluation of the suitability and effectiveness of a variety of algorithms for collection suggestion, we have found that the performances of individual algorithms have often varied over the different test data sets and performance measures. However, this was not unexpected, as previous research in the collection selection domain has often reported on this result. This makes clear the need to thoroughly test algorithms on a variety of test data, in order to identify the strongest and most consistent algorithm; we cannot make reliable judgements from one data set, or indeed on one type of data set (synthetic or realistic) alone.

In our work, we have made a significant contribution towards identifying an appropriate algorithm for ranking collections in a collection suggestion search service. Of all the algorithms we tested in Chapters 6 and 7, we found CORI and the Doddle_RC and Doddle_RC+RP algorithms to show the strongest and most consistent performances, across the test data sets and performance measures. As such, these algorithms are potential candidates for implementation in a collection suggestion search service.

However, while we have identified during our experiments that CORI, Doddle_RC and Doddle_RC+RP perform noticeably and significantly better than other algorithms, it is clear that these algorithms are not perfect, and there is still scope for performance gains. For example, when observing the graphs for the $\mathcal{R}_n$, $\hat{\mathcal{R}}_n$ and $\mathcal{P}_n$ measures, we see that the performance curves for the best algorithms are some way below the optimal performance baseline.

Of particular importance for collection suggestion is to correctly identify the best collection to answer a user's query. This is represented by the Correct@1 measure (giving the percentage of queries for which an algorithm correctly identifies the top ranked collection). However, the algorithm scores on this measure are much lower than we may expect for an algorithm to be effective for this task. On the TREC-based data, the highest Correct@1 scores were 26% on short queries (Doddle_RC+RP), and 28% on long queries (Doddle_RC). The scores are slightly better on the RTD set: 32% on the Titles metadata (Doddle_RC), and 23.5% on the Titles and Descriptions metadata (Doddle_RC and Doddle_RC+RP). There is clear scope for improvement on these.

An additional issue is present with the CORI algorithm. CORI is a mature algorithm, that has frequently been used as a benchmark in collection selection experiments. However, as discussed in Chapter 2 (Section 2.4.3), CORI uses parameters, the values of which should ideally be customised for each data set and query, to achieve optimal performance [13]. While we employed commonly used parameter values in our implementation of CORI (those suggested by Callan et al. [6] in the original specification of the algorithm), we cannot be sure that these values would be sufficient for an operational environment in which the data set would regularly be updated. Indeed, we certainly could not change those values for each

query. As such, the suitability of CORI for implementation in a search service for collections is still in question.

Despite this, given that CORI is considered an established collection selection benchmark (that has performed admirably in our experiments), developing new algorithms (Doddle_RC and Doddle_RC+RP) that are similarly, or more, effective than CORI is a notable achievement. In addition, we have learnt that two collection-level statistics (how common a query term is in a collection, and proportion of documents in a collection containing a query term) are important and useful for ranking collections for collection suggestion. In contrast, the document-level statistic (in-document term frequency) is not helpful in this particular setting.

8.3 Limitations

The work presented in this document does have some limitations. In particular, we do not provide a complete solution to the collection suggestion problem; rather, we have conducted fundamental research (the development of a methodology for evaluating the effectiveness of collection ranking algorithms, compilation of test data and an initial evaluation of a range of algorithms) that provides a strong foundation for further work in the area. While we have conducted an initial evaluation of a range of collection ranking algorithms, and found some algorithms to perform satisfactorily, there is scope for the development of a more effective algorithm.

For our empirical work, we have used a variety of test data sets; this ensures that algorithms are tested in a range of environments. However, each of the data sets we use have associated weaknesses. For example, while the synthetic TREC-based data sets are easily reproducible, and have human-assessed relevance judgements, they are not representative of the intended operational environment (as discussed in Chapter 2). To ensure that we do test algorithms on realistic test data, we compiled data sets from real digital repositories. However, due to the impracticalities of producing human-assessed document relevance judgements, we opted to programmatically generate synthetic relevance judgements, using document ranking algorithms. This approach is itself not ideal, due to the limited effectiveness of the two algorithms used. Therefore, within our test data there is a clear trade-off between synthetic test collections using reliable human-assessed relevance judgements, and realistic test collections with synthetic relevance judgements. This is unlikely to be something that can be overcome without significant investment of time and resources.

In the following section we reflect on possible avenues for future work, with respect to progressing towards an effective search service for domain-specific collections.

8.4 Future Work

The task of supporting a user in identifying resources that are relevant to their current and future interests is a valuable one. We have made several fundamental contributions towards this task, which lay the foundations for future, systematic development and evaluation of an optimal collection ranking algorithm for the task.

Specifically, we have developed a rigorous methodology and compiled varied test data, for evaluating the suitability and performance of algorithms for the collection suggestion task. Without these, extensive evaluation of algorithms for collection suggestion could not take place. In addition, we have specified a new collection ranking algorithm, designed specifically for collection suggestion; and have conducted an initial evaluation of both our own algorithm (and its alternative variants), and a large range of existing algorithms from relevant domains.

Through our empirical work, we have highlighted approaches and algorithms that are not effective for collection suggestion. We have also identified some algorithms, such as CORI and our own Doddle_RC and Doddle_RC+RP algorithms, that appear to be adequate for recommending collections. However, there is still scope for performance gains.

Given this, future work in this domain could involve a systematic process of combining effective algorithms tested herein, or their component parts, to seek out improvements in performance. Alternatively, there may be other appropriate and valuable metrics that have not been identified here. While the apparatus we have developed to support our empirical work would make this task more manageable, such a systematic and in-depth evaluation of algorithms is a substantial piece of work in its own right.

As discussed in Chapter 2, previous research [13] has found that the performance of CORI varies greatly based on the values used in its parameters. As such, future work in the collection suggestion domain could investigate whether the performance of CORI could be improved by adjusting these values. However, as the effectiveness of these values also changes based on the data set used and query to be executed, finding optimal values for these parameters within an operational environment seems impracticable.

A further potential direction for future algorithmic work is to investigate the use of alternative collection attributes; for example, the reputation of a collection, or the freshness of a collection (how frequently new material is added). This is along a similar vein to the work of Fuhr [19, 20, 21, 39], which looks at the costs associated with visiting a collection. However, while interesting, such work would require amendments to the evaluation methodology we have developed (specifically, the optimal ranking), to reflect the importance of these new characteristics to ranking collections. Again, this approach represents a significant body of work in its own right, and therefore is beyond the scope of this thesis.

Our empirical work presented herein has taken a purely objective approach to experimentation; looking at the effectiveness of algorithms with various performance measures, over a robust and reproducible test environment. Such an approach has significant value; only after a suitable collection ranking algorithm is identified for this task, can user-based evaluation (subjectively measuring the value of a collection suggestion service, and the system's effectiveness from a user's standpoint) occur on a mature prototype. Empirical work of this type is left as future work.

Appendices

This part of the thesis provides appendices to accompany our main discussions in the previous chapters. We summarise the contents of the appendices, and the chapters to which they are relevant, as follows:

Appendix A (Abstract Test Scenarios): This appendix accompanies Chapter 3 (specifically, Section 3.4), in which we introduce the scenario-based testing component of our strategy for evaluating algorithms for collection suggestion. Appendix A provides an overview of the scenario-based testing strategy and its motivation. In addition, complete specifications of each test scenario are given.

Appendix B (Test Data): This appendix accompanies Chapter 4, in which we present and discuss our choice of test data for our empirical work. Appendix B includes data not appropriate to include within Chapter 4: lists of test queries, the list of collections within our open access repository Refined Test Data (RTD) set, and graphs showing the number of relevant documents for each query.

Appendix C (Surrogate Relevance Judgements for Open Access Repository Test Data Sets): This appendix accompanies Chapter 4, Section 4.2.4 in particular, in which we introduce our technique for generating surrogate document relevance judgements for our open access repository test data sets. The relevance judgements are necessary for generating an 'optimal' ranking of collections. Appendix C describes how we arrived at our technique for generating surrogate relevance judgements.

Appendix D (Algorithm Performance Scores): This appendix accompanies Chapters 5, 6 and 7, in which we conduct a large-scale evaluation of our own collection suggestion algorithm, in addition to testing the suitability of algorithms from the domains of collection selection and query performance prediction. Appendix D includes complete performance scores for all implemented algorithms, over all test data sets.

Appendix A

Abstract Test Scenarios

This appendix provides details of the test scenarios we employ as part of our algorithm evaluation strategy, described in Chapter 3. In Section A.1 we begin by providing a brief overview of the scenario-based testing strategy, and its motivation. Following this, in Section A.2 we recap the format of the scenarios, and discuss our choice of values for term and collection statistics. Finally, in Section A.3 we provide detailed specifications of each test scenario.

A.1 Overview

As discussed in Chapter 3, scenario-based testing is an initial stage in our evaluation of algorithms, with respect to the collection suggestion task. In scenario-based testing, we specify a series of hypothetical situations, involving only a small number of collections. This enables us to easily reason about the best ordering of the collections, according to the specific objectives of collection suggestion.

The scenario-based testing essentially provides a 'health-check' for algorithms. Due to their simplicity, execution of these tests is cheap, enabling us to quickly determine which algorithms are likely to be suitable for the collection suggestion task. As a result, we can rule out poor performers, prior to executing the more comprehensive and expensive baseline-based testing (see Section 3.2 in Chapter 3).

A.2 Scenario Characteristics

In this section we recap the format of a test scenario, and discuss how we reason about the optimal ordering of collections within a scenario. In addition, we also describe how we arrived at values for the various term and collection statistics.

A.2.1 Format of a Scenario

In each test scenario, we specify a query in abstract terms, in the form $t_1 \ldots t_n$. This represents the information need of the user.

In addition, in each scenario we model three collections, which we reference as C_A, C_B and C_C. For each collection we specify the statistics that are likely to be used by the algorithms we are testing: the number of documents it contains, the total number of terms it contains, and the term and document frequencies of each of the query terms.

By varying the quantities and ratios of the collection statistics, in addition to the query length and number of query terms matched, we create different test cases. As such, we can identify any strengths and weaknesses of the algorithms, and determine whether they rank collections in accordance with the goals of the collection suggestion task.

A.2.2 Justification of Optimal Ranking

Due to the small number of collections modelled, it is easy to reason about the optimal ordering of collections within a scenario. For collection suggestion, a highly ranked collection should contain a large number of relevant documents, and these documents should comprise a large proportion of the collection; this suggests the collection is *about* the query topic.

These properties are expressed in criteria for highly ranked collections, specified by Zobel [66]. They state that a collection should be ranked highly if for each query term:

1. The term occurs in the collection;
2. The term is common in the collection (relative to the other collections);
3. The collection contains a relatively high proportion of documents featuring the term; and
4. There are likely to be documents in the collection in which the term is relatively frequent [66].

We use these criteria to reason about the best ordering of collections within a scenario. However, in the interests of simplicity and consistency, we specify the attributes of the collections within a scenario such that the best ordering is always C_A, C_B, C_C. That is, C_A is always the best collection to match the query. C_B is considered a 'runner up', while C_C is always distinctly worse than the other two.

A.2.3 Term and Collection Statistics

In each test scenario specified in Section A.3, we specify collection statistics (the number of documents and terms), and term statistics (term and document frequencies) for each query term. To keep calculations simple, these values are generally small, most notably in terms of collection size.

However, we do attempt to make the scenario statistic values realistic of an operational environment. As such, total terms, and term and document frequencies are based on values observed in real collection data.

Prior to the development of our open access repository data sets (see Section 4.2 in Chapter 4), we harvested metadata from a small sample of collections. In this data, we

found metadata documents (using only Title and Description metadata fields) to contain an average of 90 terms. We used this figure to calculate the total terms in each collection in the scenarios, given the number of documents each collection contains. In addition, pairs of term and document frequencies were randomly selected from this harvested data, to provide realistic statistics for query terms.

A.3 Specification of Scenarios

In the following sections we give specifications of each scenario used in our scenario-based testing approach. For each we provide: a description of the hypothetical situation we are modelling; the specification of the collection statistics; and a justification of the choice of the optimal ordering of the collections, with reference to the criteria given in Section A.2.2.

A.3.1 Scenario 1

Summary: Collections are equal in size. We vary the number of documents that contain the query terms.

Scenario Description: Consider the situation where we have three repositories of a similar size. One collection is a specialist collection, where all of its documents are concerned with a specific topic; for example, Java programming. The second collection is a more general collection, with documents covering a range of topics, such as a computer science collection. The third collection is more general still, containing documents on a variety of subjects.

A user poses a query, such as "Java GUI programming Swing JFrame". We would expect the specific collection, which contains material on Java programming, to be the most suitable to answer this query; and indeed be a useful source of information for future queries on this topic.

Scenario Specification: To model this scenario we have three collections: C_A, C_B and C_C. Each contains 100 documents, and comprises a total of 9000 terms. We consider C_A to be the specialist Java programming collection. C_B is the computer science collection, and C_C is the more general collection. The term statistics for the query, which is of the form $t_1\ t_2\ t_3\ t_4\ t_5$, are given in Table A.1.

Optimal Collection Ordering: C_A, C_B, C_C. Our justification for this is as follows:

1. Each collection contains all terms in the query;
2. Each query term is most common in C_A;
3. C_A has the highest proportion of documents that feature each query term; and
4. In general, C_A is most likely to contain documents in which the query terms are relatively frequent.

As such, we identify C_A as the best candidate to meet the information need.

Table A.1: Collection and query term statistics for Scenario 1; $f_{c,t}$ and $df_{c,t}$ give the number of occurrences and document frequency of term t in collection c, respectively.

	C_A (100 docs; 9000 terms)		C_B (100 docs; 9000 terms)		C_C (100 docs; 9000 terms)	
	$f_{c,t}$	$df_{c,t}$	$f_{c,t}$	$df_{c,t}$	$f_{c,t}$	$df_{c,t}$
t_1	53	14	28	7	1	1
t_2	13	6	10	3	1	1
t_3	36	7	8	3	1	1
t_4	3	3	2	2	1	1
t_5	8	5	1	1	1	1

A.3.2 Scenario 2

Summary: Collections differ in size, however the *proportion* of documents that contain the query terms (in C_A and C_B) is the same.

Scenario Description: In this scenario, two of our collections specialise in the same subject, astronomy for example. The two collections contain a similar proportion of term occurrences within the documents, but one collection is much larger than the other. The third collection is a general collection, containing documents on a wide range of different topics.

For an astronomy related query, it is not explicitly clear which of the first two collections would suit the information need best, as they appear to be roughly equal. However, in this instance it is likely that the first, larger, collection would offer better long term value to the user.

Scenario Specification: For our model, C_A is the larger astronomy collection, containing 200 documents and a total of 18000 terms. The second astronomy collection, C_B, is half the size of C_A, with 100 documents and 9000 terms. Collection C_C also contains 100 documents and 9000 terms. The term statistics for the query, which is of the form $t_1\ t_2\ t_3\ t_4\ t_5$, are given in Table A.2.

Optimal Collection Ordering: C_A, C_B, C_C. Our justification for this is as follows:

1. Each collection contains all terms in the query;
2. Each query term is equally common in C_A and C_B;
3. C_A and C_B have an equal proportion of documents featuring the query terms; and
4. C_A and C_B are equally likely to contain documents in which the query terms are relatively frequent.

As we can see, distinguishing between repositories C_A and C_B is difficult, as they are considered equal based on the criteria above. However, we suggest that C_A is the better candidate to meet the information need as it has *more* documents about the subject. It is a larger collection, and thus may have more content with future relevance.

Table A.2: Collection and query term statistics for Scenario 2; $f_{c,t}$ and $df_{c,t}$ give the number of occurrences and document frequency of term t in collection c, respectively.

	C_A (200 docs; 18000 terms)		C_B (100 docs; 9000 terms)		C_C (100 docs; 9000 terms)	
	$f_{c,t}$	$df_{c,t}$	$f_{c,t}$	$df_{c,t}$	$f_{c,t}$	$df_{c,t}$
t_1	106	28	53	14	1	1
t_2	26	12	13	6	1	1
t_3	72	14	36	7	1	1
t_4	6	6	3	3	1	1
t_5	16	10	8	5	1	1

A.3.3 Scenario 3

Summary: Collections differ in size, but the *quantity* of documents that contain the query terms (in C_A and C_B) is the same.

Scenario Description: One collection in this scenario is a specialist collection, containing documents on just one topic, World War II for example. A second collection is one that collates documents from other sources. As such it contains the same documents as the first, plus some additional content on an unrelated subject, such as the Black Death. As the two subjects are disjoint, there is no overlap in their vocabularies. The third collection is a general collection, containing documents on a wide array of subjects.

A user submits a query relating to World War II. In this instance we can argue that either of the first two repositories would be a suitable candidate to browse. However, we suggest that the smaller, specialist collection would be more appropriate: it is specifically concerned with the query topic, and it is not diluted with irrelevant material.

Scenario Specification: Our model collections are as follows: C_A is the specialist collection, containing 100 documents and a total of 9000 terms. C_B has 200 documents (100 of which are also in C_A), and 18000 terms. C_C is the general collection, containing 100 documents and 9000 terms. The term statistics for the query, t_1 t_2 t_3 t_4 t_5, are given in Table A.3. We note that since there is no overlap in the subject vocabularies for collections C_A and C_B, the values are identical.

Optimal Collection Ordering: C_A, C_B, C_C. Our justification for this is as follows:

1. Each collection contains all terms in the query;
2. Each query term is more common in C_A;
3. C_A has the highest proportion of documents featuring the query terms; and
4. C_A and C_B are equally likely to contain documents in which the query terms are relatively frequent.

From this, we gain support that the specialist collection (C_A) should be ranked first.

Table A.3: Collection and query term statistics for Scenario 3; $f_{c,t}$ and $df_{c,t}$ give the number of occurrences and document frequency of term t in collection c, respectively.

	C_A (100 docs; 9000 terms)		C_B (200 docs; 18000 terms)		C_C (100 docs; 9000 terms)	
	$f_{c,t}$	$df_{c,t}$	$f_{c,t}$	$df_{c,t}$	$f_{c,t}$	$df_{c,t}$
t_1	53	14	53	14	1	1
t_2	13	6	13	6	1	1
t_3	36	7	36	7	1	1
t_4	3	3	3	3	1	1
t_5	8	5	8	5	1	1

A.3.4 Scenario 4

Summary: Collections are equal in size. A single-term query is used. We vary the number of documents that contain the query term.

Scenario Description: This is a simplistic scenario, where we aim only to check that an algorithm can produce a suitable ranking when a query consists of only one term. That is, the algorithm does not depend on there being more than one query term.

Scenario Specification: We model this scenario with three collections of equal size: C_A, C_B and C_C. Each contains 100 documents, and comprises a total of 9000 terms. The term statistics for the query consisting of term t_1 are given in Table A.4.

Optimal Collection Ordering: C_A, C_B, C_C. Our justification for this is as follows:

1. Each collection contains all the terms in the query;
2. The query term is more common in C_A;
3. C_A has the highest proportion of documents featuring the query term; and
4. C_B is most likely to contain documents in which the query term is relatively frequent.

As such, it seems reasonable that C_A should be ranked first, followed by C_B.

Table A.4: Collection and query term statistics for Scenario 4; $f_{c,t}$ and $df_{c,t}$ give the number of occurrences and document frequency of term t in collection c, respectively.

	C_A (100 docs; 9000 terms)		C_B (100 docs; 9000 terms)		C_C (100 docs; 9000 terms)	
	$f_{c,t}$	$df_{c,t}$	$f_{c,t}$	$df_{c,t}$	$f_{c,t}$	$df_{c,t}$
t_1	53	14	13	6	1	1

A.3.5 Scenario 5

Summary: Collections are equal in size. C_B has the same term occurrences as C_A for some query terms, but does not match all query terms.

Scenario Description: In this scenario, we have three collections of similar size. The first collection is a specialist collection, dealing with material on Java programming. The second collection is a more general collection, but contains material related to that in the first, for example a computer science collection. The third collection is very general, and has small numbers of documents about many different subjects.

We issue the query "Java GUI programming Swing JFrame". The specialist collection is able to match all query terms, and thus is seen as the most suitable collection to browse. The second collection matches some query terms, but cannot match the more subject specific terms, such as "Swing" and "JFrame", for example. Finally, the third collection, due to its generality, can match only one term.

Scenario Specification: We model this scenario with collections: C_A, C_B and C_C. C_A is the specialist collection, C_B is the general computer science collection, and C_C is the very general collection. Table A.5 shows the term statistics for the query, t_1 t_2 t_3 t_4 t_5.

Optimal Collection Ordering: C_A, C_B, C_C. Our justification for this is as follows:

1. C_A is the only collection containing all the query terms;
2. In general, query terms are more common in C_A;
3. In general, C_A has a higher proportion of documents featuring query terms; and
4. In general, C_A is most likely to contain documents in which the terms are relatively frequent.

This suggests that C_A should be ranked highest.

A.3.6 Scenario 6

Summary: Collections are equal in size. C_B has higher occurrences of some query terms than C_A, but does not match all query terms.

Table A.5: Collection and query term statistics for Scenario 5; $f_{c,t}$ and $df_{c,t}$ give the number of occurrences and document frequency of term t in collection c, respectively.

	C_A (100 docs; 9000 terms)		C_B (100 docs; 9000 terms)		C_C (100 docs; 9000 terms)	
	$f_{c,t}$	$df_{c,t}$	$f_{c,t}$	$df_{c,t}$	$f_{c,t}$	$df_{c,t}$
t_1	53	14	53	14	1	1
t_2	13	6	13	6	0	0
t_3	36	7	36	7	0	0
t_4	3	3	0	0	0	0
t_5	8	5	0	0	0	0

Scenario Description: In this scenario we again consider three collections of similar size. The first two collections cover multiple topics within a broad subject area, for example a computer science collection. The third collection has documents from many different subject areas.

We issue a query relating to computer science, and find that the first collection has moderate occurrences of all query terms. The second collection however matches only some of the query terms, but the number of occurrences of these terms are much higher than the first. The third collection matches only one term. The question is, should we favour the collection with moderate numbers of all query terms, or that with higher occurrences of some query terms? Our view is that the first collection is a better match, as all query terms are present.

Scenario Specification: Collections C_A, C_B and C_C are used to model this scenario, each containing 100 documents and 9000 terms. C_A represents the computer science collection containing moderate occurrences of all query terms. C_B models the second computer science collection and C_C the general collection. The term statistics for the query, t_1 t_2 t_3 t_4 t_5, are given in Table A.6.

Optimal Collection Ordering: C_A, C_B, C_C. Our justification for this is as follows:

1. C_A is the only collection containing all the query terms;
2. Terms present in C_B are more common than those in C_A;
3. Terms present in C_B occur in a higher proportion of documents than those in C_A; and
4. In general, C_A is most likely to contain documents in which the terms are relatively frequent.

As such, it is difficult to determine which collection should rank highest. However, in this instance, a collection that matches all query terms (C_A) is favourable.

Table A.6: Collection and query term statistics for Scenario 6; $f_{c,t}$ and $df_{c,t}$ give the number of occurrences and document frequency of term t in collection c, respectively.

	C_A (100 docs; 9000 terms)		C_B (100 docs; 9000 terms)		C_C (100 docs; 9000 terms)	
	$f_{c,t}$	$df_{c,t}$	$f_{c,t}$	$df_{c,t}$	$f_{c,t}$	$df_{c,t}$
t_1	28	7	60	21	1	1
t_2	10	3	0	0	0	0
t_3	8	3	53	7	0	0
t_4	2	2	0	0	0	0
t_5	1	1	3	3	0	0

A.3.7 Scenario 7

Summary: Collections differ in size. C_B is larger than C_A, with a higher quantity (but a smaller proportion) of relevant documents. This represents polysemy.

Scenario Description: Here, we have a situation similar to Scenario 3: we again have a specialist collection containing documents on World War II. The second collection also contains these documents, along with some additional content. However, on this occasion the additional content is on a similar subject, World War I for example. As such, there is an overlap in the vocabularies of the two topics (this is an example of polysemy; where words may have multiple meanings). The third collection is as before, a general collection with documents covering a range of subjects.

As in Scenario 3, the query is related to World War II, and as such we would prefer the specialist collection to be ranked highest. Our reasoning in this instance is that if the user conducted a search at the second collection, they would likely be shown documents that were not relevant to their information need, as a result of the polysemy problem.

Scenario Specification: To model the scenario, C_A is the specialist collection, containing 100 documents with 9000 terms. C_B, with the additional content, has 200 documents, with 18000 terms in total. Finally, C_C is the general collection and has 100 documents, containing a total of 9000 terms. Table A.7 shows the term statistics for the query, which is of the form $t_1\ t_2\ t_3\ t_4\ t_5$. We see that in this scenario, C_B has additional occurrences of some of the query terms to represent the overlap in vocabularies.

Optimal Collection Ordering: C_A, C_B, C_C. Our justification for this is as follows:

1. Each collection contains all terms in the query;
2. In general, the query terms are more common in C_B;
3. C_A has the highest proportion of documents featuring each query term; and

4. In general, C_B is most likely to contain documents in which the query terms are relatively frequent.

Here, the criteria lean towards suggesting C_B as the best collection. However, we have determined from the scenario description that C_A would be the most effective collection to answer the user's information need. This demonstrates that polysemy is a challenging problem to overcome.

Table A.7: Collection and query term statistics for Scenario 7; $f_{c,t}$ and $df_{c,t}$ give the number of occurrences and document frequency of term t in collection c, respectively.

	C_A (100 docs; 9000 terms)		C_B (200 docs; 18000 terms)		C_C (100 docs; 9000 terms)	
	$f_{c,t}$	$df_{c,t}$	$f_{c,t}$	$df_{c,t}$	$f_{c,t}$	$df_{c,t}$
t_1	53	14	82	21	1	1
t_2	13	6	26	9	1	1
t_3	36	7	36	7	1	1
t_4	3	3	3	3	1	1
t_5	8	5	12	9	1	1

Appendix B

Test Data

In this appendix we present additional data (such as lists collections within the data sets, lists of test queries, and graphs indicating the number of relevant documents per test query) associated with the test data sets we use in our evaluation of algorithms for collection suggestion.

The majority of this information is associated with our open access repository test data sets. As such, Sections B.1 and B.2 provide data associated with the Initial and Refined Test Data sets respectively, while Section B.3 deals with the TREC-based test data sets.

B.1 Initial Test Data (ITD)

In this section we specify the test queries (Section B.1.1) associated with our Initial Test Data open access repository data set, and present the number of relevant documents associated with each query (Section B.1.2), according to our surrogate document relevance judgements.

B.1.1 Queries

The set of 50 test queries associated with the ITD set is as follows:

1. preventing transmission of infectious disease
2. techniques for diagnosis of mental illness
3. fluorescence imaging endoscope for early detection of gastrointestinal malignancy
4. hadron collider
5. orthodontics
6. paediatric dentistry
7. 3d archaeological reconstruction and visualisation
8. glass blowing
9. elgar
10. lorenz manifold
11. dublin core metadata
12. mass spectrometry

13. standards of secondary school education in the UK
14. solar flares
15. contamination of soil from depleted uranium
16. ancient egyptian pharaohs
17. culture in tudor england
18. glaucoma
19. macular dystrophy
20. common causes of heart disease
21. human rights act 1998
22. capabilities and limitations of long wavelength observations from space
23. effect of breast feeding on intelligence in children
24. chinese exports
25. causes and prevention of mrsa in hospitals
26. the rise and fall of adolf hitler and nazi germany
27. mummification process and burial customs in ancient egypt
28. theology
29. development coordination disorder in children
30. schwarzschild black holes
31. operation overlord normandy landings 1944
32. structural operational semantics
33. semantic web
34. hidden markov models
35. psoriasis
36. tantric buddhism
37. game theory
38. economic management health
39. parkinson disease senility
40. medieval culture
41. distribution of sediment on the seabed
42. extensional plate tectonics
43. coastal erosion
44. diagnosis of pancreatic cancer
45. managing menopause with hormone replacement therapy
46. late antique and medieval mosaics in italy
47. economics of taxation
48. situation-aware wireless networks
49. agricultural soil properties
50. higgs boson

B.1.2 Relevant Documents Per Query

The graphs in Figures B.1 and B.2 show the number of relevant documents associated with each test query, for the Title metadata and Title and Description metadata respectively, according to our surrogate document relevance judgements.

We observe that the number of relevant documents ranges drastically between different

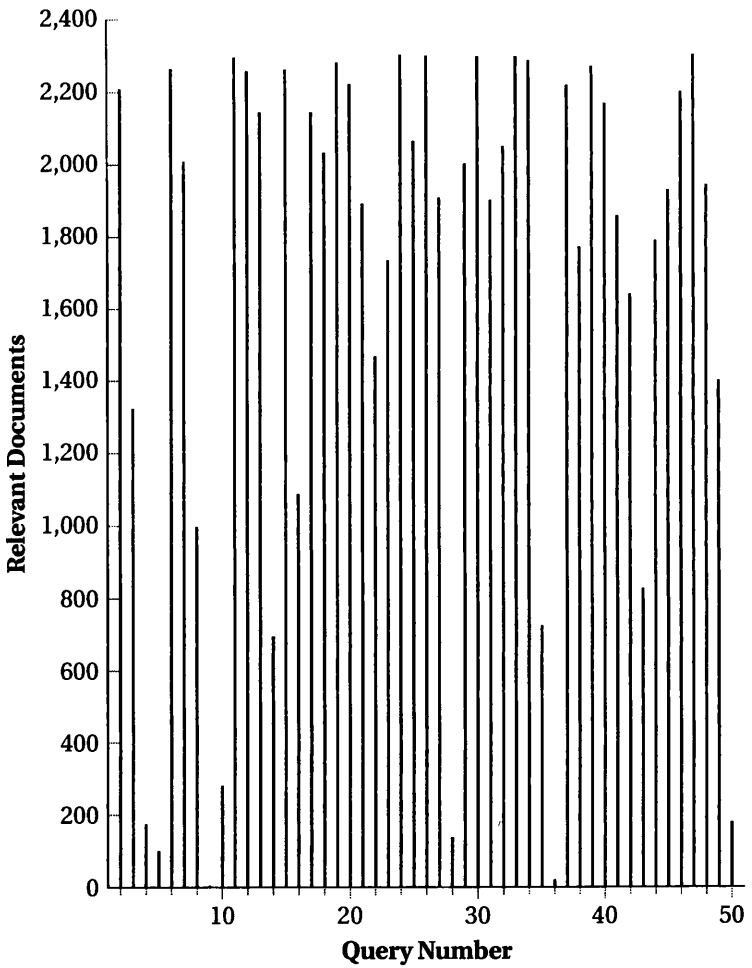


Figure B.1: Number of relevant documents (based on Title only data) for each ITD query.

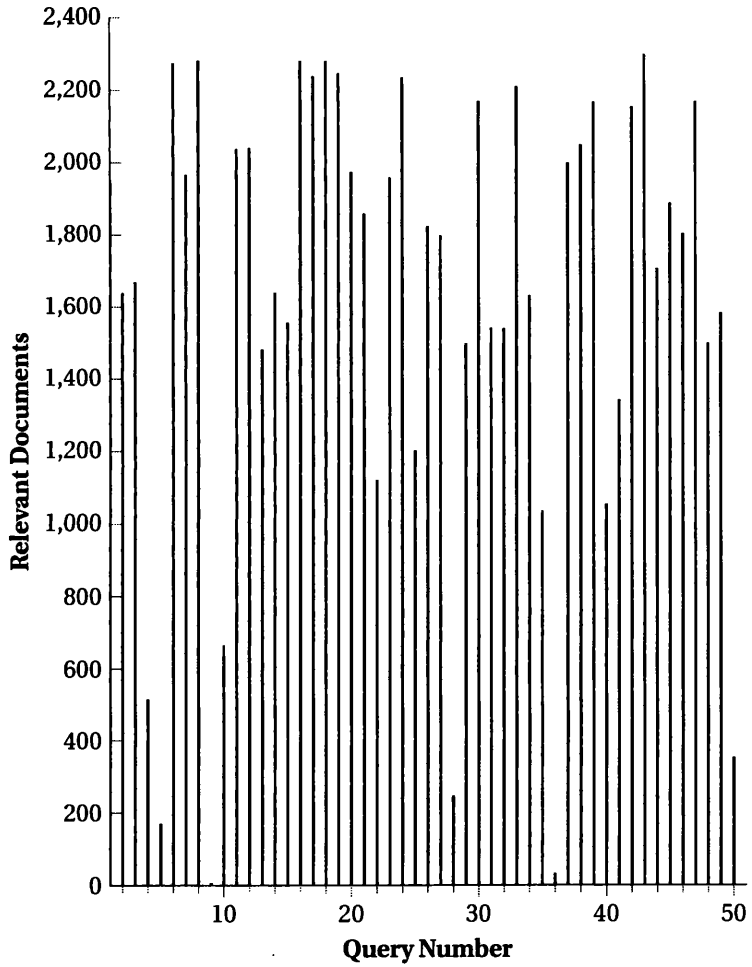


Figure B.2: Number of relevant documents (based on Title and Description data) for each ITD query.

queries: some match over 2000 documents, while others return only a handful of relevant documents. While a similar picture can be observed in similar graphs in Sections B.2 and B.3 for the RTD and TREC-based data sets, the ITD set seems to have a larger number of relevant documents per query, on average. This may be a side effect of the presence of the very large PubMed database, which may include many documents with low relevance to the queries. In addition, the presence of this collection in ITD means the data set contains many more documents than the other test collections.

B.2 Refined Test Data (RTD)

In this section we provide data associated with the Refined Test Data set: a list of all harvested repositories and the number of collections they contain (Section B.2.1), test queries (Section B.2.2), and graphs showing the number of relevant documents per query (Section B.2.3).

B.2.1 Collections

Table B.1 lists the 100 real repositories we harvested for the RTD set, and the number of documents each contains.

Table B.1: Collections in the Refined Test Data set.

#	Repository Name	Number of Documents
1	Caltech Graduate Aeronautical Laboratories Solid Mechanics Technical Reports	7
2	Edinburgh DataShare	12
3	Latin American Development Archive (LADARK)	13
4	Caltech Graduate Aeronautical Laboratories Fluid Mechanics Technical Reports	28
5	Caltech Large-Eddy Simulation and Subgrid-Scale Modeling for Turbulent Mixing and Reactive Flows	29
6	ECS Student Portfolio (University of Southampton)	37
7	Glamorgan Dspace	49
8	Nottingham Modern Languages Publications Archive	63
9	Te Tumu Eprints Repository (University of Otago)	69
10	CaltechMALN	72
11	Caltech Archives Oral Histories Online	110
12	CAV2001: Fourth International Symposium on Cavitation	111
13	OpenDEPOT.org	132
14	Open Repositories 2008 Publications	143
15	Caltech Control and Dynamical Systems Technical Reports	147

Table B.1: Collections in the Refined Test Data set (continued).

#	Repository Name	Number of Documents
16	Analytical Sciences Digital Library	451
17	University of Birmingham Research Archive E-prints Repository	496
18	eCrystals - University of Southampton	496
19	Open Access Institutional Repository at Robert Gordon University	530
20	Anglia Ruskin Research Online	539
21	British History Online	551
22	University of Liverpool Research Archive	722
23	University of Birmingham Research Archive E-papers Repository	759
24	CEDA Repository	776
25	Digital Resource Commons - University of Toledo	933
26	University of Limerick Institutional Repository	948
27	Virginia Tech Computer Science Technical Reports	983
28	University of Chester Digital Repository	994
29	Research@St Andrews	1111
30	Harvard Smithsonian Digital Video Library	1197
31	Roehampton University Research Repository	1204
32	Georgetown Law Scholarly Commons	1220
33	MIMS EPrints (University of Manchester)	1240
34	Nottingham eTheses	1294
35	ETSU Electronic Thesis and Dissertation Archive	1324
36	Scholarly Commons @ AUT University	1411
37	Solent Electronic Archive	1429
38	RADAR (Oxford Brookes University)	1522
39	Otago University Research Archive	1547
40	University of Birmingham Research Archive E-theses Repository	1605
41	Sussex Research Online	1607
42	ResearchArchive at Victoria University of Wellington	1694
43	Wolverhampton Intellectual Repository and E-theses	1737
44	Pharmacy Eprints	1928
45	Glasgow Theses Service	2081
46	Colorado State University Libraries Digital Repository	2282
47	Digital Commons@Becker	2284
48	NECTAR	2758
49	Shocker Open Access Repository	2778
50	DSpace at Drexel University Library	3007
51	Nature Preceedings	3011

Table B.1: Collections in the Refined Test Data set (continued).

#	Repository Name	Number of Documents
52	bepress Legal Repository	3139
53	Bowling Green State University Digital Resource Commons	3191
54	Documenting the American South	3206
55	University of Lincoln Institutional Repository	3249
56	DSpace at New York University	3270
57	DigitalCommons@Bryant University	3281
58	Sheffield Hallam University Research Archive	3315
59	Goldsmiths Research Online	3567
60	DigitalCommons@Pace	3673
61	Woods Hole Open Access Server	4631
62	Murdoch University Research Repository	5052
63	Brunel University Research Archive	5120
64	Aquatic Commons	5661
65	ResearchSpace@Auckland	6064
66	KU ScholarWorks	6150
67	Institute of Education EPrints	6322
68	Leicester Research Archive	6464
69	Center for Jewish History Digital Collections	7049
70	IUScholarWorks	7615
71	Durham Research Online	7877
72	Research Online @ ECU	7927
73	Irish Health Repository	8495
74	DLynx - Rhodes College Archives Digital Collection	9542
75	University of Tasmania Eprints Repository	9886
76	Kingston University Research Repository	11342
77	Digital Repository at the University of Maryland	11393
78	Jorum Open	12350
79	Macquarie University Research Online	14523
80	DigitalCommons@ILR	14640
81	Minds @ UNiversity of Wisconsin	14985
82	Digital Library for Earth System Education	15291
83	Archive of European Integration	15985
84	Flinders Academic Commons	17173
85	eCommons@Cornell	17817
86	Connexions	18108
87	Caltech Authors	23140
88	La Trobe University Research Repository	24180
89	LSE Research Online	27509
90	PEAK Digital	30855

Table B.1: Collections in the Refined Test Data set (continued).

#	Repository Name	Number of Documents
91	JScholarship	32418
92	Warwick Research Archives Portal Repository	37058
93	Enlighten (Glasgow)	43277
94	Texas A&M Repository	45276
95	Leodis - A photographic archive of Leeds	56852
96	Adelaide Research & Scholarship	60109
97	e-Prints Soton	63654
98	FSU Libraries Digital Library Center Institutional Repository	70464
99	University of Queensland eSpace	144208
100	DSpace @ Cambridge	196224

B.2.2 Queries

In this section we list the queries used in the RTD set, as generated using the programmatic approach discussed in Section 4.2.3 of Chapter 4. We first provide a set of 50 test queries, followed by a set of 200.

Set of 50 Queries

The set of 50 test queries associated with the RTD set is as follows:

1. pseudostratified columnar epithelium
2. darlingtonia state natural site
3. db file multiblock read count
4. biochemical activity of wee1
5. baldock health care center
6. how to preadsorb antibody
7. john grisham latest novel
8. que tipo de cuerpo tengo
9. crocodylia alligatoridae
10. definition of exopolymer
11. immeasurably long time
12. gideons international
13. hemotropic mycoplasma
14. battle of stalingrad
15. ten rillington place
16. dr dino creationists
17. methyl cyclopentane
18. flagman of america
19. portland indymedia
20. how to make cocito
21. cheech and chong
22. fty720 treatment

23. vayda's seafood
24. farmscapes game
25. captopril drug
26. figurative art
27. hauptmann voss
28. solace meaning
29. baywood greens
30. elena semenova
31. mayra veronica
32. akhri chataan
33. peter motley
34. trf auctions
35. pindolol isa
36. owen humpage
37. egon schiele
38. schlink haus
39. mark mcguinn
40. piazza honda
41. unurban cafe
42. flsa status
43. mark straka
44. expert tire
45. john breen
46. huh7 cells
47. purdin mo
48. thesaurus
49. d4x rpm
50. ptld

Set of 200 Queries

The set of 200 test queries associated with the RTD set is as follows:

1. new t-mobile sidekick 4 cell phone shuriken
2. regular panelist wait wait don't tell me
3. bultmann new testament and mythology
4. nonseminomatous testicular cancer
5. lampbrush and polytene chromosome
6. unión de naciones suramericanas
7. macroalbuminuria and definition
8. mission hospital mission viejo
9. legatta a un granello di sabia
10. acemoglu johnson and robinson
11. emmeline pankhurst biography
12. cataclastic metamorphic rock
13. chaser lounge jubblies forum
14. whanaungatanga relationships
15. san gimignano italy tuscany
16. function of dithiothreitol
17. golden state water company
18. arnhold and s bleichroeder
19. ubat kuatkan tenaga batin
20. delosperma table mountain

21. schizophreniform disorder
22. pinellia ternata rhizome
23. march militaire schubert
24. macrodermabrasion munich
25. acamprosate side effects
26. benedetto's land o lakes
27. sphaerodactylus elegans
28. reese koffler stanfield
29. polycythemia rubra vera
30. chilopsis desert willow
31. scedosporium sinusitis
32. trichonympha sphaerica
33. harry papaconstantinou
34. acer negundo box elder
35. pedilanthus variegatus
36. remsenburg speonk ufsd
37. american unilateralism
38. gibsonia pennsylvania
39. macroeconomist salary
40. changbaishan mountain
41. colonocytes wikipedia
42. fenbendazole for cats
43. bernadette of lourdes
44. sw franciszek z asyzu
45. image of a xenosaurus
46. unembedded journalist
47. steam turbogenerator
48. ichthyosis congenita
49. trilayer endometrium
50. phrenic nerve damage
51. autostereoscopic lcd
52. arundinaria auricoma
53. reserve at bankside
54. nanoclay properties
55. eurycoma longifolia
56. lippmann collection
57. city of herculaneum
58. halatuju pendidikan
59. untersuchung kinder
60. scytalidium species
61. folsom lake college
62. club shaped bacilli
63. westat rockville md
64. veer tejaji maharaj
65. arockv klang valley
66. ecomuseum montreal
67. la cucina italiana
68. tolson real estate
69. biffarius arenosus
70. collonil nanospray
71. nhprc funding 2008
72. steven windmueller
73. familien stammbaum
74. skyglobe for vista
75. geith attachments

76. tychonoff theorem
77. huitzuco guerrero
78. kahanamoku lagoon
79. ailanthus control
80. trackless trolley
81. 6x9 speaker boxes
82. hk celebrity news
83. plotosus lineatus
84. kingship of jesus
85. photo restoration
86. thom vollenweider
87. shepparton cinema
88. model and acting
89. ouray chalet inn
90. oedipus composer
91. guitar strumming
92. cornelia day spa
93. heslops michigan
94. prenzlauer allee
95. jasper maskelyne
96. jennifer lothrop
97. microgram symbol
98. rumex sanguineus
99. canicula summary
100. javelina hunting
101. drisdale lakeway
102. pro cycling gear
103. yonder gillihan
104. waneeta beckley
105. maidana vs khan
106. brisket recipes
107. cooking brisket
108. robert sturcken
109. pinnacle studio
110. joe morgenstern
111. john darer blog
112. vesid rochester
113. leonidas sparta
114. pridamiral pack
115. giovanna wheels
116. puyallup nissan
117. using a router
118. velveting beef
119. mamounia hotel
120. eric kapitulik
121. enterocytozoon
122. hiroshi nohara
123. scott sargeant
124. define pyrexia
125. rueben recipes
126. merkur beograd
127. dysplastic hip
128. long underslip
129. spurdog shark
130. creemers 1994

131. acbc anoka mn
132. tony hoagland
133. molokai ferry
134. gemma anscomb
135. sony dsc p120
136. osteoma cutis
137. dave tolleris
138. rueben carter
139. zoltan takacs
140. vyapam bhopal
141. metal metroid
142. that darn cat
143. coltman idaho
144. skincell 29a
145. nalini singh
146. cyp1b1 helix
147. pilbara iron
148. volusia mall
149. hanger opnet
150. mike culotta
151. gobbledygook
152. craig drezek
153. tukey method
154. bremner duke
155. artist kahlo
156. sfds houston
157. fishcode stf
158. ephim video
159. batata cafe
160. fukuda test
161. lovie smith
162. linuron red
163. quintum cms
164. interaction
165. bbfc search
166. john spratt
167. tek systems
168. vasa previa
169. normal jvp
170. deoxo unit
171. mike judge
172. hymap 2011
173. rtdc india
174. dr axelrod
175. mtp player
176. lila downs
177. john horan
178. tpx triton
179. alpha 3000
180. la fortuna
181. dr sisodia
182. 124e patch
183. sweat pea3
184. bti group
185. doris day

186. tupman ca
187. palm m130
188. rcn email
189. mama mias
190. cse india
191. meinhardt
192. parousia
193. mett tc
194. merz rv
195. 0in cdc
196. my sace
197. veitch
198. brca2
199. efta
200. ibm

B.2.3 Relevant Documents Per Query

The graphs in Figures B.3 and B.4 show the number of relevant documents per query in the set of 50 test queries (as determined by surrogate document relevance judgements), for the Title metadata and Title and Description metadata, respectively. Figures B.5 and B.6 show the same information, for the set of 200 test queries.

B.3 TREC Test Data

In this section we provide additional data associated with the TREC-based test data sets, discussed in Chapter 4.

B.3.1 Relevant Documents Per Query

Figure B.7 shows the number of relevant documents per query for the TREC-based test data sets. We observe that on average, the number of relevant documents per query is lower than that of the ITD and RTD sets. For the TREC-based data sets, document relevance judgements are made by human assessors, who may be more discerning than a programmatic approach.

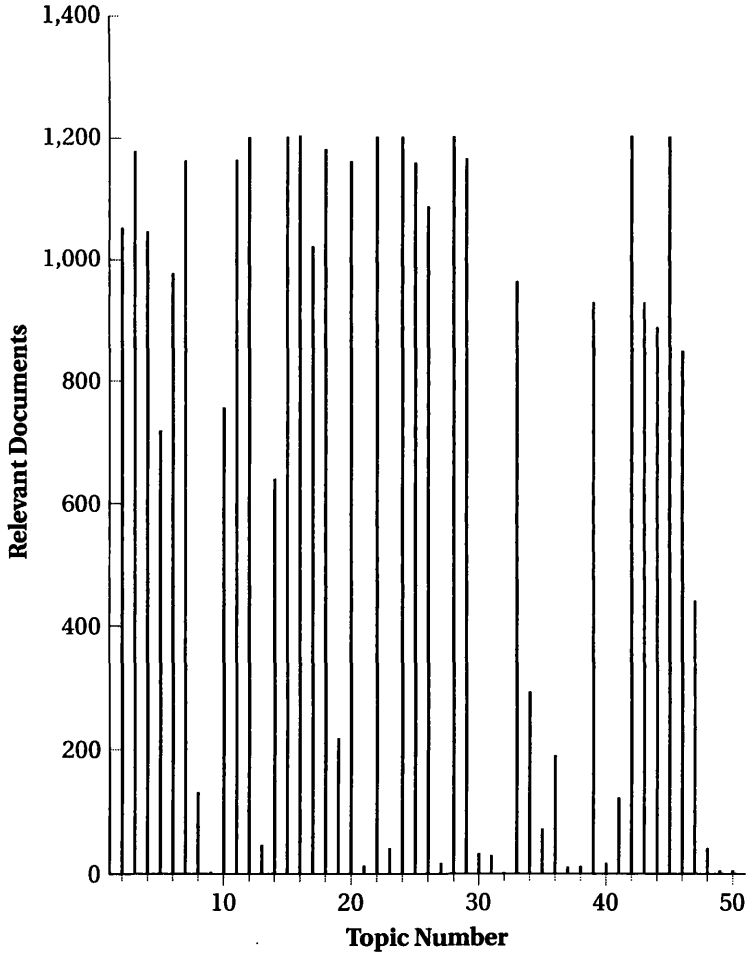


Figure B.3: Number of relevant documents (based on Title only data) for each RTD query, in the set of 50 queries.

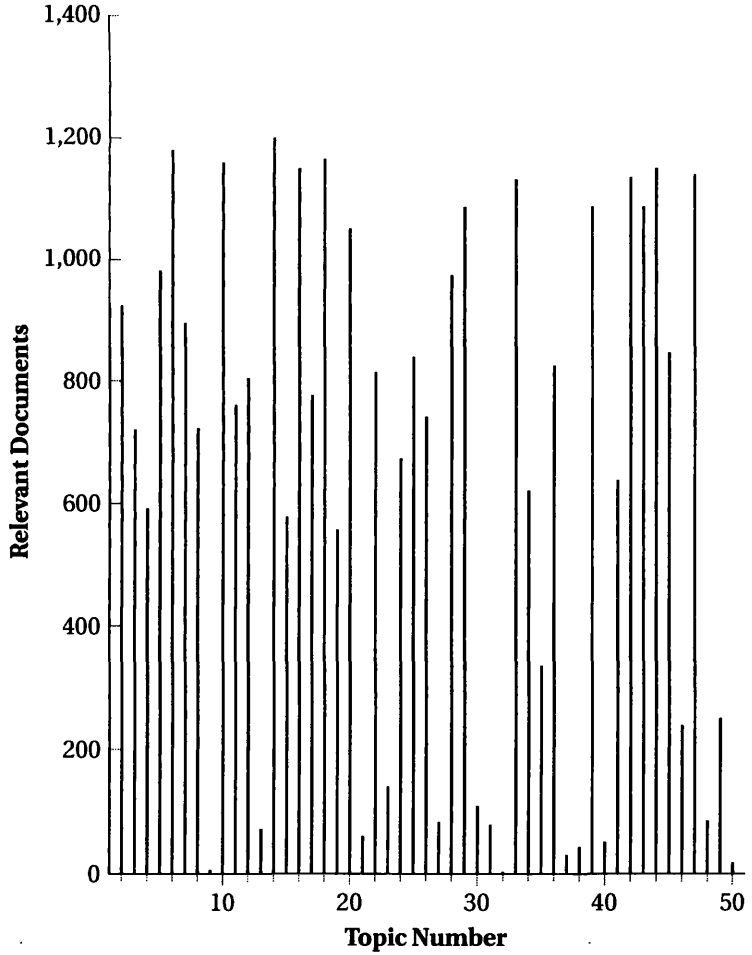


Figure B.4: Number of relevant documents (based on Title and Description data) for each RTD query, in the set of 50 queries.

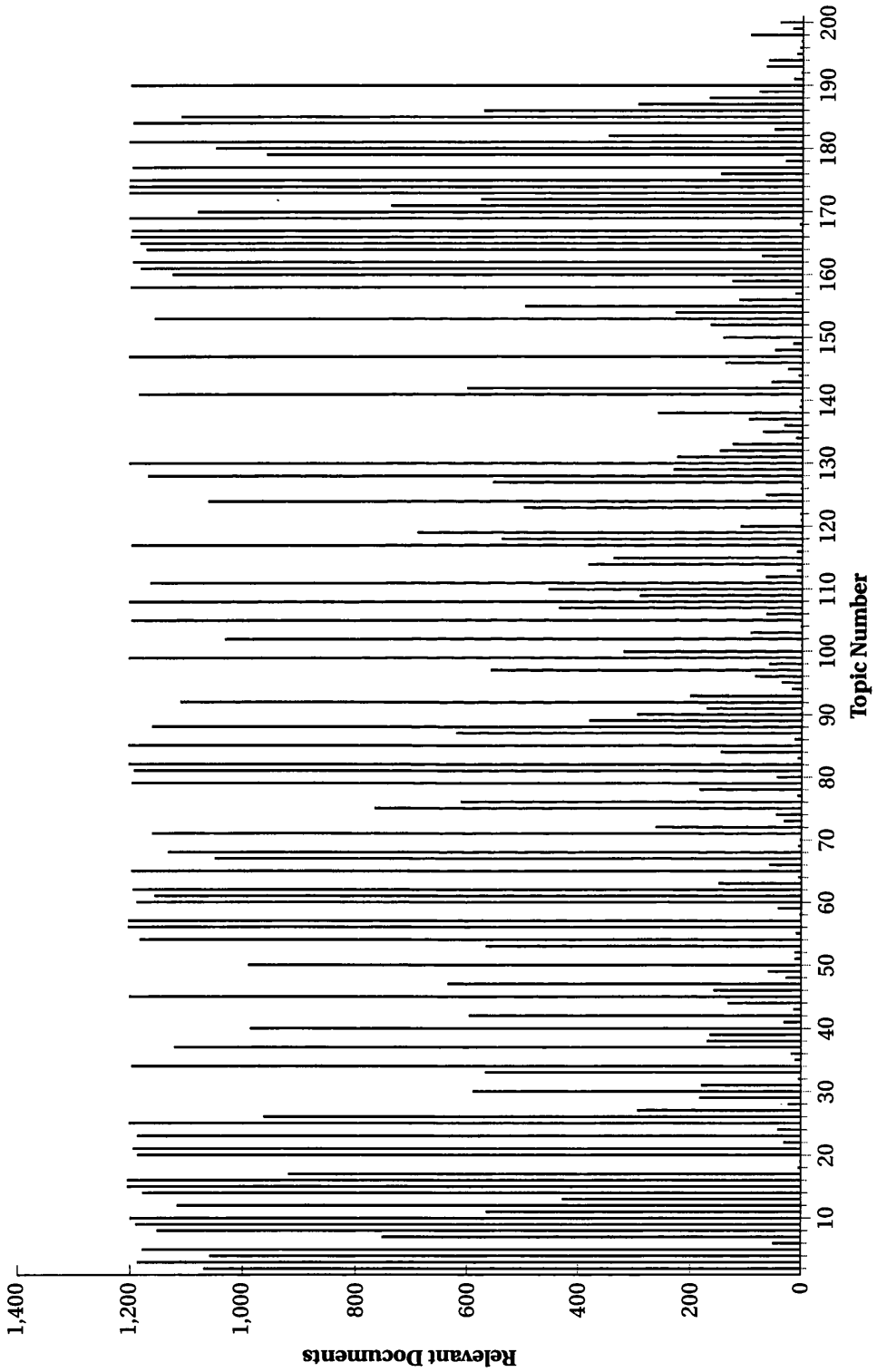


Figure B.5: Number of relevant documents (based on Title only data) for each RTD query, in the set of 200 queries.

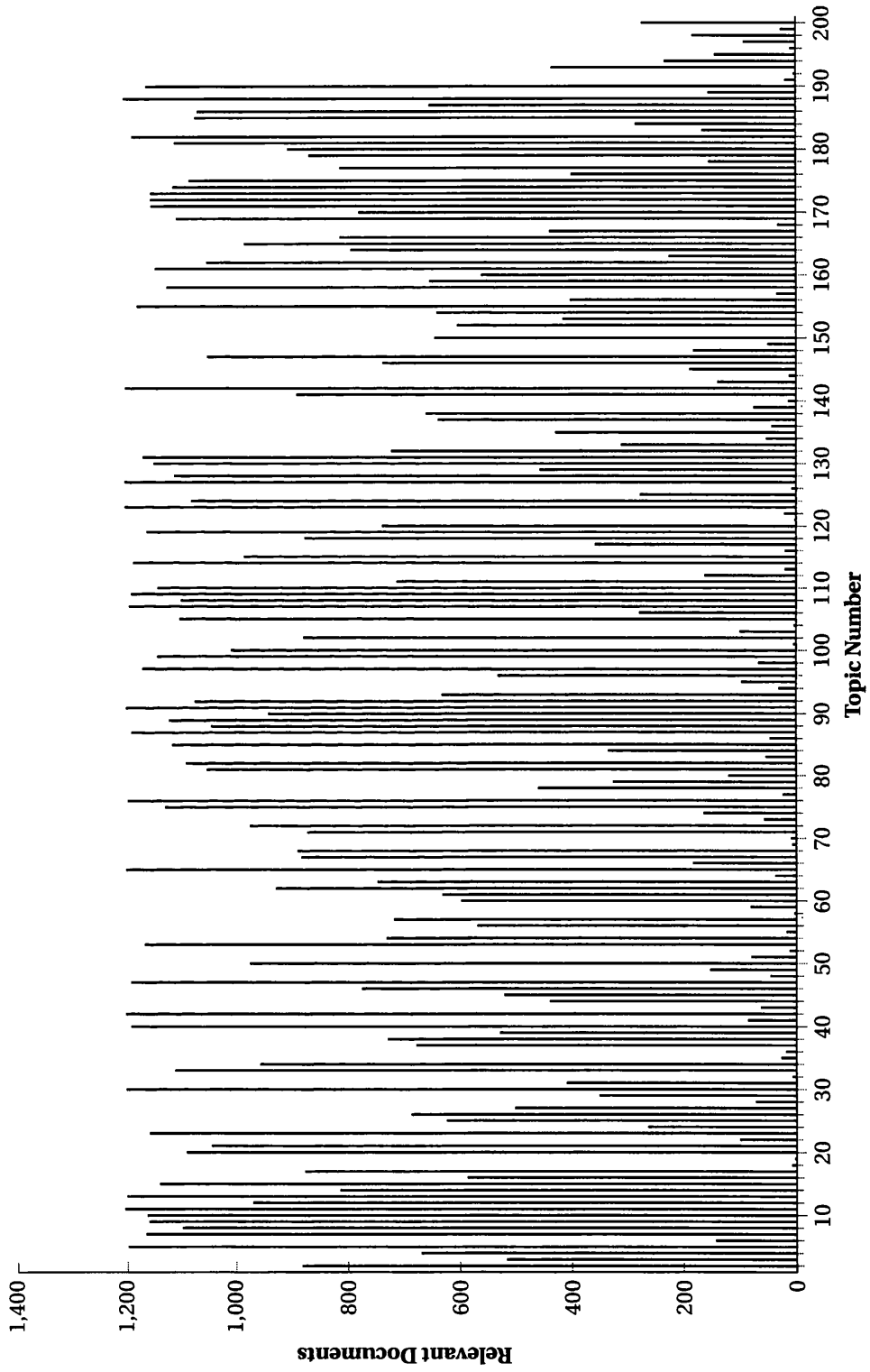


Figure B.6: Number of relevant documents (based on Title and Description data) for each RTD query, in the set of 200 queries.

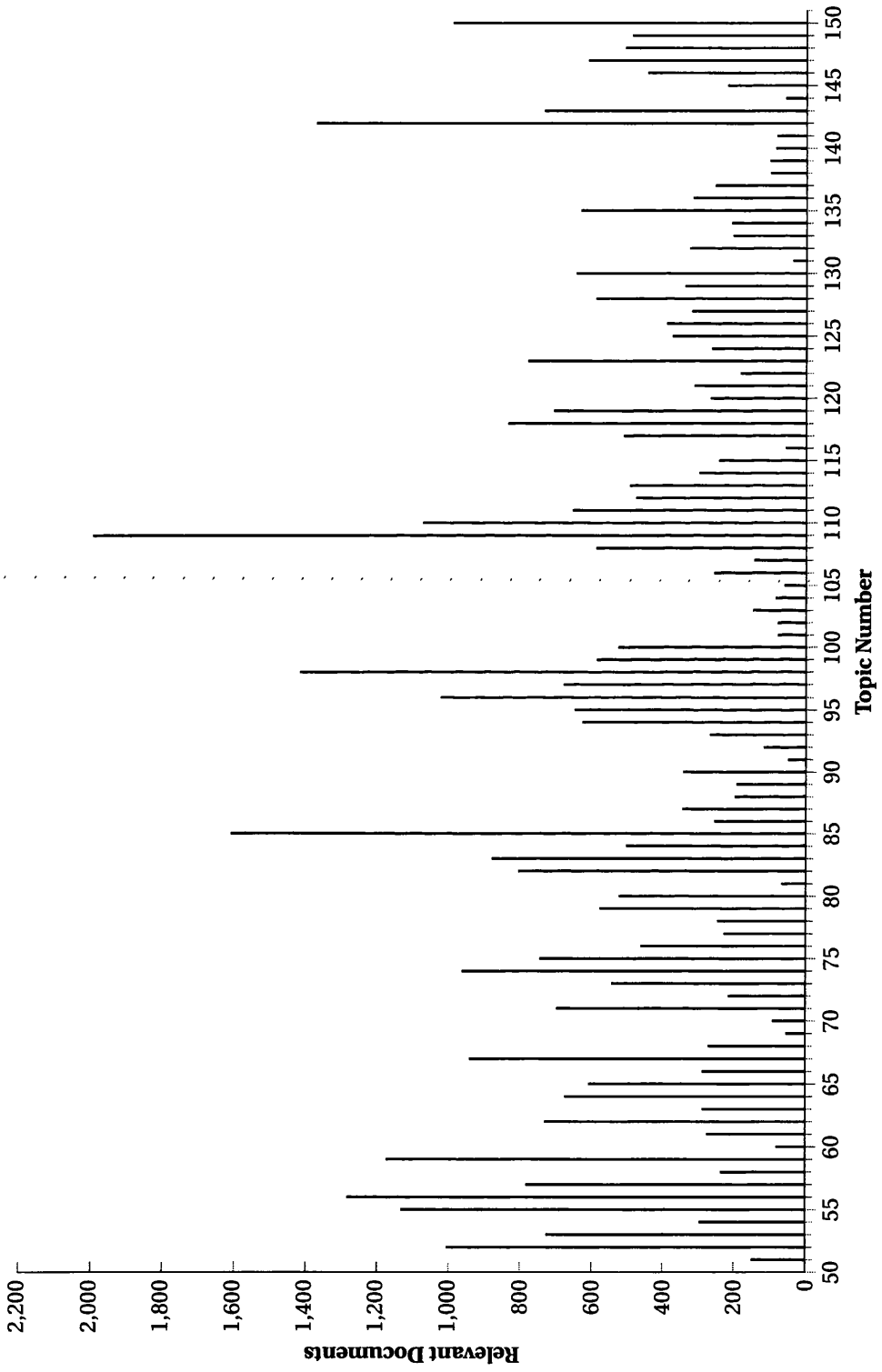


Figure B.7: Number of relevant documents for each TREC topic used.

Appendix C

Surrogate Relevance Judgements for Open Access Repository Test Data Sets

In Section 2.3.1 of Chapter 2, we described how collection selection algorithms can be evaluated in terms of how well they estimate an *optimal* ordering of collections. To generate an optimal collection ordering for a given query, we require knowledge of the documents relevant to that query: the document relevance judgements. When testing algorithms using the several TREC test data sets discussed in Section 4.1, generating the optimal collection ranking is straightforward: the TREC corpus comes with relevance judgements for each query. However, the open access repository data sets, described in Section 4.2, are built from real, operational repositories; as such, we do not have relevance judgements for the documents and queries within these data sets.

One method for creating relevance judgements for a set of documents and queries is to have a human assessor evaluate the relevance of every document, for every query. This is impractical for our open access repository data sets, due to their magnitude. We therefore investigate several strategies for programmatically generating *surrogate* document relevance judgements for the open access repository data sets. The following sections describe these strategies and discuss our empirical results.

C.1 Experiment Set-up

We present and investigate several possible strategies for generating surrogate relevance judgements for our open access repository data sets, some of which draw on the TREC approach to compiling relevance judgements, known as pooling.

In the pooling approach, the document ranking systems participating in TREC submit their top n relevant documents for each query, into a pool. A human assessor then judges each document in the pool for relevance to the query [25]. This technique is less intensive for the human assessors than evaluating every document for every query.

However, we pursue an entirely automated approach to producing relevance judgements for our test data sets: given the size of our data sets, assessing even a pooled sample

of documents for every query would still be very labour intensive.

For our experiment to find an optimal programmatic method for generating document relevance judgements, we utilise three document ranking algorithms – the Lucene search algorithm [15], BM25 [42] and *tf-idf*– in seven different configurations, described below:

Algorithm agreement: A document is classed as relevant if all three algorithms agree it is relevant.

Majority voting: A document is classed as relevant if two or more algorithms agree it is relevant.

Pool all: All documents returned by each algorithm are classed as relevant.

Lucene only: Relevant documents are determined by the Lucene algorithm only.

BM25 only: Relevant documents are determined by the BM25 algorithm only.

***tf-idf* only:** Relevant documents are determined by the *tf-idf* algorithm only.

Lucene and BM25 (Agreement): A document is classed as relevant if both the Lucene and BM25 algorithms agree it is relevant.

Lucene and BM25 (Pool): All documents returned by the Lucene and BM25 algorithms are classed as relevant.

To evaluate the suitability of these configurations for producing document relevance judgements, we test their ability to generate the relevance judgements associated with the TREC corpus. We execute TREC topics 51-150, as these topics include relevance judgements for all documents in the corpus. We formulate the topics as both short and long queries (refer back to Section 4.1.7 in Chapter 4 for details).

We use the Apache Lucene[1] search engine library to build a document index for the TREC corpus. When executing a search, each algorithm returns the top n relevant documents; where n is 0.1% of the total number of documents in the corpus (for a collection containing one million documents, up to one thousand relevant documents would be returned).

C.2 Results

For each strategy for generating surrogate relevance judgements, we calculate the precision and recall achieved on each query. The precision value shows the proportion of documents classified by the strategy as relevant, that are also relevant according to TREC. The recall value represents the proportion of documents deemed relevant by TREC, that were classified as relevant by the given configuration. To give a summary of performance, we average the precision and recall values over all queries; Table C.1 presents our results.

[1] `http://lucene.apache.org/`

Table C.1: Precision and recall values for each strategy for generating surrogate relevance judgements, averaged over all queries.

	Short Queries		Long Queries	
	Precision	**Recall**	**Precision**	**Recall**
Algorithm agreement	0.27	0.19	0.33	0.22
Majority voting	0.17	0.32	0.23	0.45
Pool all	0.10	0.40	0.14	0.55
Lucene only	0.15	0.32	0.21	0.47
BM25 only	0.15	0.33	0.21	0.47
TF-IDF only	0.12	0.26	0.13	0.28
Lucene and BM25 (Agreement)	0.17	0.29	0.24	0.43
Lucene and BM25 (Pool)	0.13	0.36	0.19	0.51

We observe that the algorithm agreement strategy exhibits the highest average precision, but also the lowest average recall. In contrast, pooling all documents gives the highest average recall, but with very poor precision. The majority voting method sits in the middle of these two. Thus, it appears that by using more document ranking algorithms to scrutinise the relevance of document, we become more certain of its relevance; however, this also leads us to sacrifice some relevant documents.

Looking at the performance of the three algorithms individually, we see that Lucene and BM25 perform similarly, with *tf-idf* performing much worse. Indeed, if we compare the performance of algorithm agreement with Lucene and BM25 (Agreement), we see that algorithm agreement has better precision, but worse recall than Lucene and BM25 (Agreement). This suggests that while *tf-idf* helps to confirm the relevance of some documents, it may result in some relevant documents being thrown away. The reverse is the case when comparing pool all with Lucene and BM25 (Pool); pooling all relevant documents from each algorithm has worse precision, but better recall than Lucene and BM25 (Pool). This suggests that while *tf-idf* offers some relevant documents that Lucene and BM25 did not, some of these documents are not actually relevant according to TREC.

The most balanced configurations, in terms of the trade-off between precision and recall, are majority voting, and Lucene and BM25 (Agreement). Since *tf-idf* shows poor performance, it is unlikely to add much to the majority voting configuration. We therefore suggest that Lucene and BM25 (Agreement) is the most appropriate strategy to use to generate surrogate document relevance judgements, for our two open access repository test data sets.

C.3 Summary

Following our experiments to investigate the performance of several programmatic strategies for generating surrogate relevance judgements, we produce such relevance judgements for our open access repository test data sets in the following way: a document index for the

harvested metadata is built using the Apache Lucene search engine library. For each query, all documents in a test data set are ranked using BM25, and the Lucene search algorithm; each returning the top n relevant documents (where n is 0.1% of the total number of harvested documents). A list of surrogate relevance judgements is generated by taking the intersect of the relevant documents from the two algorithms. Thus, a document is deemed relevant if both algorithms agree it is relevant.

From this list, we can determine the number of relevant documents in each collection, and thus generate an optimal ranking as per the method described in Section 3.2.

Appendix D

Algorithm Performance Scores

In this chapter we provide tables of complete performance scores for all implemented algorithms, on all test data sets. Therefore, performance scores not discussed in the main body of this thesis can be found here.

For each data set, we provide the following tables:

- Spearman rank correlation scores (giving correlation with both FsBR and SBR);
- Z-Test results, indicating whether any algorithm shows a performance significantly different from any other;
- Blest and Da Costa weighted rank correlation scores (giving correlation with FsBR);
- $\mathcal{R}_n$ scores at selected values of n;
- $\hat{\mathcal{R}}_n$ scores at selected values of n;
- $\mathcal{P}_n$ scores at selected values of n; and
- Precision@5 and Correct@1 scores.

D.1 Initial Test Data (ITD)

Table D.1: ITD average Spearman rank correlations, for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		**SBR**	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
Doddle	**0.82**	**0.72**	0.42	0.29
Doddle_RC	**0.80**	**0.72**	0.32	0.23
Doddle_RP	**0.80**	**0.62**	0.33	0.10
Doddle_RF	**0.77**	**0.69**	**0.70**	**0.54**
Doddle_RC+RP	**0.80**	**0.68**	0.33	0.16
Doddle_RC+RF	**0.82**	**0.75**	**0.46**	0.38
Doddle_RP+RF	**0.82**	**0.69**	**0.47**	0.31
Doddle_×	**0.77**	**0.64**	0.30	0.12
Doddle_RC×RP	**0.77**	**0.64**	0.29	0.10
Doddle_RC×RF	**0.80**	**0.71**	0.35	0.26
Doddle_RP×RF	**0.80**	**0.65**	0.35	0.14
Doddle_W	**0.81**	**0.71**	0.39	0.23
SBR	***0.51***	***0.48***	—	—

Table D.2: ITD average Spearman rank correlations, for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		**SBR**	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
bGlOSS	**0.69**	**0.70**	**0.58**	**0.53**
CORI	**0.81**	**0.78**	**0.72**	**0.73**
Cosine Measure	**0.76**	**0.73**	**0.70**	**0.64**
Inner Product	**0.79**	**0.75**	**0.78**	**0.78**
Skew	**0.80**	**0.74**	**0.77**	**0.78**
Highest-available Similarity	**0.78**	**0.67**	**0.80**	**0.85**
CVV	**0.79**	**0.72**	**0.78**	**0.80**
DFPROP	**0.80**	**0.74**	**0.77**	**0.79**
Distribution of Informative Amt	0.21	−0.02	0.17	−0.04
SCS	**0.59**	**0.45**	**0.79**	**0.70**
AvICTF	**0.60**	**0.46**	**0.79**	**0.70**
NSCQ	**0.72**	**0.64**	**0.85**	**0.88**
Doddle	**0.82**	**0.72**	0.42	0.29
SBR	***0.51***	***0.48***	—	—

Table D.3: ITD Z-Test results, showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Title data.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-availaable Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AvICTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
Algorithms																									
bGlOSS									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CORI									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Cosine Measure									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Inner Product									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Skew									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Highest-availaable Similarity									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CVV									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DFPROP									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Distribution of Informative Amt	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓												
SCS									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
AvICTF									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NSCQ									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RP									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RF									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC+RP									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC+RF									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RP+RF									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_×									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC×RP									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC×RF									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RP×RF									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_W									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SBR									✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table D.4: ITD Z-Test results, showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Title and Description data.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AvICTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
bGlOSS									✓																
CORI									✓																
Cosine Measure									✓																
Inner Product									✓																
Skew									✓																
Highest-available Similarity									✓																
CVV									✓																
DFPROP									✓																
Distribution of Informative Amt	✓	✓	✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SCS																									
AvICTF																									
NSCQ									✓																
Doddle									✓																
Doddle_RC									✓																
Doddle_RP									✓																
Doddle_RF									✓																
Doddle_RC+RP									✓																
Doddle_RC+RF									✓																
Doddle_RP+RF									✓																
Doddle_×									✓																
Doddle_RC×RP									✓																
Doddle_RC×RF									✓																
Doddle_RP×RF									✓																
Doddle_W									✓																
SBR									✓																

Table D.5: ITD average Blest and Da Costa weighted rank correlations, for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR.

	Blest		**Da Costa**	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
Doddle	**0.84**	**0.75**	**0.81**	**0.72**
Doddle_RC	**0.82**	**0.76**	**0.79**	**0.73**
Doddle_RP	**0.83**	**0.67**	**0.80**	**0.64**
Doddle_RF	**0.77**	**0.71**	**0.75**	**0.68**
Doddle_RC+RP	**0.83**	**0.73**	**0.80**	**0.69**
Doddle_RC+RF	**0.84**	**0.77**	**0.82**	**0.75**
Doddle_RP+RF	**0.84**	**0.72**	**0.82**	**0.70**
Doddle_×	**0.80**	**0.68**	**0.77**	**0.65**
Doddle_RC×RP	**0.80**	**0.68**	**0.77**	**0.65**
Doddle_RC×RF	**0.83**	**0.75**	**0.80**	**0.72**
Doddle_RP×RF	**0.83**	**0.69**	**0.80**	**0.66**
Doddle_W	**0.84**	**0.75**	**0.81**	**0.72**
SBR	***0.50***	***0.46***	***0.48***	***0.44***

Table D.6: ITD average Blest and Da Costa weighted rank correlations, for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR.

	Blest		**Da Costa**	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
bGlOSS	**0.70**	**0.74**	**0.73**	**0.73**
CORI	**0.81**	**0.78**	**0.80**	**0.77**
Cosine Measure	**0.75**	**0.73**	**0.74**	**0.72**
Inner Product	**0.78**	**0.74**	**0.77**	**0.72**
Skew	**0.79**	**0.74**	**0.78**	**0.72**
Highest-available Similarity	**0.77**	**0.66**	**0.75**	**0.63**
CVV	**0.78**	**0.71**	**0.76**	**0.69**
DFPROP	**0.79**	**0.74**	**0.78**	**0.72**
Distribution of Informative Amt	0.25	0.02	0.24	0.02
SCS	**0.56**	0.42	**0.54**	0.39
AvICTF	**0.57**	0.43	**0.55**	0.40
NSCQ	**0.70**	**0.62**	**0.68**	**0.60**
Doddle	**0.84**	**0.75**	**0.81**	**0.72**
SBR	***0.50***	***0.46***	***0.48***	***0.44***

Table D.7: ITD average $\mathcal{R}_n$ scores, for the various configurations of the Doddle algorithm, executed over Title data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.75	0.76	0.76	0.50	0.77	0.78	0.76	0.73	0.71	0.75	0.76	0.75	*0.33*
2	0.79	0.80	0.80	0.59	0.80	0.80	0.81	0.77	0.75	0.77	0.76	0.80	*0.20*
3	0.82	0.83	0.82	0.68	0.83	0.83	0.82	0.80	0.81	0.82	0.80	0.82	*0.31*
4	0.86	0.86	0.85	0.77	0.85	0.87	0.86	0.86	0.86	0.86	0.85	0.86	*0.45*
5	0.89	0.89	0.88	0.82	0.89	0.89	0.88	0.87	0.87	0.89	0.88	0.89	*0.52*
6	0.91	0.90	0.90	0.86	0.90	0.91	0.91	0.89	0.89	0.91	0.90	0.91	*0.66*
7	0.93	0.93	0.93	0.89	0.93	0.93	0.93	0.91	0.92	0.93	0.93	0.93	*0.75*
8	0.95	0.95	0.94	0.91	0.95	0.94	0.95	0.93	0.93	0.94	0.95	0.94	*0.80*
9	0.96	0.96	0.96	0.93	0.97	0.96	0.96	0.95	0.95	0.97	0.97	0.96	*0.88*
10	0.97	0.98	0.98	0.95	0.98	0.97	0.97	0.97	0.97	0.98	0.98	0.98	*0.91*
11	0.98	0.98	0.98	0.96	0.98	0.98	0.98	0.98	0.98	0.99	0.98	0.99	*0.93*
12	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	*0.96*
13	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.98*
14	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
15	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
16	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.8: ITD average $\mathcal{R}_n$ scores, for existing algorithms and Doddle, executed over Title data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.66	0.68	0.50	0.40	0.40	0.38	0.41	0.40	0.11	0.10	0.10	0.35	0.75	*0.33*
2	0.71	0.69	0.57	0.57	0.59	0.57	0.54	0.58	0.16	0.19	0.20	0.41	0.79	*0.20*
3	0.77	0.75	0.66	0.68	0.68	0.68	0.62	0.68	0.24	0.32	0.33	0.53	0.82	*0.31*
4	0.81	0.82	0.77	0.77	0.77	0.75	0.77	0.77	0.29	0.45	0.49	0.61	0.86	*0.45*
5	0.84	0.86	0.81	0.84	0.85	0.83	0.84	0.86	0.39	0.56	0.57	0.74	0.89	*0.52*
6	0.86	0.88	0.84	0.88	0.90	0.86	0.90	0.90	0.46	0.65	0.65	0.77	0.91	*0.66*
7	0.85	0.91	0.88	0.92	0.93	0.90	0.93	0.93	0.51	0.74	0.75	0.86	0.93	*0.75*
8	0.87	0.93	0.89	0.95	0.94	0.92	0.95	0.94	0.57	0.81	0.81	0.90	0.95	*0.80*
9	0.89	0.95	0.93	0.96	0.96	0.95	0.96	0.96	0.62	0.87	0.87	0.93	0.96	*0.88*
10	0.91	0.96	0.95	0.97	0.98	0.98	0.98	0.98	0.69	0.92	0.93	0.96	0.97	*0.91*
11	0.93	0.97	0.97	0.98	0.99	0.99	0.99	0.99	0.77	0.94	0.95	0.98	0.98	*0.93*
12	0.93	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.84	0.96	0.96	0.99	0.99	*0.96*
13	0.94	1.00	0.99	1.00	1.00	1.00	1.00	1.00	0.93	0.99	0.99	0.99	1.00	*0.98*
14	0.95	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	1.00	1.00	1.00	1.00	*1.00*
15	0.96	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	*1.00*
16	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.9: ITD average $\mathcal{R}_n$ scores, for the various configurations of the Doddle algorithm, executed over Title and Description data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.71	0.77	0.68	0.45	0.71	0.75	0.68	0.66	0.71	0.69	0.69	0.71	*0.32*
2	0.75	0.78	0.68	0.55	0.74	0.81	0.72	0.73	0.69	0.77	0.70	0.76	*0.20*
3	0.81	0.82	0.74	0.63	0.77	0.82	0.74	0.76	0.75	0.81	0.76	0.79	*0.27*
4	0.83	0.85	0.77	0.71	0.83	0.84	0.77	0.81	0.82	0.82	0.78	0.83	*0.40*
5	0.85	0.87	0.82	0.76	0.85	0.86	0.83	0.83	0.83	0.85	0.81	0.85	*0.47*
6	0.87	0.87	0.84	0.81	0.86	0.88	0.86	0.83	0.83	0.86	0.83	0.88	*0.64*
7	0.89	0.91	0.84	0.85	0.88	0.90	0.88	0.86	0.85	0.90	0.86	0.89	*0.73*
8	0.91	0.92	0.86	0.88	0.90	0.92	0.89	0.88	0.89	0.92	0.87	0.91	*0.79*
9	0.93	0.94	0.89	0.92	0.92	0.94	0.91	0.90	0.91	0.94	0.89	0.92	*0.88*
10	0.95	0.95	0.91	0.93	0.95	0.96	0.93	0.94	0.93	0.95	0.92	0.95	*0.91*
11	0.97	0.97	0.94	0.96	0.96	0.97	0.96	0.95	0.95	0.98	0.93	0.97	*0.93*
12	0.98	0.98	0.96	0.98	0.98	0.99	0.97	0.97	0.97	0.98	0.97	0.98	*0.97*
13	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.98	0.98	0.99	0.99	0.99	*0.97*
14	1.00	1.00	0.99	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.99	1.00	*0.99*
15	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.99*
16	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.10: ITD average $\mathcal{R}_n$ scores, for existing algorithms and Doddle, executed over Title and Description data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.69	0.72	0.54	0.40	0.40	0.30	0.39	0.40	0.10	0.08	0.09	0.34	0.71	*0.32*
2	0.69	0.68	0.63	0.51	0.53	0.41	0.42	0.53	0.13	0.16	0.17	0.32	0.75	*0.20*
3	0.78	0.74	0.70	0.65	0.65	0.53	0.54	0.65	0.17	0.27	0.28	0.41	0.81	*0.27*
4	0.82	0.81	0.76	0.75	0.74	0.61	0.72	0.74	0.21	0.37	0.38	0.51	0.83	*0.40*
5	0.85	0.84	0.82	0.83	0.81	0.70	0.81	0.82	0.24	0.49	0.49	0.69	0.85	*0.47*
6	0.88	0.87	0.85	0.88	0.87	0.75	0.85	0.87	0.31	0.56	0.56	0.75	0.87	*0.64*
7	0.89	0.90	0.88	0.90	0.89	0.85	0.89	0.90	0.38	0.61	0.61	0.83	0.89	*0.73*
8	0.91	0.93	0.88	0.94	0.92	0.91	0.91	0.92	0.45	0.65	0.66	0.88	0.91	*0.79*
9	0.92	0.95	0.92	0.95	0.95	0.94	0.94	0.95	0.52	0.71	0.71	0.91	0.93	*0.88*
10	0.95	0.97	0.95	0.97	0.97	0.97	0.96	0.97	0.58	0.79	0.79	0.93	0.95	*0.91*
11	0.96	0.97	0.97	0.98	0.98	0.98	0.97	0.98	0.65	0.89	0.89	0.96	0.97	*0.93*
12	0.97	0.99	0.98	0.99	0.99	0.99	0.98	0.99	0.75	0.92	0.92	0.98	0.98	*0.97*
13	0.98	1.00	0.99	1.00	0.99	1.00	0.99	0.99	0.85	0.99	0.99	0.99	0.99	*0.97*
14	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.93	1.00	1.00	0.99	1.00	*0.99*
15	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97	1.00	1.00	1.00	1.00	*0.99*
16	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.11: ITD average $\hat{\mathcal{R}}_n$ scores, for the various configurations of the Doddle algorithm, executed over Title data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.30	0.30	0.30	0.19	0.30	0.32	0.30	0.29	0.28	0.30	0.30	0.30	*0.12*
2	0.48	0.48	0.48	0.36	0.48	0.48	0.49	0.46	0.46	0.47	0.46	0.48	*0.12*
3	0.60	0.60	0.60	0.49	0.60	0.60	0.60	0.58	0.59	0.60	0.58	0.60	*0.22*
4	0.70	0.69	0.69	0.62	0.69	0.70	0.70	0.69	0.69	0.69	0.69	0.69	*0.37*
5	0.77	0.77	0.77	0.72	0.77	0.77	0.77	0.75	0.76	0.77	0.77	0.78	*0.45*
6	0.83	0.83	0.83	0.79	0.83	0.83	0.83	0.82	0.82	0.83	0.83	0.83	*0.61*
7	0.88	0.88	0.88	0.84	0.88	0.88	0.89	0.87	0.87	0.88	0.88	0.88	*0.71*
8	0.92	0.92	0.92	0.88	0.92	0.91	0.92	0.90	0.90	0.91	0.92	0.92	*0.77*
9	0.95	0.95	0.95	0.92	0.95	0.94	0.94	0.94	0.93	0.95	0.95	0.95	*0.86*
10	0.96	0.97	0.97	0.94	0.97	0.96	0.96	0.96	0.96	0.97	0.97	0.97	*0.90*
11	0.98	0.98	0.98	0.96	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	*0.93*
12	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	*0.96*
13	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.98*
14	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
15	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
16	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.12: ITD average $\hat{\mathcal{R}}_n$ scores, for existing algorithms and Doddle, executed over Title data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.26	0.27	0.20	0.17	0.17	0.16	0.17	0.17	0.04	0.03	0.03	0.14	0.30	*0.12*
2	0.42	0.42	0.35	0.36	0.37	0.36	0.33	0.36	0.08	0.12	0.12	0.24	0.48	*0.12*
3	0.56	0.56	0.48	0.50	0.50	0.50	0.46	0.50	0.17	0.23	0.24	0.39	0.60	*0.22*
4	0.66	0.67	0.63	0.63	0.63	0.61	0.63	0.63	0.23	0.36	0.40	0.50	0.70	*0.37*
5	0.73	0.75	0.71	0.73	0.74	0.72	0.73	0.75	0.34	0.50	0.50	0.65	0.77	*0.45*
6	0.79	0.80	0.77	0.81	0.82	0.79	0.82	0.82	0.43	0.60	0.60	0.71	0.83	*0.61*
7	0.81	0.86	0.83	0.87	0.88	0.85	0.88	0.88	0.49	0.71	0.72	0.82	0.88	*0.71*
8	0.85	0.90	0.87	0.92	0.91	0.90	0.92	0.91	0.56	0.79	0.79	0.87	0.92	*0.77*
9	0.87	0.94	0.92	0.95	0.95	0.94	0.95	0.95	0.61	0.86	0.86	0.92	0.95	*0.86*
10	0.91	0.95	0.94	0.96	0.97	0.97	0.97	0.97	0.69	0.92	0.92	0.96	0.96	*0.90*
11	0.92	0.97	0.96	0.98	0.98	0.99	0.99	0.98	0.77	0.94	0.95	0.98	0.98	*0.93*
12	0.93	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.84	0.96	0.96	0.98	0.99	*0.96*
13	0.94	1.00	0.99	1.00	1.00	1.00	1.00	1.00	0.93	0.99	0.99	0.99	1.00	*0.98*
14	0.95	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	1.00	1.00	1.00	1.00	*1.00*
15	0.96	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	*1.00*
16	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.13: ITD average $\hat{\mathcal{R}}_n$ scores, for the various configurations of the Doddle algorithm, executed over Title and Description data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.28	0.31	0.27	0.17	0.28	0.30	0.26	0.26	0.28	0.28	0.27	0.28	*0.12*
2	0.46	0.48	0.41	0.32	0.45	0.49	0.44	0.45	0.43	0.47	0.42	0.46	*0.12*
3	0.60	0.61	0.55	0.46	0.57	0.60	0.54	0.57	0.56	0.60	0.56	0.58	*0.20*
4	0.68	0.69	0.63	0.58	0.68	0.68	0.63	0.67	0.67	0.67	0.64	0.68	*0.33*
5	0.74	0.76	0.72	0.67	0.75	0.76	0.73	0.73	0.73	0.75	0.71	0.75	*0.41*
6	0.80	0.80	0.77	0.75	0.79	0.81	0.79	0.77	0.76	0.79	0.77	0.81	*0.59*
7	0.85	0.86	0.80	0.81	0.84	0.85	0.83	0.81	0.81	0.86	0.82	0.84	*0.69*
8	0.88	0.89	0.83	0.85	0.87	0.89	0.86	0.85	0.86	0.89	0.84	0.88	*0.76*
9	0.91	0.92	0.88	0.90	0.90	0.92	0.90	0.89	0.90	0.92	0.88	0.91	*0.86*
10	0.94	0.95	0.90	0.92	0.94	0.95	0.92	0.93	0.92	0.95	0.91	0.94	*0.90*
11	0.96	0.97	0.93	0.96	0.96	0.97	0.96	0.95	0.95	0.97	0.93	0.97	*0.93*
12	0.98	0.98	0.96	0.98	0.98	0.99	0.97	0.97	0.97	0.98	0.97	0.98	*0.96*
13	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.98	0.98	0.99	0.99	0.99	*0.97*
14	1.00	1.00	0.99	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.99	1.00	*0.99*
15	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.99*
16	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.14: ITD average $\hat{\mathcal{R}}_n$ scores, for existing algorithms and Doddle, executed over Title and Description data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.27	0.28	0.20	0.16	0.16	0.12	0.15	0.16	0.03	0.03	0.03	0.14	0.28	*0.12*
2	0.41	0.41	0.39	0.32	0.33	0.26	0.26	0.33	0.07	0.10	0.11	0.20	0.46	*0.12*
3	0.57	0.55	0.52	0.48	0.48	0.39	0.40	0.48	0.12	0.20	0.21	0.31	0.60	*0.20*
4	0.67	0.66	0.62	0.62	0.61	0.51	0.60	0.60	0.17	0.31	0.31	0.43	0.68	*0.33*
5	0.75	0.74	0.73	0.73	0.72	0.62	0.71	0.72	0.21	0.44	0.44	0.61	0.74	*0.41*
6	0.81	0.80	0.79	0.81	0.80	0.69	0.79	0.80	0.28	0.51	0.51	0.70	0.80	*0.59*
7	0.84	0.85	0.83	0.86	0.85	0.81	0.84	0.85	0.36	0.58	0.58	0.79	0.85	*0.69*
8	0.88	0.90	0.86	0.91	0.90	0.89	0.88	0.90	0.44	0.64	0.64	0.85	0.88	*0.76*
9	0.90	0.94	0.90	0.93	0.94	0.93	0.92	0.93	0.52	0.70	0.70	0.90	0.91	*0.86*
10	0.94	0.96	0.94	0.96	0.96	0.96	0.95	0.96	0.57	0.78	0.78	0.93	0.94	*0.90*
11	0.96	0.97	0.96	0.98	0.98	0.98	0.97	0.98	0.65	0.89	0.89	0.95	0.96	*0.93*
12	0.97	0.99	0.98	0.99	0.99	0.99	0.98	0.99	0.74	0.92	0.92	0.98	0.98	*0.96*
13	0.98	1.00	0.99	1.00	0.99	1.00	0.99	0.99	0.85	0.99	0.99	0.99	0.99	*0.97*
14	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.93	1.00	1.00	0.99	1.00	*0.99*
15	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97	1.00	1.00	1.00	1.00	*0.99*
16	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.15: ITD average $\mathcal{P}_n$ scores, for the various configurations of the Doddle algorithm, executed over Title data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.98	0.94	0.98	1.00	0.98	0.98	0.98	0.94	0.92	0.96	0.98	0.98	*0.98*
2	0.96	0.95	0.95	1.00	0.95	0.97	0.98	0.91	0.90	0.95	0.94	0.95	*0.82*
3	0.93	0.91	0.93	0.99	0.91	0.95	0.96	0.90	0.90	0.93	0.93	0.93	*0.87*
4	0.92	0.90	0.90	0.97	0.90	0.94	0.95	0.90	0.90	0.91	0.92	0.91	*0.91*
5	0.90	0.89	0.90	0.94	0.89	0.92	0.92	0.88	0.88	0.90	0.90	0.90	*0.85*
6	0.89	0.88	0.87	0.92	0.88	0.90	0.90	0.85	0.85	0.89	0.88	0.88	*0.86*
7	0.86	0.84	0.85	0.89	0.85	0.87	0.87	0.83	0.83	0.85	0.86	0.85	*0.85*
8	0.84	0.81	0.82	0.86	0.82	0.84	0.84	0.80	0.80	0.82	0.83	0.83	*0.83*
9	0.81	0.78	0.79	0.82	0.79	0.81	0.81	0.77	0.77	0.79	0.80	0.80	*0.81*
10	0.77	0.76	0.75	0.79	0.75	0.78	0.78	0.75	0.75	0.76	0.76	0.77	*0.76*
11	0.74	0.72	0.72	0.75	0.72	0.74	0.74	0.72	0.72	0.72	0.73	0.73	*0.73*
12	0.70	0.70	0.70	0.72	0.70	0.71	0.71	0.70	0.69	0.70	0.69	0.70	*0.70*
13	0.67	0.67	0.67	0.68	0.67	0.68	0.68	0.67	0.67	0.67	0.67	0.67	*0.66*
14	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64	*0.64*
15	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	*0.60*
16	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	*0.57*

Table D.16: ITD average $\mathcal{P}_n$ scores, for existing algorithms and Doddle, executed over Title data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	1.00	1.00	0.96	1.00	1.00	1.00	1.00	1.00	0.64	0.92	0.92	1.00	0.98	*0.98*
2	0.98	1.00	0.98	1.00	1.00	1.00	1.00	1.00	0.67	0.92	0.93	1.00	0.96	*0.82*
3	0.97	0.99	0.97	0.99	0.99	0.99	0.99	0.99	0.67	0.93	0.93	0.99	0.93	*0.87*
4	0.94	0.98	0.98	0.98	0.98	0.98	0.97	0.98	0.67	0.93	0.94	0.97	0.92	*0.91*
5	0.90	0.95	0.95	0.96	0.95	0.94	0.95	0.95	0.70	0.92	0.92	0.94	0.90	*0.85*
6	0.86	0.92	0.92	0.93	0.93	0.92	0.93	0.93	0.70	0.90	0.90	0.91	0.89	*0.86*
7	0.83	0.89	0.89	0.89	0.89	0.89	0.89	0.89	0.69	0.88	0.88	0.88	0.86	*0.85*
8	0.81	0.86	0.86	0.87	0.86	0.86	0.86	0.86	0.69	0.85	0.85	0.86	0.84	*0.83*
9	0.77	0.83	0.82	0.84	0.83	0.83	0.82	0.83	0.67	0.82	0.82	0.83	0.81	*0.81*
10	0.75	0.79	0.79	0.80	0.80	0.80	0.80	0.80	0.66	0.78	0.78	0.80	0.77	*0.76*
11	0.72	0.75	0.75	0.76	0.76	0.76	0.76	0.76	0.66	0.75	0.75	0.76	0.74	*0.73*
12	0.68	0.72	0.72	0.72	0.72	0.72	0.72	0.72	0.66	0.71	0.71	0.72	0.70	*0.70*
13	0.64	0.68	0.68	0.68	0.68	0.68	0.68	0.68	0.65	0.68	0.68	0.68	0.67	*0.66*
14	0.60	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.63	0.64	0.64	0.64	0.64	*0.64*
15	0.57	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	*0.60*
16	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	*0.57*

Table D.17: ITD average $\mathcal{P}_n$ scores, for the various configurations of the Doddle algorithm, executed over Title and Description data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.92	0.92	0.90	0.94	0.90	0.94	0.92	0.88	0.92	0.90	0.92	0.92	*0.98*
2	0.93	0.93	0.88	0.91	0.91	0.95	0.94	0.92	0.86	0.93	0.93	0.93	*0.81*
3	0.90	0.91	0.85	0.91	0.88	0.93	0.91	0.87	0.85	0.90	0.90	0.89	*0.87*
4	0.89	0.89	0.85	0.92	0.88	0.92	0.88	0.85	0.86	0.88	0.87	0.88	*0.90*
5	0.87	0.86	0.83	0.91	0.86	0.90	0.89	0.84	0.83	0.87	0.86	0.87	*0.85*
6	0.87	0.84	0.81	0.89	0.83	0.88	0.87	0.80	0.79	0.84	0.83	0.86	*0.85*
7	0.83	0.82	0.78	0.86	0.79	0.85	0.84	0.77	0.77	0.83	0.80	0.81	*0.85*
8	0.81	0.79	0.75	0.84	0.77	0.82	0.80	0.75	0.75	0.80	0.76	0.78	*0.83*
9	0.77	0.76	0.73	0.82	0.74	0.80	0.78	0.73	0.73	0.77	0.74	0.76	*0.82*
10	0.75	0.74	0.70	0.79	0.73	0.78	0.75	0.72	0.71	0.75	0.72	0.74	*0.77*
11	0.73	0.73	0.69	0.76	0.71	0.76	0.73	0.70	0.70	0.73	0.69	0.72	*0.72*
12	0.70	0.69	0.67	0.73	0.69	0.72	0.70	0.68	0.68	0.70	0.67	0.70	*0.70*
13	0.67	0.67	0.65	0.69	0.66	0.68	0.68	0.65	0.65	0.67	0.65	0.67	*0.66*
14	0.64	0.64	0.63	0.65	0.64	0.65	0.64	0.63	0.63	0.64	0.63	0.64	*0.65*
15	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	*0.61*
16	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	*0.57*

Table D.18: ITD average $\mathcal{P}_n$ scores, for existing algorithms and Doddle, executed over Title and Description data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	1.00	1.00	1.00	1.00	1.00	0.96	1.00	1.00	0.48	0.82	0.82	1.00	0.92	*0.98*
2	0.98	1.00	0.99	1.00	1.00	0.95	1.00	1.00	0.52	0.84	0.85	0.96	0.93	*0.81*
3	0.97	0.99	0.98	0.99	0.99	0.96	0.99	0.99	0.57	0.85	0.86	0.97	0.90	*0.87*
4	0.95	0.98	0.96	0.97	0.96	0.95	0.96	0.96	0.57	0.85	0.86	0.94	0.89	*0.90*
5	0.92	0.95	0.94	0.95	0.94	0.92	0.94	0.94	0.57	0.86	0.86	0.93	0.87	*0.85*
6	0.89	0.93	0.92	0.92	0.92	0.91	0.92	0.92	0.59	0.83	0.84	0.91	0.87	*0.85*
7	0.84	0.89	0.88	0.90	0.89	0.89	0.89	0.89	0.59	0.82	0.83	0.88	0.83	*0.85*
8	0.83	0.86	0.84	0.87	0.87	0.87	0.85	0.87	0.60	0.79	0.80	0.86	0.81	*0.83*
9	0.79	0.83	0.81	0.83	0.84	0.84	0.84	0.84	0.59	0.78	0.78	0.83	0.77	*0.82*
10	0.76	0.80	0.79	0.80	0.81	0.80	0.80	0.80	0.60	0.76	0.76	0.79	0.75	*0.77*
11	0.73	0.77	0.76	0.77	0.77	0.77	0.76	0.77	0.60	0.74	0.74	0.76	0.73	*0.72*
12	0.69	0.73	0.73	0.73	0.73	0.74	0.73	0.73	0.60	0.71	0.71	0.73	0.70	*0.70*
13	0.65	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.61	0.69	0.69	0.68	0.67	*0.66*
14	0.61	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.61	0.65	0.65	0.65	0.64	*0.65*
15	0.58	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.60	0.61	0.61	0.61	0.61	*0.61*
16	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	*0.57*

Table D.19: ITD average precision within the top 5 collections, and the number of queries (out of 50) for which the first ranked collection is correct, for the various configurations of the Doddle algorithm.

	Precision @ 5		Correct @ Pos. 1			
Algorithms	Titles	Titles & Desc.	Titles		Titles & Desc.	
Doddle	0.73	0.68	27	*(54%)*	25	*(50%)*
Doddle_RC	0.76	0.72	29	*(58%)*	30	*(60%)*
Doddle_RP	0.76	0.65	28	*(56%)*	23	*(46%)*
Doddle_RF	0.67	0.64	12	*(24%)*	11	*(22%)*
Doddle_RC+RP	0.76	0.69	28	*(56%)*	26	*(52%)*
Doddle_RC+RF	0.72	0.69	30	*(60%)*	29	*(58%)*
Doddle_RP+RF	0.72	0.66	28	*(56%)*	23	*(46%)*
Doddle_×	0.73	0.67	27	*(54%)*	24	*(48%)*
Doddle_RC×RP	0.75	0.67	27	*(54%)*	26	*(52%)*
Doddle_RC×RF	0.75	0.69	28	*(56%)*	25	*(50%)*
Doddle_RP×RF	0.75	0.66	28	*(56%)*	24	*(48%)*
Doddle_W	0.75	0.69	27	*(54%)*	25	*(50%)*
SBR	*0.43*	*0.42*	7	*(14%)*	7	*(14%)*

Table D.20: ITD average precision within the top 5 collections, and the number of queries (out of 50) for which the first ranked collection is correct, for existing algorithms and Doddle.

	Precision @ 5		Correct @ Pos. 1			
Algorithms	Titles	Titles & Desc.	Titles		Titles & Desc.	
bGlOSS	0.63	0.67	23	*(46%)*	23	*(46%)*
CORI	0.71	0.70	22	*(44%)*	22	*(44%)*
Cosine Measure	0.66	0.69	14	*(28%)*	14	*(28%)*
Inner Product	0.69	0.68	10	*(20%)*	9	*(18%)*
Skew	0.68	0.66	10	*(20%)*	9	*(18%)*
Highest-available Similarity	0.67	0.58	9	*(18%)*	8	*(16%)*
CVV	0.67	0.65	11	*(22%)*	9	*(18%)*
DFPROP	0.68	0.66	10	*(20%)*	9	*(18%)*
Distribution of Informative Amt	0.31	0.18	2	*(4%)*	2	*(4%)*
SCS	0.51	0.43	0	*(0%)*	0	*(0%)*
AvICTF	0.51	0.43	0	*(0%)*	0	*(0%)*
NSCQ	0.61	0.57	8	*(16%)*	8	*(16%)*
Doddle	0.73	0.68	27	*(54%)*	25	*(50%)*
SBR	*0.43*	*0.42*	*7*	*(14%)*	*7*	*(14%)*

D.2 Refined Test Data (RTD)

Table D.21: RTD average Spearman rank correlations (50 queries), for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		SBR	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
Doddle	**0.86**	**0.65**	**0.38**	**0.30**
Doddle_RC	**0.87**	**0.72**	**0.37**	**0.34**
Doddle_RP	**0.86**	**0.60**	**0.36**	**0.23**
Doddle_RF	**0.80**	**0.64**	**0.53**	**0.44**
Doddle_RC+RP	**0.86**	**0.66**	**0.37**	**0.28**
Doddle_RC+RF	**0.86**	**0.70**	**0.39**	**0.37**
Doddle_RP+RF	**0.85**	**0.60**	**0.39**	**0.29**
Doddle_×	**0.86**	**0.65**	**0.36**	**0.28**
Doddle_RC×RP	**0.86**	**0.65**	**0.36**	**0.27**
Doddle_RC×RF	**0.86**	**0.71**	**0.37**	**0.35**
Doddle_RP×RF	**0.85**	**0.61**	**0.37**	**0.26**
Doddle_W	**0.86**	**0.66**	**0.37**	**0.29**
SBR	***0.42***	***0.46***	—	—

Table D.22: RTD average Spearman rank correlations (50 queries), for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		**SBR**	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
bGlOSS	**0.19**	**0.27**	0.15	**0.22**
CORI	**0.89**	**0.77**	**0.53**	**0.57**
Cosine Measure	**0.80**	**0.63**	**0.52**	**0.47**
Inner Product	**0.85**	**0.75**	**0.61**	**0.65**
Skew	**0.85**	**0.75**	**0.61**	**0.65**
Highest-available Similarity	**0.85**	**0.79**	**0.61**	**0.69**
CVV	**0.85**	**0.75**	**0.61**	**0.66**
DFPROP	**0.85**	**0.75**	**0.61**	**0.66**
Distribution of Informative Amt	**0.55**	**0.35**	**0.27**	0.10
SCS	**0.62**	**0.47**	**0.53**	**0.46**
AvICTF	**0.62**	**0.47**	**0.53**	**0.46**
NSCQ	**0.81**	**0.72**	**0.65**	**0.71**
Doddle	**0.86**	**0.65**	**0.38**	**0.30**
SBR	***0.42***	***0.46***	—	—

Table D.23: RTD Z-Test results (50 queries), showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Title data.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AvICTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
bGlOSS	✓		✓																						
CORI	✓																								
Cosine Measure	✓																								
Inner Product	✓																								
Skew	✓																								
Highest-available Similarity	✓																								
CVV	✓																								
DFPROP	✓																								
Distribution of Informative Amt	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SCS	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
AvICTF	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NSCQ	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RP	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RF	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC+RP	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC+RF	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RP+RF	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_×	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC×RP	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC×RF	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RP×RF	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_W	✓										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SBR																									

Table D.24: RTD Z-Test results (50 queries), showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Title and Description data.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AviCTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
bGlOSS	✓		✓																						
CORI	✓																								
Cosine Measure	✓					✓																			
Inner Product	✓																								
Skew	✓																								
Highest-available Similarity	✓		✓																						
CVV	✓																								
DFPROP	✓																								
Distribution of Informative Amt																									
SCS																									
AviCTF	✓																								
NSCQ	✓																								
Doddle	✓																								
Doddle_RC	✓																								
Doddle_RP	✓																								
Doddle_RF	✓																								
Doddle_RC×RP	✓																								
Doddle_RC×RF	✓																								
Doddle_RP×RF	✓																								
Doddle_W	✓																								
SBR	✓		✓	✓	✓	✓	✓	✓			✓	✓	✓	✓		✓	✓		✓	✓	✓	✓		✓	

Table D.25: RTD average Spearman rank correlations (200 queries), for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		SBR	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
Doddle	**0.92**	**0.76**	**0.40**	**0.35**
Doddle_RC	**0.93**	**0.80**	**0.39**	**0.38**
Doddle_RP	**0.92**	**0.73**	**0.39**	**0.29**
Doddle_RF	**0.87**	**0.72**	**0.51**	**0.46**
Doddle_RC+RP	**0.93**	**0.77**	**0.39**	**0.33**
Doddle_RC+RF	**0.92**	**0.78**	**0.41**	**0.41**
Doddle_RP+RF	**0.92**	**0.73**	**0.41**	**0.34**
Doddle_×	**0.92**	**0.76**	**0.39**	**0.33**
Doddle_RC×RP	**0.92**	**0.76**	**0.39**	**0.32**
Doddle_RC×RF	**0.92**	**0.79**	**0.40**	**0.39**
Doddle_RP×RF	**0.92**	**0.73**	**0.39**	**0.32**
Doddle_W	**0.92**	**0.76**	**0.40**	**0.34**
SBR	***0.42***	***0.45***	—	—

Table D.26: RTD average Spearman rank correlations (200 queries), for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		SBR	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
bGlOSS	**0.19**	**0.25**	0.14	**0.20**
CORI	**0.93**	**0.84**	**0.50**	**0.54**
Cosine Measure	**0.86**	**0.73**	**0.49**	**0.49**
Inner Product	**0.90**	**0.81**	**0.57**	**0.62**
Skew	**0.90**	**0.81**	**0.57**	**0.62**
Highest-available Similarity	**0.90**	**0.83**	**0.57**	**0.65**
CVV	**0.90**	**0.81**	**0.57**	**0.63**
DFPROP	**0.90**	**0.81**	**0.57**	**0.62**
Distribution of Informative Amt	**0.68**	**0.45**	**0.33**	**0.20**
SCS	**0.74**	**0.56**	**0.52**	**0.48**
AvICTF	**0.74**	**0.56**	**0.52**	**0.48**
NSCQ	**0.86**	**0.78**	**0.60**	**0.67**
Doddle	**0.92**	**0.76**	**0.40**	**0.35**
SBR	***0.42***	***0.45***	—	—

Table D.27: RTD Z-Test results (200 queries), showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Title data.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AvICTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
bGlOSS	✓		✓																						✓
CORI																									
Cosine Measure	✓												✓	✓	✓		✓	✓	✓	✓	✓	✓	✓		✓
Inner Product																									
Skew																									
Highest-available Similarity																									
CVV																									
DFPROP																									
Distribution of Informative Amt	✓		✓								✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SCS	✓		✓								✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
AvICTF	✓		✓									✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NSCQ	✓		✓								✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle												✓				✓									✓
Doddle_RC												✓				✓									✓
Doddle_RP												✓				✓									✓
Doddle_RF	✓											✓	✓	✓	✓		✓			✓	✓	✓	✓		✓
Doddle_RC+RP												✓				✓									✓
Doddle_RC+RF												✓				✓									✓
Doddle_RP+RF												✓				✓									✓
Doddle_×												✓				✓									✓
Doddle_RC×RP												✓				✓									✓
Doddle_RC×RF												✓				✓									✓
Doddle_RP×RF												✓				✓									✓
Doddle_W												✓				✓									✓
SBR	✓		✓									✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	

Table D.28: RTD Z-Test results (200 queries), showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Title and Description data.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AvICTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
bGlOSS										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CORI	✓								✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Cosine Measure	✓								✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Inner Product	✓								✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Skew	✓								✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Highest-available Similarity	✓								✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CVV	✓								✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DFPROP	✓								✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Distribution of Informative Amt																									
SCS	✓																								
AvICTF	✓	✓	✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NSCQ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC+RP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC+RF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Doddle_RP+RF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
Doddle_×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓
Doddle_RC×RP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓
Doddle_RC×RF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓
Doddle_RP×RF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
Doddle_W	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
SBR	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

Table D.29: RTD average Blest and Da Costa weighted rank correlations (50 queries), for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR.

	Blest		**Da Costa**	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
Doddle	**0.85**	**0.63**	**0.86**	**0.66**
Doddle_RC	**0.86**	**0.70**	**0.87**	**0.73**
Doddle_RP	**0.85**	**0.59**	**0.85**	**0.61**
Doddle_RF	**0.75**	**0.58**	**0.79**	**0.63**
Doddle_RC+RP	**0.85**	**0.65**	**0.86**	**0.67**
Doddle_RC+RF	**0.85**	**0.67**	**0.86**	**0.71**
Doddle_RP+RF	**0.84**	**0.58**	**0.85**	**0.61**
Doddle_×	**0.85**	**0.63**	**0.85**	**0.66**
Doddle_RC×RP	**0.85**	**0.64**	**0.86**	**0.66**
Doddle_RC×RF	**0.85**	**0.68**	**0.86**	**0.71**
Doddle_RP×RF	**0.84**	**0.58**	**0.85**	**0.61**
Doddle_W	**0.85**	**0.64**	**0.86**	**0.66**
SBR	*0.05*	***0.17***	***0.48***	***0.49***

Table D.30: RTD average Blest and Da Costa weighted rank correlations (50 queries), for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR.

	Blest		**Da Costa**	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
bGlOSS	0.13	0.09	**0.65**	**0.61**
CORI	**0.86**	**0.72**	**0.88**	**0.78**
Cosine Measure	**0.73**	**0.51**	**0.80**	**0.68**
Inner Product	**0.82**	**0.68**	**0.84**	**0.74**
Skew	**0.82**	**0.68**	**0.83**	**0.74**
Highest-available Similarity	**0.82**	**0.72**	**0.84**	**0.79**
CVV	**0.82**	**0.68**	**0.84**	**0.75**
DFPROP	**0.82**	**0.68**	**0.83**	**0.75**
Distribution of Informative Amt	**0.54**	**0.35**	**0.55**	**0.38**
SCS	**0.55**	**0.39**	**0.58**	**0.45**
AvICTF	**0.55**	**0.39**	**0.58**	**0.45**
NSCQ	**0.76**	**0.64**	**0.78**	**0.71**
Doddle	**0.85**	**0.63**	**0.86**	**0.66**
SBR	*0.05*	***0.17***	***0.48***	***0.49***

Table D.31: RTD average Blest and Da Costa weighted rank correlations (200 queries), for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR.

	Blest		**Da Costa**	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
Doddle	**0.92**	**0.75**	**0.92**	**0.76**
Doddle_RC	**0.92**	**0.79**	**0.92**	**0.80**
Doddle_RP	**0.92**	**0.72**	**0.92**	**0.73**
Doddle_RF	**0.84**	**0.68**	**0.85**	**0.70**
Doddle_RC+RP	**0.92**	**0.76**	**0.92**	**0.77**
Doddle_RC+RF	**0.91**	**0.77**	**0.92**	**0.78**
Doddle_RP+RF	**0.91**	**0.71**	**0.91**	**0.72**
Doddle_×	**0.92**	**0.75**	**0.92**	**0.76**
Doddle_RC×RP	**0.92**	**0.76**	**0.92**	**0.76**
Doddle_RC×RF	**0.92**	**0.77**	**0.92**	**0.79**
Doddle_RP×RF	**0.91**	**0.72**	**0.92**	**0.73**
Doddle_W	**0.92**	**0.75**	**0.92**	**0.76**
SBR	*0.02*	*0.14*	***0.49***	***0.49***

Table D.32: RTD average Blest and Da Costa weighted rank correlations (200 queries), for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR.

	Blest		**Da Costa**	
Algorithms	Titles	Titles & Desc.	Titles	Titles & Desc.
bGlOSS	**0.17**	0.10	**0.66**	**0.61**
CORI	**0.91**	**0.80**	**0.92**	**0.84**
Cosine Measure	**0.81**	**0.63**	**0.85**	**0.76**
Inner Product	**0.88**	**0.77**	**0.88**	**0.81**
Skew	**0.88**	**0.77**	**0.88**	**0.81**
Highest-available Similarity	**0.87**	**0.78**	**0.88**	**0.82**
CVV	**0.88**	**0.77**	**0.89**	**0.80**
DFPROP	**0.88**	**0.77**	**0.88**	**0.80**
Distribution of Informative Amt	**0.68**	**0.44**	**0.69**	**0.46**
SCS	**0.68**	**0.49**	**0.69**	**0.53**
AvICTF	**0.68**	**0.49**	**0.69**	**0.53**
NSCQ	**0.83**	**0.72**	**0.84**	**0.76**
Doddle	**0.92**	**0.75**	**0.92**	**0.76**
SBR	*0.02*	*0.14*	***0.49***	***0.49***

Table D.33: RTD average $\mathcal{R}_n$ scores (50 queries), for the various configurations of the Doddle algorithm, executed over Title data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.39	0.43	0.38	0.30	0.39	0.41	0.37	0.36	0.39	0.40	0.37	0.39	*0.01*
10	0.82	0.86	0.83	0.63	0.85	0.83	0.81	0.83	0.84	0.83	0.80	0.84	*0.27*
20	0.90	0.92	0.90	0.73	0.91	0.90	0.89	0.90	0.91	0.91	0.89	0.91	*0.42*
30	0.93	0.94	0.93	0.82	0.93	0.93	0.93	0.93	0.94	0.93	0.92	0.93	*0.53*
40	0.95	0.95	0.94	0.86	0.95	0.95	0.94	0.95	0.95	0.95	0.94	0.94	*0.66*
50	0.96	0.97	0.97	0.91	0.97	0.97	0.96	0.97	0.97	0.97	0.96	0.97	*0.75*
60	0.98	0.98	0.98	0.94	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	*0.83*
70	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	*0.89*
80	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.96*
90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.34: RTD average $\mathcal{R}_n$ scores (50 queries), for existing algorithms and Doddle, executed over Title data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.23	0.43	0.31	0.46	0.38	0.42	0.51	0.41	0.07	0.21	0.21	0.33	0.39	*0.01*
10	0.14	0.83	0.61	0.73	0.73	0.73	0.76	0.73	0.33	0.39	0.39	0.54	0.82	*0.27*
20	0.17	0.91	0.70	0.83	0.83	0.82	0.84	0.83	0.42	0.47	0.47	0.69	0.90	*0.42*
30	0.23	0.93	0.77	0.88	0.88	0.89	0.89	0.88	0.53	0.57	0.57	0.78	0.93	*0.53*
40	0.34	0.94	0.82	0.92	0.93	0.93	0.93	0.93	0.64	0.66	0.66	0.86	0.95	*0.66*
50	0.41	0.96	0.90	0.95	0.95	0.95	0.95	0.95	0.71	0.75	0.75	0.93	0.96	*0.75*
60	0.51	0.98	0.95	0.97	0.97	0.97	0.97	0.97	0.79	0.82	0.82	0.96	0.98	*0.83*
70	0.67	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.89	0.91	0.91	0.99	0.99	*0.89*
80	0.78	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97	0.99	0.99	1.00	1.00	*0.96*
90	0.89	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	*1.00*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.35: RTD average $\mathcal{R}_n$ scores (50 queries), for the various configurations of the Doddle algorithm, executed over Title and Description data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.36	0.40	0.39	0.17	0.39	0.33	0.30	0.34	0.37	0.31	0.26	0.39	*0.02*
10	0.69	0.75	0.66	0.41	0.72	0.69	0.63	0.71	0.72	0.72	0.64	0.70	*0.26*
20	0.76	0.81	0.73	0.58	0.78	0.78	0.69	0.76	0.76	0.78	0.70	0.76	*0.37*
30	0.81	0.86	0.77	0.67	0.82	0.84	0.76	0.80	0.81	0.85	0.77	0.81	*0.47*
40	0.85	0.90	0.81	0.75	0.86	0.88	0.81	0.85	0.85	0.88	0.81	0.86	*0.61*
50	0.89	0.93	0.86	0.80	0.90	0.92	0.85	0.89	0.89	0.92	0.86	0.89	*0.70*
60	0.94	0.96	0.90	0.88	0.94	0.96	0.90	0.93	0.93	0.95	0.90	0.93	*0.81*
70	0.97	0.98	0.94	0.94	0.97	0.97	0.94	0.96	0.96	0.97	0.94	0.97	*0.88*
80	0.99	0.99	0.98	0.98	0.99	0.99	0.98	0.98	0.98	0.99	0.97	0.99	*0.95*
90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.99*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.36: RTD average $\mathcal{R}_n$ scores (50 queries), for existing algorithms and Doddle, executed over Title and Description data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.31	0.45	0.25	0.39	0.31	0.35	0.54	0.36	0.07	0.13	0.13	0.26	0.36	*0.02*
10	0.23	0.72	0.48	0.61	0.61	0.59	0.64	0.62	0.19	0.28	0.28	0.40	0.69	*0.26*
20	0.24	0.82	0.56	0.74	0.74	0.75	0.74	0.74	0.35	0.39	0.39	0.59	0.76	*0.37*
30	0.29	0.85	0.63	0.81	0.81	0.84	0.82	0.81	0.44	0.49	0.49	0.70	0.81	*0.47*
40	0.39	0.88	0.70	0.85	0.85	0.91	0.85	0.85	0.54	0.56	0.56	0.80	0.85	*0.61*
50	0.47	0.92	0.77	0.89	0.89	0.94	0.89	0.89	0.62	0.64	0.64	0.87	0.89	*0.70*
60	0.57	0.95	0.84	0.93	0.94	0.97	0.93	0.93	0.74	0.73	0.73	0.92	0.94	*0.81*
70	0.69	0.98	0.92	0.97	0.97	0.98	0.97	0.97	0.81	0.81	0.81	0.96	0.97	*0.88*
80	0.80	0.99	0.97	0.99	0.99	1.00	0.99	0.99	0.90	0.90	0.91	0.99	0.99	*0.95*
90	0.90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	1.00	1.00	1.00	1.00	*0.99*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.37: RTD average $\mathcal{R}_n$ scores (200 queries), for the various configurations of the Doddle algorithm, executed over Title data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.47	0.49	0.49	0.31	0.49	0.46	0.46	0.46	0.48	0.44	0.43	0.48	*0.01*
10	0.89	0.90	0.89	0.66	0.90	0.87	0.87	0.89	0.90	0.87	0.87	0.89	*0.26*
20	0.94	0.95	0.95	0.78	0.95	0.93	0.93	0.94	0.95	0.94	0.94	0.95	*0.40*
30	0.96	0.97	0.96	0.86	0.97	0.96	0.96	0.96	0.96	0.96	0.96	0.96	*0.51*
40	0.97	0.98	0.97	0.91	0.98	0.97	0.97	0.98	0.98	0.97	0.97	0.97	*0.65*
50	0.98	0.98	0.98	0.94	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	*0.75*
60	0.99	0.99	0.99	0.97	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	*0.85*
70	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.93*
80	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.97*
90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.38: RTD average $\mathcal{R}_n$ scores (200 queries), for existing algorithms and Doddle, executed over Title data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.22	0.47	0.30	0.45	0.41	0.45	0.54	0.42	0.08	0.19	0.19	0.30	0.47	*0.01*
10	0.16	0.85	0.62	0.75	0.75	0.71	0.77	0.76	0.34	0.38	0.38	0.58	0.89	*0.26*
20	0.19	0.91	0.72	0.85	0.86	0.84	0.87	0.86	0.49	0.52	0.52	0.74	0.94	*0.40*
30	0.25	0.94	0.80	0.91	0.91	0.91	0.92	0.91	0.62	0.64	0.64	0.83	0.96	*0.51*
40	0.32	0.96	0.87	0.95	0.95	0.95	0.95	0.95	0.71	0.73	0.73	0.91	0.97	*0.65*
50	0.39	0.97	0.92	0.97	0.97	0.97	0.97	0.97	0.78	0.81	0.81	0.94	0.98	*0.75*
60	0.49	0.99	0.95	0.99	0.99	0.99	0.99	0.99	0.86	0.89	0.89	0.97	0.99	*0.85*
70	0.64	1.00	0.99	1.00	1.00	1.00	1.00	1.00	0.92	0.95	0.95	0.99	1.00	*0.93*
80	0.76	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	0.99	0.99	1.00	1.00	*0.97*
90	0.88	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	*1.00*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.39: RTD average $\mathcal{R}_n$ scores (200 queries), for the various configurations of the Doddle algorithm, executed over Title and Description data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.35	0.41	0.37	0.20	0.40	0.33	0.33	0.34	0.39	0.31	0.33	0.38	*0.01*
10	0.73	0.78	0.74	0.47	0.77	0.72	0.69	0.74	0.78	0.71	0.69	0.74	*0.25*
20	0.82	0.86	0.81	0.61	0.84	0.82	0.79	0.83	0.84	0.82	0.80	0.83	*0.37*
30	0.86	0.90	0.85	0.72	0.87	0.88	0.83	0.87	0.87	0.88	0.84	0.87	*0.48*
40	0.90	0.93	0.87	0.79	0.91	0.91	0.87	0.90	0.90	0.92	0.87	0.90	*0.62*
50	0.93	0.95	0.91	0.85	0.93	0.94	0.90	0.93	0.93	0.94	0.90	0.93	*0.71*
60	0.96	0.97	0.94	0.92	0.96	0.97	0.93	0.96	0.96	0.97	0.94	0.96	*0.81*
70	0.98	0.99	0.97	0.97	0.98	0.99	0.97	0.98	0.98	0.99	0.97	0.98	*0.90*
80	0.99	1.00	0.99	0.99	1.00	1.00	0.99	0.99	0.99	1.00	0.99	0.99	*0.96*
90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.40: RTD average $\mathcal{R}_n$ scores (200 queries), for existing algorithms and Doddle, executed over Title and Description data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.26	0.39	0.21	0.35	0.34	0.27	0.50	0.36	0.05	0.13	0.13	0.23	0.35	*0.01*
10	0.21	0.75	0.50	0.66	0.65	0.60	0.69	0.66	0.23	0.31	0.31	0.46	0.73	*0.25*
20	0.24	0.84	0.61	0.78	0.78	0.76	0.79	0.78	0.36	0.40	0.40	0.63	0.82	*0.37*
30	0.30	0.89	0.69	0.85	0.85	0.85	0.85	0.85	0.46	0.49	0.49	0.74	0.86	*0.48*
40	0.37	0.92	0.75	0.90	0.90	0.91	0.90	0.90	0.56	0.58	0.59	0.83	0.90	*0.62*
50	0.45	0.95	0.81	0.93	0.93	0.96	0.93	0.93	0.64	0.67	0.67	0.88	0.93	*0.71*
60	0.53	0.97	0.88	0.96	0.96	0.98	0.96	0.96	0.72	0.76	0.76	0.94	0.96	*0.81*
70	0.66	0.99	0.94	0.99	0.99	0.99	0.98	0.99	0.82	0.85	0.85	0.98	0.98	*0.90*
80	0.76	1.00	0.98	1.00	1.00	1.00	1.00	1.00	0.93	0.94	0.94	1.00	0.99	*0.96*
90	0.88	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	*1.00*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.41: RTD average $\hat{\mathcal{R}}_n$ scores (50 queries), for the various configurations of the Doddle algorithm, executed over Title data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.16	0.17	0.16	0.13	0.16	0.16	0.15	0.15	0.16	0.16	0.15	0.16	*0.01*
10	0.62	0.64	0.62	0.50	0.63	0.62	0.61	0.62	0.63	0.62	0.61	0.63	*0.20*
20	0.77	0.78	0.77	0.64	0.77	0.77	0.76	0.77	0.77	0.77	0.76	0.77	*0.36*
30	0.85	0.85	0.85	0.76	0.85	0.85	0.84	0.85	0.85	0.85	0.84	0.85	*0.48*
40	0.90	0.91	0.90	0.83	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90	*0.63*
50	0.94	0.95	0.94	0.89	0.95	0.94	0.94	0.95	0.95	0.95	0.94	0.95	*0.73*
60	0.98	0.98	0.98	0.94	0.98	0.98	0.97	0.98	0.98	0.98	0.97	0.98	*0.82*
70	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	*0.89*
80	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.96*
90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.42: RTD average $\hat{\mathcal{R}}_n$ scores (50 queries), for existing algorithms and Doddle, executed over Title data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.07	0.17	0.13	0.18	0.15	0.15	0.19	0.16	0.05	0.09	0.09	0.12	0.16	*0.01*
10	0.11	0.63	0.49	0.58	0.57	0.57	0.59	0.57	0.29	0.34	0.34	0.43	0.62	*0.20*
20	0.14	0.77	0.61	0.72	0.72	0.72	0.72	0.72	0.39	0.44	0.44	0.61	0.77	*0.36*
30	0.20	0.85	0.71	0.81	0.81	0.81	0.81	0.81	0.50	0.54	0.54	0.72	0.85	*0.48*
40	0.32	0.89	0.79	0.88	0.88	0.88	0.88	0.88	0.61	0.64	0.64	0.82	0.90	*0.63*
50	0.40	0.94	0.88	0.93	0.93	0.93	0.93	0.93	0.70	0.74	0.74	0.91	0.94	*0.73*
60	0.50	0.97	0.94	0.97	0.97	0.97	0.97	0.97	0.79	0.82	0.82	0.95	0.98	*0.82*
70	0.67	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.89	0.91	0.91	0.99	0.99	*0.89*
80	0.78	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97	0.99	0.99	1.00	1.00	*0.96*
90	0.89	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	*1.00*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.43: RTD average $\hat{\mathcal{R}}_n$ scores (50 queries), for the various configurations of the Doddle algorithm, executed over Title and Description data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.13	0.14	0.15	0.05	0.14	0.12	0.11	0.13	0.14	0.11	0.10	0.14	*0.01*
10	0.53	0.56	0.51	0.32	0.55	0.52	0.49	0.54	0.55	0.54	0.49	0.54	*0.18*
20	0.67	0.72	0.65	0.52	0.69	0.69	0.62	0.68	0.68	0.69	0.62	0.68	*0.32*
30	0.76	0.81	0.73	0.64	0.78	0.79	0.72	0.76	0.77	0.80	0.73	0.77	*0.43*
40	0.84	0.88	0.79	0.73	0.84	0.86	0.79	0.83	0.83	0.86	0.80	0.84	*0.59*
50	0.89	0.92	0.86	0.79	0.89	0.91	0.85	0.88	0.89	0.91	0.85	0.89	*0.69*
60	0.94	0.96	0.90	0.88	0.94	0.95	0.90	0.93	0.93	0.95	0.90	0.93	*0.81*
70	0.97	0.98	0.94	0.94	0.97	0.97	0.94	0.96	0.96	0.97	0.94	0.97	*0.88*
80	0.99	0.99	0.98	0.98	0.99	0.99	0.98	0.98	0.98	0.99	0.97	0.99	*0.95*
90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.99*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.44: RTD average $\hat{\mathcal{R}}_n$ scores (50 queries), for existing algorithms and Doddle, executed over Title and Description data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.08	0.15	0.09	0.14	0.11	0.14	0.18	0.13	0.04	0.04	0.04	0.08	0.13	*0.01*
10	0.17	0.54	0.38	0.46	0.46	0.45	0.48	0.46	0.15	0.23	0.23	0.32	0.53	*0.18*
20	0.21	0.72	0.51	0.66	0.66	0.66	0.66	0.66	0.32	0.35	0.35	0.53	0.67	*0.32*
30	0.27	0.80	0.60	0.77	0.77	0.80	0.77	0.77	0.43	0.47	0.47	0.67	0.76	*0.43*
40	0.38	0.86	0.69	0.83	0.83	0.89	0.83	0.83	0.53	0.55	0.55	0.79	0.84	*0.59*
50	0.47	0.91	0.76	0.88	0.88	0.93	0.89	0.88	0.62	0.64	0.64	0.86	0.89	*0.69*
60	0.57	0.95	0.84	0.93	0.93	0.97	0.93	0.93	0.73	0.73	0.73	0.92	0.94	*0.81*
70	0.68	0.98	0.92	0.97	0.97	0.98	0.97	0.97	0.81	0.81	0.81	0.96	0.97	*0.88*
80	0.80	0.99	0.97	0.99	0.99	1.00	0.99	0.99	0.90	0.90	0.91	0.99	0.99	*0.95*
90	0.90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	1.00	1.00	1.00	1.00	*0.99*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.45: RTD average $\hat{\mathcal{R}}_n$ scores (200 queries), for the various configurations of the Doddle algorithm, executed over Title data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.21	0.21	0.21	0.14	0.21	0.20	0.21	0.20	0.21	0.20	0.20	0.21	*0.00*
10	0.68	0.69	0.69	0.54	0.69	0.67	0.67	0.68	0.69	0.67	0.67	0.69	*0.19*
20	0.83	0.83	0.83	0.71	0.83	0.82	0.82	0.83	0.83	0.82	0.82	0.83	*0.35*
30	0.90	0.90	0.90	0.81	0.90	0.90	0.89	0.90	0.90	0.90	0.90	0.90	*0.48*
40	0.94	0.94	0.94	0.88	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	*0.63*
50	0.97	0.97	0.97	0.93	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	*0.74*
60	0.99	0.99	0.99	0.97	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.99	*0.84*
70	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.93*
80	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.97*
90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.46: RTD average $\hat{\mathcal{R}}_n$ scores (200 queries), for existing algorithms and Doddle, executed over Title data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.08	0.20	0.13	0.19	0.18	0.18	0.21	0.18	0.05	0.09	0.09	0.14	0.21	*0.00*
10	0.13	0.66	0.51	0.60	0.60	0.57	0.62	0.60	0.30	0.34	0.34	0.48	0.68	*0.19*
20	0.16	0.81	0.66	0.76	0.77	0.75	0.77	0.77	0.46	0.49	0.49	0.67	0.83	*0.35*
30	0.23	0.88	0.76	0.86	0.86	0.85	0.86	0.86	0.59	0.62	0.62	0.79	0.90	*0.48*
40	0.30	0.93	0.84	0.92	0.92	0.92	0.92	0.92	0.69	0.72	0.72	0.88	0.94	*0.63*
50	0.38	0.96	0.91	0.96	0.96	0.96	0.96	0.96	0.77	0.81	0.81	0.93	0.97	*0.74*
60	0.49	0.98	0.95	0.98	0.98	0.98	0.98	0.98	0.86	0.89	0.89	0.97	0.99	*0.84*
70	0.64	1.00	0.99	1.00	1.00	1.00	1.00	1.00	0.92	0.95	0.95	0.99	1.00	*0.93*
80	0.76	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	0.99	0.99	1.00	1.00	*0.97*
90	0.88	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	*1.00*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.47: RTD average $\hat{\mathcal{R}}_n$ scores (200 queries), for the various configurations of the Doddle algorithm, executed over Title and Description data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.14	0.15	0.15	0.08	0.16	0.12	0.13	0.13	0.15	0.12	0.13	0.15	*0.00*
10	0.55	0.58	0.56	0.37	0.58	0.54	0.53	0.56	0.58	0.54	0.53	0.56	*0.18*
20	0.72	0.75	0.71	0.54	0.73	0.72	0.69	0.72	0.73	0.72	0.70	0.72	*0.31*
30	0.81	0.84	0.79	0.68	0.82	0.82	0.78	0.81	0.82	0.82	0.79	0.82	*0.44*
40	0.87	0.90	0.85	0.77	0.88	0.89	0.85	0.88	0.88	0.89	0.85	0.88	*0.61*
50	0.92	0.95	0.90	0.85	0.92	0.93	0.90	0.92	0.92	0.94	0.90	0.92	*0.70*
60	0.96	0.97	0.94	0.92	0.96	0.97	0.93	0.95	0.96	0.97	0.94	0.96	*0.81*
70	0.98	0.99	0.97	0.97	0.98	0.99	0.97	0.98	0.98	0.99	0.97	0.98	*0.90*
80	0.99	1.00	0.99	0.99	1.00	1.00	0.99	0.99	0.99	1.00	0.99	0.99	*0.96*
90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.48: RTD average $\hat{\mathcal{R}}_n$ scores (200 queries), for existing algorithms and Doddle, executed over Title and Description data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.09	0.13	0.09	0.14	0.14	0.11	0.19	0.14	0.03	0.06	0.06	0.10	0.14	*0.00*
10	0.16	0.56	0.39	0.50	0.50	0.46	0.52	0.50	0.20	0.26	0.26	0.36	0.55	*0.18*
20	0.21	0.73	0.55	0.68	0.68	0.66	0.69	0.68	0.33	0.37	0.37	0.55	0.72	*0.31*
30	0.28	0.83	0.65	0.80	0.80	0.80	0.80	0.80	0.44	0.47	0.47	0.70	0.81	*0.44*
40	0.36	0.90	0.73	0.88	0.88	0.88	0.87	0.88	0.55	0.57	0.58	0.81	0.87	*0.61*
50	0.45	0.94	0.81	0.92	0.92	0.95	0.92	0.92	0.64	0.67	0.67	0.88	0.92	*0.70*
60	0.53	0.96	0.88	0.96	0.96	0.98	0.96	0.96	0.72	0.76	0.76	0.93	0.96	*0.81*
70	0.66	0.99	0.94	0.99	0.99	0.99	0.98	0.99	0.82	0.85	0.85	0.98	0.98	*0.90*
80	0.76	1.00	0.98	1.00	1.00	1.00	1.00	1.00	0.93	0.94	0.94	1.00	0.99	*0.96*
90	0.88	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	*1.00*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.49: RTD average $\mathcal{P}_n$ scores (50 queries), for the various configurations of the Doddle algorithm, executed over Title data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.96	1.00	0.96	0.90	0.96	0.98	0.96	0.94	0.94	0.98	0.96	0.96	*0.62*
10	0.82	0.84	0.83	0.82	0.83	0.82	0.82	0.83	0.82	0.83	0.82	0.83	*0.73*
20	0.74	0.74	0.73	0.74	0.74	0.74	0.73	0.73	0.73	0.74	0.73	0.74	*0.66*
30	0.69	0.69	0.68	0.69	0.69	0.69	0.69	0.68	0.68	0.68	0.68	0.68	*0.64*
40	0.63	0.63	0.63	0.63	0.63	0.64	0.64	0.63	0.63	0.63	0.63	0.63	*0.61*
50	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	*0.56*
60	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56	*0.53*
70	0.53	0.52	0.52	0.52	0.52	0.53	0.52	0.52	0.52	0.52	0.52	0.52	*0.51*
80	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	*0.46*
90	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	*0.42*
100	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	*0.38*

Table D.50: RTD average $\mathcal{P}_n$ scores (50 queries), for existing algorithms and Doddle, executed over Title data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.70	1.00	0.98	1.00	1.00	1.00	1.00	1.00	0.72	0.98	0.98	1.00	0.96	*0.62*
10	0.20	0.90	0.82	0.90	0.90	0.90	0.90	0.90	0.67	0.79	0.79	0.89	0.82	*0.73*
20	0.19	0.78	0.72	0.79	0.79	0.80	0.79	0.79	0.60	0.68	0.68	0.78	0.74	*0.66*
30	0.20	0.73	0.67	0.73	0.72	0.73	0.72	0.72	0.58	0.64	0.64	0.71	0.69	*0.64*
40	0.23	0.67	0.62	0.67	0.66	0.67	0.66	0.66	0.55	0.60	0.60	0.66	0.63	*0.61*
50	0.25	0.62	0.58	0.62	0.62	0.62	0.62	0.62	0.53	0.57	0.57	0.61	0.60	*0.56*
60	0.28	0.57	0.55	0.58	0.58	0.58	0.57	0.58	0.52	0.55	0.55	0.57	0.56	*0.53*
70	0.31	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.50	0.52	0.52	0.53	0.53	*0.51*
80	0.34	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.47	0.48	0.48	0.48	0.48	*0.46*
90	0.36	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	*0.42*
100	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	*0.38*

Table D.51: RTD average $\mathcal{P}_n$ scores (50 queries), for the various configurations of the Doddle algorithm, executed over Title and Description data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.94	0.94	0.84	0.80	0.86	0.92	0.88	0.90	0.86	0.92	0.84	0.92	*0.70*
10	0.76	0.82	0.72	0.76	0.76	0.80	0.73	0.76	0.76	0.80	0.74	0.75	*0.79*
20	0.71	0.75	0.68	0.72	0.71	0.74	0.68	0.71	0.70	0.74	0.68	0.71	*0.73*
30	0.66	0.70	0.63	0.68	0.66	0.70	0.63	0.66	0.66	0.70	0.64	0.66	*0.70*
40	0.62	0.66	0.60	0.64	0.63	0.66	0.60	0.63	0.63	0.66	0.60	0.62	*0.66*
50	0.59	0.62	0.57	0.60	0.59	0.61	0.57	0.59	0.58	0.62	0.57	0.59	*0.61*
60	0.55	0.57	0.53	0.56	0.55	0.57	0.53	0.54	0.54	0.57	0.53	0.55	*0.56*
70	0.51	0.52	0.49	0.52	0.51	0.52	0.50	0.50	0.50	0.52	0.49	0.51	*0.52*
80	0.47	0.48	0.46	0.48	0.47	0.48	0.47	0.47	0.47	0.48	0.47	0.47	*0.48*
90	0.44	0.43	0.43	0.44	0.43	0.44	0.43	0.43	0.43	0.44	0.43	0.43	*0.43*
100	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	*0.39*

Table D.52: RTD average $\mathcal{P}_n$ scores (50 queries), for existing algorithms and Doddle, executed over Title and Description data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.82	1.00	0.94	1.00	1.00	0.96	1.00	1.00	0.56	0.90	0.90	0.96	0.94	*0.70*
10	0.39	0.90	0.77	0.91	0.91	0.92	0.91	0.91	0.57	0.73	0.73	0.86	0.76	*0.79*
20	0.30	0.84	0.69	0.84	0.84	0.88	0.84	0.84	0.55	0.65	0.65	0.83	0.71	*0.73*
30	0.26	0.78	0.62	0.79	0.79	0.81	0.79	0.79	0.52	0.61	0.61	0.78	0.66	*0.70*
40	0.27	0.72	0.57	0.72	0.72	0.75	0.72	0.72	0.49	0.58	0.58	0.73	0.62	*0.66*
50	0.27	0.66	0.54	0.66	0.66	0.69	0.66	0.66	0.48	0.55	0.55	0.66	0.59	*0.61*
60	0.28	0.60	0.51	0.59	0.59	0.61	0.59	0.59	0.46	0.52	0.52	0.60	0.55	*0.56*
70	0.32	0.54	0.49	0.54	0.54	0.55	0.54	0.54	0.45	0.49	0.49	0.54	0.51	*0.52*
80	0.34	0.49	0.47	0.49	0.49	0.49	0.49	0.49	0.44	0.47	0.47	0.49	0.47	*0.48*
90	0.37	0.44	0.43	0.44	0.44	0.44	0.44	0.44	0.43	0.44	0.44	0.44	0.44	*0.43*
100	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	*0.39*

Table D.53: RTD average $\mathcal{P}_n$ scores (200 queries), for the various configurations of the Doddle algorithm, executed over Title data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.98	1.00	0.99	0.92	0.99	0.98	0.98	0.98	0.99	0.97	0.97	0.99	*0.61*
10	0.82	0.83	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	*0.70*
20	0.75	0.75	0.75	0.76	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	*0.64*
30	0.69	0.69	0.69	0.70	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	*0.60*
40	0.64	0.64	0.63	0.64	0.63	0.64	0.64	0.63	0.63	0.64	0.63	0.63	*0.57*
50	0.58	0.58	0.58	0.58	0.58	0.59	0.59	0.58	0.58	0.58	0.58	0.58	*0.53*
60	0.54	0.54	0.53	0.53	0.54	0.54	0.54	0.53	0.53	0.54	0.53	0.54	*0.50*
70	0.49	0.49	0.48	0.48	0.49	0.49	0.49	0.49	0.49	0.49	0.48	0.49	*0.47*
80	0.44	0.43	0.43	0.43	0.43	0.44	0.44	0.43	0.43	0.43	0.43	0.44	*0.43*
90	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	*0.39*
100	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	*0.35*

Table D.54: RTD average $\mathcal{P}_n$ scores (200 queries), for existing algorithms and Doddle, executed over Title data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.67	1.00	0.98	1.00	1.00	1.00	1.00	1.00	0.81	0.98	0.98	0.99	0.98	*0.61*
10	0.19	0.85	0.81	0.85	0.85	0.85	0.85	0.85	0.71	0.80	0.80	0.84	0.82	*0.70*
20	0.17	0.77	0.73	0.78	0.78	0.78	0.77	0.78	0.67	0.72	0.72	0.77	0.75	*0.64*
30	0.19	0.71	0.67	0.71	0.71	0.71	0.71	0.71	0.62	0.67	0.67	0.71	0.69	*0.60*
40	0.20	0.65	0.62	0.65	0.65	0.65	0.65	0.65	0.58	0.61	0.61	0.65	0.64	*0.57*
50	0.22	0.60	0.57	0.60	0.60	0.60	0.60	0.60	0.54	0.57	0.57	0.59	0.58	*0.53*
60	0.24	0.54	0.53	0.54	0.54	0.55	0.54	0.54	0.51	0.52	0.52	0.54	0.54	*0.50*
70	0.27	0.49	0.48	0.49	0.49	0.49	0.49	0.49	0.47	0.48	0.48	0.49	0.49	*0.47*
80	0.30	0.44	0.43	0.44	0.44	0.44	0.44	0.44	0.43	0.43	0.43	0.44	0.44	*0.43*
90	0.33	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	*0.39*
100	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	*0.35*

Table D.55: RTD average $\mathcal{P}_n$ scores (200 queries), for the various configurations of the Doddle algorithm, executed over Title and Description data.

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.90	0.96	0.89	0.81	0.92	0.91	0.86	0.90	0.91	0.91	0.89	0.92	*0.70*
10	0.79	0.82	0.77	0.79	0.80	0.81	0.77	0.79	0.79	0.81	0.77	0.78	*0.78*
20	0.74	0.76	0.71	0.74	0.73	0.76	0.72	0.73	0.73	0.76	0.72	0.73	*0.72*
30	0.69	0.72	0.67	0.70	0.69	0.72	0.68	0.69	0.69	0.72	0.67	0.69	*0.69*
40	0.65	0.68	0.63	0.66	0.65	0.68	0.64	0.65	0.65	0.68	0.64	0.65	*0.65*
50	0.61	0.63	0.59	0.62	0.61	0.63	0.60	0.61	0.61	0.63	0.60	0.61	*0.60*
60	0.57	0.58	0.55	0.58	0.57	0.59	0.56	0.57	0.57	0.58	0.56	0.57	*0.56*
70	0.53	0.53	0.52	0.53	0.53	0.54	0.52	0.52	0.52	0.54	0.52	0.53	*0.53*
80	0.48	0.48	0.48	0.48	0.48	0.49	0.48	0.48	0.48	0.48	0.48	0.48	*0.48*
90	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.44	*0.44*
100	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	*0.39*

Table D.56: RTD average $\mathcal{P}_n$ scores (200 queries), for existing algorithms and Doddle, executed over Title and Description data.

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.79	1.00	0.94	1.00	0.99	1.00	0.99	0.99	0.70	0.94	0.94	0.97	0.90	*0.70*
10	0.34	0.89	0.78	0.90	0.89	0.90	0.90	0.89	0.62	0.75	0.75	0.87	0.79	*0.78*
20	0.27	0.82	0.70	0.83	0.83	0.85	0.83	0.83	0.59	0.68	0.68	0.82	0.74	*0.72*
30	0.26	0.77	0.65	0.78	0.78	0.80	0.78	0.78	0.56	0.65	0.65	0.77	0.69	*0.69*
40	0.26	0.72	0.61	0.72	0.72	0.74	0.72	0.72	0.54	0.61	0.61	0.72	0.65	*0.65*
50	0.27	0.66	0.58	0.66	0.66	0.69	0.66	0.66	0.52	0.58	0.58	0.67	0.61	*0.60*
60	0.28	0.60	0.55	0.61	0.61	0.62	0.60	0.61	0.50	0.55	0.55	0.61	0.57	*0.56*
70	0.32	0.55	0.51	0.55	0.55	0.55	0.55	0.55	0.48	0.52	0.52	0.55	0.53	*0.53*
80	0.34	0.49	0.48	0.49	0.49	0.49	0.49	0.49	0.46	0.48	0.48	0.49	0.48	*0.48*
90	0.37	0.44	0.44	0.44	0.44	0.44	0.44	0.44	0.43	0.44	0.44	0.44	0.44	*0.44*
100	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	*0.39*

Table D.57: RTD average precision within the top 5 collections, and the number of queries (out of 50) for which the first ranked collection is correct, for the various configurations of the Doddle algorithm.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Titles	Titles & Desc.	Titles		Titles & Desc.	
Doddle	0.53	0.38	13	*(26%)*	11	*(22%)*
Doddle_RC	0.56	0.43	14	*(28%)*	11	*(22%)*
Doddle_RP	0.57	0.46	13	*(26%)*	13	*(26%)*
Doddle_RF	0.26	0.21	7	*(14%)*	2	*(4%)*
Doddle_RC+RP	0.57	0.44	13	*(26%)*	13	*(26%)*
Doddle_RC+RF	0.51	0.34	14	*(28%)*	8	*(16%)*
Doddle_RP+RF	0.50	0.36	12	*(24%)*	9	*(18%)*
Doddle_×	0.55	0.38	12	*(24%)*	9	*(18%)*
Doddle_RC×RP	0.56	0.44	13	*(26%)*	11	*(22%)*
Doddle_RC×RF	0.53	0.34	13	*(26%)*	8	*(16%)*
Doddle_RP×RF	0.52	0.38	12	*(24%)*	7	*(14%)*
Doddle_W	0.56	0.41	13	*(26%)*	13	*(26%)*
SBR	*0.07*	*0.09*	*0*	*(0%)*	*1*	*(2%)*

Table D.58: RTD average precision within the top 5 collections, and the number of queries (out of 50) for which the first ranked collection is correct, for existing algorithms and Doddle.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Titles	Titles & Desc.	Titles		Titles & Desc.	
bGlOSS	0.12	0.17	5	*(10%)*	4	*(8%)*
CORI	0.52	0.38	13	*(26%)*	10	*(20%)*
Cosine Measure	0.26	0.18	7	*(14%)*	5	*(10%)*
Inner Product	0.37	0.29	16	*(32%)*	10	*(20%)*
Skew	0.36	0.29	11	*(22%)*	7	*(14%)*
Highest-available Similarity	0.39	0.30	12	*(24%)*	10	*(20%)*
CVV	0.39	0.30	16	*(32%)*	18	*(36%)*
DFPROP	0.36	0.29	13	*(26%)*	10	*(20%)*
Distribution of Informative Amt	0.18	0.09	2	*(4%)*	2	*(4%)*
SCS	0.18	0.14	5	*(10%)*	2	*(4%)*
AvICTF	0.18	0.15	5	*(10%)*	2	*(4%)*
NSCQ	0.28	0.18	10	*(20%)*	6	*(12%)*
Doddle	0.53	0.38	13	*(26%)*	11	*(22%)*
SBR	*0.07*	*0.09*	0	*(0%)*	*1*	*(2%)*

Table D.59: RTD average precision within the top 5 collections, and the number of queries (out of 200) for which the first ranked collection is correct, for the various configurations of the Doddle algorithm.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Titles	Titles & Desc.	Titles		Titles & Desc.	
Doddle	0.61	0.43	59	*(29.5%)*	38	*(19.0%)*
Doddle_RC	0.64	0.48	61	*(30.5%)*	47	*(23.5%)*
Doddle_RP	0.66	0.49	68	*(34.0%)*	44	*(22.0%)*
Doddle_RF	0.32	0.25	32	*(16.0%)*	17	*(8.5%)*
Doddle_RC+RP	0.65	0.48	64	*(32.0%)*	47	*(23.5%)*
Doddle_RC+RF	0.59	0.41	55	*(27.5%)*	34	*(17.0%)*
Doddle_RP+RF	0.60	0.41	62	*(31.0%)*	35	*(17.5%)*
Doddle_×	0.62	0.44	58	*(29.0%)*	36	*(18.0%)*
Doddle_RC×RP	0.66	0.49	63	*(31.5%)*	44	*(22.0%)*
Doddle_RC×RF	0.60	0.39	52	*(26.0%)*	30	*(15.0%)*
Doddle_RP×RF	0.60	0.41	57	*(28.5%)*	36	*(18.0%)*
Doddle_W	0.63	0.45	62	*(31.0%)*	41	*(20.5%)*
SBR	*0.07*	*0.07*	0	*(0.0%)*	*1*	*(0.5%)*

Table D.60: RTD average precision within the top 5 collections, and the number of queries (out of 200) for which the first ranked collection is correct, for existing algorithms and Doddle.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Titles	Titles & Desc.	Titles		Titles & Desc.	
bGlOSS	0.11	0.15	17	*(8.5%)*	19	*(9.5%)*
CORI	0.58	0.42	53	*(26.5%)*	39	*(19.5%)*
Cosine Measure	0.29	0.23	29	*(14.5%)*	19	*(9.5%)*
Inner Product	0.45	0.34	52	*(26.0%)*	34	*(17.0%)*
Skew	0.43	0.33	44	*(22.0%)*	35	*(17.5%)*
Highest-available Similarity	0.44	0.32	48	*(24.0%)*	27	*(13.5%)*
CVV	0.46	0.38	66	*(33.0%)*	65	*(32.5%)*
DFPROP	0.44	0.34	45	*(22.5%)*	40	*(20.0%)*
Distribution of Informative Amt	0.18	0.11	9	*4.5%)*	5	*(2.5%)*
SCS	0.24	0.19	16	*8.0%)*	9	*(4.5%)*
AvICTF	0.24	0.19	16	*8.0%)*	9	*(4.5%)*
NSCQ	0.35	0.24	28	*(14.0%)*	19	*(9.5%)*
Doddle	0.61	0.43	59	*(29.5%)*	38	*(19.0%)*
SBR	*0.07*	*0.07*	0	*(0.0%)*	*1*	*(0.5%)*

D.3 SYM-236

Table D.61: SYM-236 average Spearman rank correlations, for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		**SBR**	
Algorithms	Q_s	Q_l	Q_s	Q_l
Doddle	**0.39**	**0.46**	**0.33**	**0.41**
Doddle_RC	**0.53**	**0.57**	**0.56**	**0.59**
Doddle_RP	**0.32**	**0.40**	**0.20**	**0.24**
Doddle_RF	**0.24**	**0.32**	**0.29**	**0.36**
Doddle_RC+RP	**0.44**	**0.50**	**0.36**	**0.42**
Doddle_RC+RF	**0.43**	**0.49**	**0.44**	**0.50**
Doddle_RP+RF	**0.28**	**0.36**	**0.20**	**0.30**
Doddle_×	**0.36**	**0.33**	**0.28**	**0.18**
Doddle_RC×RP	**0.41**	**0.42**	**0.34**	**0.28**
Doddle_RC×RF	**0.41**	**0.39**	**0.39**	**0.32**
Doddle_RP×RF	**0.26**	**0.29**	**0.14**	**0.13**
Doddle_W	**0.41**	**0.48**	**0.34**	**0.41**
SBR	***0.53***	***0.53***	—	—

Table D.62: SYM-236 average Spearman rank correlations, for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		**SBR**	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.58**	**0.35**	**0.67**	**0.33**
CORI	**0.65**	**0.65**	**0.85**	**0.85**
Cosine Measure	**0.53**	**0.64**	**0.75**	**0.82**
Inner Product	**0.63**	**0.65**	**0.89**	**0.89**
Skew	**0.61**	**0.62**	**0.89**	**0.89**
Highest-available Similarity	**0.64**	**0.66**	**0.91**	**0.92**
CVV	**0.59**	**0.59**	**0.90**	**0.92**
DFPROP	**0.61**	**0.62**	**0.89**	**0.91**
Distribution of Informative Amt	−0.24	−0.32	−0.18	−0.25
SCS	**0.18**	**0.32**	**0.61**	**0.75**
AvICTF	**0.19**	**0.39**	**0.62**	**0.78**
NSCQ	**0.50**	**0.58**	**0.91**	**0.93**
Doddle	**0.39**	**0.46**	**0.33**	**0.41**
SBR	***0.53***	***0.53***	—	—

Table D.63: SYM-236 Z-Test results, showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Short queries.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AvICTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
bGlOSS																									
CORI			✓																						
Cosine Measure																									
Inner Product																									
Skew																									
Highest-available Similarity																									
CVV																									
DFPROP																									
Distribution of Informative Amt																									
SCS																									
AvICTF																									
NSCQ																									
Doddle																									
Doddle_RC																									
Doddle_RP																									
Doddle_RF																									
Doddle_RC+RP																									
Doddle_RC+RF																									
Doddle_RP+RF																									
Doddle_×																									
Doddle_RC×RP																									
Doddle_RC×RF																									
Doddle_RP×RF																									
Doddle_W																									
SBR																									

Table D.64: SYM-236 Z-Test results, showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Long queries.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AvICTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
bGlOSS		✓	✓	✓	✓	✓	✓	✓	✓			✓		✓			✓								
CORI			✓	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Cosine Measure				✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Inner Product					✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Skew						✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Highest-available Similarity							✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CVV								✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DFPROP									✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Distribution of Informative Amt	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SCS									✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
AvICTF									✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NSCQ									✓				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle									✓			✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC									✓			✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RP									✓			✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RF									✓			✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC+RP									✓			✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC+RF									✓			✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Doddle_RP+RF									✓			✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
Doddle_×									✓			✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓
Doddle_RC×RP									✓			✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓
Doddle_RC×RF									✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓
Doddle_RP×RF									✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
Doddle_W									✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
SBR									✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

Table D.65: SYM-236 average Blest and Da Costa weighted rank correlations, for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR.

	Blest		Da Costa	
Algorithms	Q_s	Q_l	Q_s	Q_l
Doddle	**0.37**	**0.43**	**0.44**	**0.52**
Doddle_RC	**0.52**	**0.55**	**0.58**	**0.63**
Doddle_RP	**0.31**	**0.37**	**0.39**	**0.46**
Doddle_RF	**0.21**	**0.27**	**0.29**	**0.37**
Doddle_RC+RP	**0.42**	**0.48**	**0.49**	**0.56**
Doddle_RC+RF	**0.41**	**0.45**	**0.48**	**0.54**
Doddle_RP+RF	**0.25**	**0.32**	**0.34**	**0.42**
Doddle_×	**0.33**	**0.30**	**0.42**	**0.40**
Doddle_RC×RP	**0.39**	**0.40**	**0.47**	**0.48**
Doddle_RC×RF	**0.38**	**0.36**	**0.46**	**0.45**
Doddle_RP×RF	**0.22**	**0.25**	**0.32**	**0.35**
Doddle_W	**0.39**	**0.45**	**0.47**	**0.54**
SBR	***0.53***	***0.53***	***0.58***	***0.58***

Table D.66: SYM-236 average Blest and Da Costa weighted rank correlations, for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR.

	Blest		Da Costa	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.63**	**0.53**	**0.66**	**0.63**
CORI	**0.65**	**0.63**	**0.70**	**0.70**
Cosine Measure	**0.53**	**0.62**	**0.58**	**0.70**
Inner Product	**0.63**	**0.63**	**0.68**	**0.70**
Skew	**0.61**	**0.60**	**0.66**	**0.67**
Highest-available Similarity	**0.65**	**0.65**	**0.69**	**0.71**
CVV	**0.58**	**0.56**	**0.63**	**0.63**
DFPROP	**0.61**	**0.60**	**0.66**	**0.67**
Distribution of Informative Amt	−0.23	−0.34	−0.11	−0.19
SCS	**0.18**	**0.30**	**0.22**	**0.35**
AvICTF	**0.19**	**0.38**	**0.23**	**0.43**
NSCQ	**0.51**	**0.56**	**0.54**	**0.62**
Doddle	**0.37**	**0.43**	**0.44**	**0.52**
SBR	***0.53***	***0.53***	***0.58***	***0.58***

Table D.67: SYM-236 average $\mathcal{R}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.07	0.18	0.06	0.06	0.09	0.12	0.03	0.09	0.11	0.14	0.08	0.08	*0.14*
10	0.25	0.37	0.21	0.10	0.29	0.25	0.17	0.24	0.27	0.22	0.18	0.27	*0.21*
20	0.33	0.47	0.29	0.15	0.40	0.33	0.24	0.31	0.37	0.31	0.25	0.37	*0.26*
30	0.41	0.55	0.37	0.19	0.48	0.41	0.30	0.39	0.45	0.38	0.30	0.44	*0.34*
40	0.47	0.60	0.43	0.24	0.54	0.48	0.35	0.46	0.52	0.46	0.36	0.51	*0.44*
50	0.53	0.64	0.48	0.29	0.59	0.55	0.41	0.51	0.57	0.52	0.41	0.56	*0.52*
60	0.57	0.69	0.53	0.35	0.63	0.60	0.46	0.56	0.61	0.57	0.46	0.60	*0.55*
70	0.62	0.73	0.58	0.42	0.67	0.65	0.51	0.61	0.65	0.63	0.51	0.65	*0.56*
80	0.67	0.77	0.63	0.49	0.71	0.69	0.56	0.65	0.69	0.68	0.55	0.70	*0.62*
90	0.72	0.81	0.68	0.57	0.76	0.74	0.62	0.70	0.74	0.73	0.61	0.74	*0.69*
100	0.76	0.85	0.73	0.65	0.79	0.79	0.68	0.75	0.77	0.77	0.67	0.78	*0.77*
110	0.80	0.87	0.77	0.71	0.83	0.82	0.74	0.79	0.81	0.82	0.73	0.82	*0.83*
120	0.84	0.90	0.81	0.76	0.86	0.86	0.79	0.82	0.85	0.86	0.78	0.85	*0.89*
130	0.87	0.92	0.84	0.81	0.89	0.90	0.83	0.86	0.88	0.89	0.81	0.88	*0.92*
140	0.90	0.93	0.87	0.85	0.91	0.92	0.86	0.89	0.90	0.91	0.85	0.91	*0.94*
150	0.92	0.95	0.90	0.89	0.93	0.94	0.90	0.91	0.92	0.94	0.88	0.92	*0.97*
160	0.94	0.96	0.92	0.93	0.94	0.96	0.92	0.93	0.93	0.95	0.91	0.94	*0.98*
170	0.95	0.97	0.94	0.95	0.96	0.97	0.95	0.95	0.95	0.97	0.93	0.95	*0.99*
180	0.97	0.98	0.96	0.97	0.97	0.98	0.96	0.96	0.96	0.98	0.95	0.97	*1.00*
190	0.98	0.99	0.97	0.98	0.98	0.99	0.98	0.97	0.97	0.99	0.97	0.98	*1.00*
200	0.99	1.00	0.98	0.99	0.99	1.00	0.99	0.99	0.98	0.99	0.98	0.99	*1.00*
210	1.00	1.00	0.99	1.00	0.99	1.00	0.99	0.99	0.99	1.00	0.99	1.00	*1.00*
220	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
230	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.68: SYM-236 average $\mathcal{R}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.35	0.37	0.18	0.32	0.32	0.30	0.23	0.31	0.02	0.01	0.01	0.13	0.07	*0.14*
10	0.42	0.47	0.29	0.43	0.42	0.40	0.35	0.42	0.03	0.03	0.03	0.20	0.25	*0.21*
20	0.50	0.55	0.37	0.51	0.50	0.49	0.44	0.50	0.04	0.05	0.05	0.25	0.33	*0.26*
30	0.56	0.61	0.44	0.57	0.56	0.56	0.50	0.56	0.05	0.07	0.07	0.31	0.41	*0.34*
40	0.61	0.66	0.50	0.63	0.61	0.62	0.56	0.61	0.05	0.09	0.09	0.37	0.47	*0.44*
50	0.66	0.71	0.56	0.68	0.66	0.68	0.61	0.66	0.06	0.13	0.13	0.44	0.53	*0.52*
60	0.71	0.75	0.61	0.73	0.70	0.73	0.65	0.70	0.08	0.17	0.17	0.50	0.57	*0.55*
70	0.75	0.79	0.66	0.77	0.74	0.77	0.70	0.73	0.10	0.22	0.23	0.55	0.62	*0.56*
80	0.79	0.83	0.71	0.81	0.78	0.81	0.75	0.77	0.12	0.28	0.29	0.60	0.67	*0.62*
90	0.82	0.86	0.75	0.84	0.82	0.85	0.79	0.81	0.16	0.35	0.36	0.66	0.72	*0.69*
100	0.85	0.89	0.80	0.87	0.85	0.87	0.83	0.85	0.20	0.42	0.43	0.72	0.76	*0.77*
110	0.88	0.91	0.84	0.89	0.88	0.89	0.87	0.88	0.24	0.50	0.51	0.78	0.80	*0.83*
120	0.90	0.93	0.88	0.92	0.90	0.92	0.90	0.90	0.29	0.58	0.59	0.84	0.84	*0.89*
130	0.92	0.94	0.91	0.93	0.92	0.94	0.92	0.92	0.35	0.66	0.67	0.90	0.87	*0.92*
140	0.94	0.96	0.94	0.95	0.94	0.96	0.94	0.94	0.41	0.75	0.75	0.94	0.90	*0.94*
150	0.95	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.48	0.81	0.82	0.97	0.92	*0.97*
160	0.96	0.98	0.97	0.98	0.98	0.99	0.98	0.98	0.55	0.87	0.88	0.98	0.94	*0.98*
170	0.97	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.63	0.92	0.93	0.99	0.95	*0.99*
180	0.98	1.00	0.99	0.99	0.99	1.00	1.00	0.99	0.70	0.97	0.97	0.99	0.97	*1.00*
190	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.78	0.99	0.99	1.00	0.98	*1.00*
200	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.86	1.00	1.00	1.00	0.99	*1.00*
210	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.92	1.00	1.00	1.00	1.00	*1.00*
220	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97	1.00	1.00	1.00	1.00	*1.00*
230	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.69: SYM-236 average $\mathcal{R}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.17	0.27	0.14	0.10	0.21	0.19	0.16	0.10	0.12	0.10	0.13	0.18	*0.14*
10	0.31	0.43	0.25	0.14	0.36	0.31	0.24	0.22	0.26	0.21	0.21	0.35	*0.21*
20	0.40	0.53	0.37	0.20	0.47	0.39	0.31	0.31	0.38	0.30	0.29	0.44	*0.26*
30	0.47	0.58	0.46	0.24	0.54	0.47	0.37	0.39	0.47	0.37	0.35	0.51	*0.34*
40	0.55	0.63	0.52	0.30	0.59	0.55	0.44	0.44	0.53	0.45	0.41	0.58	*0.44*
50	0.60	0.67	0.56	0.36	0.64	0.60	0.50	0.50	0.57	0.52	0.47	0.62	*0.52*
60	0.65	0.71	0.61	0.43	0.67	0.66	0.56	0.55	0.61	0.58	0.53	0.66	*0.55*
70	0.69	0.76	0.66	0.50	0.72	0.71	0.61	0.60	0.66	0.63	0.57	0.71	*0.56*
80	0.73	0.80	0.70	0.58	0.76	0.75	0.66	0.65	0.70	0.69	0.62	0.74	*0.62*
90	0.77	0.84	0.74	0.65	0.80	0.79	0.71	0.69	0.75	0.74	0.66	0.79	*0.69*
100	0.81	0.87	0.78	0.72	0.84	0.83	0.75	0.74	0.79	0.78	0.70	0.82	*0.77*
110	0.85	0.90	0.82	0.78	0.87	0.86	0.80	0.78	0.83	0.82	0.75	0.86	*0.83*
120	0.88	0.92	0.85	0.83	0.90	0.89	0.84	0.81	0.86	0.86	0.79	0.89	*0.89*
130	0.90	0.94	0.87	0.87	0.92	0.92	0.87	0.85	0.89	0.89	0.83	0.91	*0.92*
140	0.93	0.96	0.90	0.90	0.94	0.94	0.90	0.88	0.91	0.91	0.86	0.93	*0.94*
150	0.95	0.97	0.92	0.93	0.95	0.96	0.92	0.91	0.93	0.94	0.89	0.95	*0.97*
160	0.96	0.98	0.94	0.95	0.96	0.97	0.95	0.93	0.95	0.96	0.91	0.96	*0.98*
170	0.97	0.99	0.95	0.97	0.97	0.98	0.96	0.94	0.96	0.97	0.94	0.97	*0.99*
180	0.98	0.99	0.97	0.98	0.98	0.99	0.97	0.96	0.97	0.98	0.96	0.98	*1.00*
190	0.99	1.00	0.98	0.99	0.99	1.00	0.98	0.97	0.98	0.99	0.97	0.99	*1.00*
200	0.99	1.00	0.98	1.00	0.99	1.00	0.99	0.98	0.99	0.99	0.98	0.99	*1.00*
210	1.00	1.00	0.99	1.00	1.00	1.00	1.00	0.99	0.99	1.00	0.99	1.00	*1.00*
220	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
230	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.70: SYM-236 average $\mathcal{R}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.24	0.38	0.38	0.39	0.33	0.40	0.27	0.33	0.01	0.03	0.06	0.22	0.17	*0.14*
10	0.35	0.48	0.46	0.47	0.43	0.46	0.36	0.44	0.03	0.07	0.09	0.27	0.31	*0.21*
20	0.39	0.55	0.54	0.54	0.52	0.54	0.44	0.51	0.05	0.10	0.15	0.36	0.40	*0.26*
30	0.44	0.60	0.60	0.60	0.57	0.60	0.51	0.56	0.05	0.14	0.20	0.43	0.47	*0.34*
40	0.47	0.65	0.65	0.65	0.62	0.65	0.56	0.61	0.07	0.18	0.25	0.50	0.55	*0.44*
50	0.49	0.69	0.70	0.69	0.67	0.70	0.60	0.66	0.08	0.23	0.31	0.57	0.60	*0.52*
60	0.52	0.74	0.75	0.74	0.71	0.75	0.65	0.70	0.10	0.28	0.37	0.62	0.65	*0.55*
70	0.55	0.77	0.79	0.78	0.75	0.79	0.69	0.74	0.12	0.34	0.42	0.66	0.69	*0.56*
80	0.57	0.81	0.83	0.82	0.79	0.82	0.74	0.78	0.14	0.41	0.49	0.71	0.73	*0.62*
90	0.62	0.85	0.86	0.85	0.83	0.86	0.78	0.82	0.17	0.48	0.56	0.77	0.77	*0.69*
100	0.65	0.89	0.89	0.88	0.86	0.88	0.83	0.86	0.20	0.55	0.63	0.81	0.81	*0.77*
110	0.67	0.91	0.91	0.90	0.89	0.90	0.87	0.89	0.24	0.62	0.69	0.85	0.85	*0.83*
120	0.69	0.93	0.93	0.92	0.91	0.91	0.90	0.91	0.28	0.69	0.76	0.89	0.88	*0.89*
130	0.72	0.95	0.95	0.93	0.93	0.94	0.92	0.93	0.32	0.77	0.82	0.92	0.90	*0.92*
140	0.74	0.96	0.97	0.95	0.94	0.96	0.94	0.94	0.36	0.83	0.87	0.94	0.93	*0.94*
150	0.77	0.98	0.98	0.97	0.97	0.97	0.97	0.97	0.41	0.88	0.92	0.97	0.95	*0.97*
160	0.79	0.98	0.98	0.98	0.98	0.99	0.98	0.98	0.47	0.93	0.96	0.98	0.96	*0.98*
170	0.80	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.53	0.97	0.98	0.99	0.97	*0.99*
180	0.83	1.00	1.00	0.99	0.99	1.00	0.99	1.00	0.60	0.99	1.00	0.99	0.98	*1.00*
190	0.86	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.67	1.00	1.00	1.00	0.99	*1.00*
200	0.88	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.74	1.00	1.00	1.00	0.99	*1.00*
210	0.89	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.81	1.00	1.00	1.00	1.00	*1.00*
220	0.90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.90	1.00	1.00	1.00	1.00	*1.00*
230	0.96	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97	1.00	1.00	1.00	1.00	*1.00*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.71: SYM-236 average $\hat{\mathcal{R}}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.01	0.01	0.00	0.00	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.01	*0.01*
10	0.10	0.15	0.08	0.04	0.12	0.10	0.07	0.09	0.11	0.09	0.07	0.11	*0.08*
20	0.20	0.28	0.17	0.09	0.24	0.20	0.14	0.19	0.22	0.19	0.15	0.22	*0.15*
30	0.30	0.40	0.27	0.14	0.35	0.30	0.22	0.29	0.33	0.28	0.22	0.32	*0.25*
40	0.39	0.50	0.35	0.20	0.44	0.40	0.29	0.38	0.43	0.38	0.30	0.42	*0.36*
50	0.47	0.57	0.42	0.26	0.52	0.49	0.37	0.46	0.51	0.46	0.37	0.50	*0.46*
60	0.54	0.64	0.49	0.32	0.59	0.56	0.43	0.53	0.57	0.53	0.43	0.56	*0.51*
70	0.60	0.70	0.56	0.40	0.64	0.62	0.49	0.58	0.63	0.61	0.49	0.62	*0.54*
80	0.66	0.75	0.62	0.47	0.70	0.68	0.55	0.64	0.68	0.67	0.54	0.68	*0.61*
90	0.71	0.80	0.68	0.56	0.75	0.73	0.61	0.69	0.73	0.72	0.60	0.73	*0.69*
100	0.76	0.84	0.73	0.65	0.79	0.78	0.68	0.74	0.77	0.77	0.67	0.77	*0.76*
110	0.80	0.87	0.77	0.71	0.83	0.82	0.74	0.78	0.81	0.82	0.73	0.81	*0.83*
120	0.84	0.90	0.81	0.76	0.86	0.86	0.79	0.82	0.84	0.85	0.78	0.85	*0.89*
130	0.87	0.92	0.84	0.81	0.89	0.89	0.83	0.86	0.88	0.89	0.81	0.88	*0.92*
140	0.90	0.93	0.87	0.85	0.91	0.92	0.86	0.89	0.90	0.91	0.85	0.91	*0.94*
150	0.92	0.95	0.90	0.89	0.93	0.94	0.90	0.91	0.92	0.94	0.88	0.92	*0.97*
160	0.94	0.96	0.92	0.93	0.94	0.96	0.92	0.93	0.93	0.95	0.91	0.94	*0.98*
170	0.95	0.97	0.94	0.95	0.96	0.97	0.95	0.95	0.95	0.97	0.93	0.95	*0.99*
180	0.97	0.98	0.96	0.97	0.97	0.98	0.96	0.96	0.96	0.98	0.95	0.97	*1.00*
190	0.98	0.99	0.97	0.98	0.98	0.99	0.98	0.97	0.97	0.99	0.97	0.98	*1.00*
200	0.99	1.00	0.98	0.99	0.99	1.00	0.99	0.99	0.98	0.99	0.98	0.99	*1.00*
210	1.00	1.00	0.99	1.00	0.99	1.00	0.99	0.99	0.99	1.00	0.99	1.00	*1.00*
220	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
230	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.72: SYM-236 average $\hat{\mathcal{R}}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.02	0.02	0.01	0.02	0.02	0.02	0.01	0.02	0.00	0.00	0.00	0.01	0.01	*0.01*
10	0.16	0.18	0.11	0.17	0.16	0.16	0.14	0.16	0.01	0.01	0.01	0.08	0.10	*0.08*
20	0.30	0.33	0.21	0.31	0.30	0.29	0.26	0.30	0.02	0.03	0.03	0.14	0.20	*0.15*
30	0.41	0.44	0.31	0.42	0.41	0.41	0.37	0.41	0.04	0.05	0.05	0.23	0.30	*0.25*
40	0.50	0.54	0.41	0.52	0.50	0.50	0.46	0.50	0.04	0.07	0.07	0.31	0.39	*0.36*
50	0.58	0.62	0.49	0.60	0.58	0.60	0.54	0.58	0.06	0.11	0.11	0.39	0.47	*0.46*
60	0.66	0.70	0.57	0.67	0.65	0.68	0.61	0.65	0.07	0.15	0.16	0.46	0.54	*0.51*
70	0.72	0.76	0.63	0.73	0.71	0.74	0.67	0.70	0.10	0.21	0.21	0.53	0.60	*0.54*
80	0.77	0.81	0.70	0.79	0.77	0.79	0.73	0.76	0.12	0.27	0.28	0.59	0.66	*0.61*
90	0.81	0.85	0.74	0.83	0.81	0.84	0.78	0.80	0.16	0.35	0.35	0.65	0.71	*0.69*
100	0.84	0.88	0.80	0.86	0.85	0.87	0.83	0.84	0.20	0.42	0.43	0.72	0.76	*0.76*
110	0.88	0.91	0.84	0.89	0.88	0.89	0.86	0.88	0.24	0.49	0.51	0.78	0.80	*0.83*
120	0.90	0.93	0.88	0.91	0.90	0.92	0.90	0.90	0.29	0.58	0.59	0.84	0.84	*0.89*
130	0.92	0.94	0.91	0.93	0.92	0.94	0.92	0.92	0.35	0.66	0.67	0.90	0.87	*0.92*
140	0.94	0.96	0.94	0.95	0.94	0.96	0.94	0.94	0.41	0.75	0.75	0.94	0.90	*0.94*
150	0.95	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.48	0.81	0.82	0.97	0.92	*0.97*
160	0.96	0.98	0.97	0.98	0.98	0.99	0.98	0.98	0.55	0.87	0.88	0.98	0.94	*0.98*
170	0.97	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.63	0.92	0.93	0.99	0.95	*0.99*
180	0.98	1.00	0.99	0.99	0.99	1.00	1.00	0.99	0.70	0.97	0.97	0.99	0.97	*1.00*
190	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.78	0.99	0.99	1.00	0.98	*1.00*
200	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.86	1.00	1.00	1.00	0.99	*1.00*
210	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.92	1.00	1.00	1.00	1.00	*1.00*
220	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97	1.00	1.00	1.00	1.00	*1.00*
230	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.73: SYM-236 average $\hat{\mathcal{R}}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.01	0.02	0.01	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	*0.01*
10	0.13	0.17	0.10	0.06	0.15	0.12	0.10	0.09	0.11	0.09	0.09	0.14	*0.08*
20	0.25	0.32	0.23	0.12	0.28	0.24	0.19	0.19	0.23	0.18	0.18	0.26	*0.15*
30	0.35	0.43	0.34	0.18	0.40	0.35	0.28	0.29	0.35	0.28	0.26	0.38	*0.25*
40	0.45	0.51	0.43	0.25	0.49	0.45	0.37	0.37	0.44	0.38	0.35	0.48	*0.36*
50	0.53	0.59	0.50	0.32	0.56	0.53	0.45	0.45	0.51	0.47	0.42	0.55	*0.46*
60	0.60	0.66	0.57	0.40	0.63	0.61	0.52	0.52	0.57	0.55	0.49	0.62	*0.51*
70	0.67	0.73	0.63	0.48	0.69	0.68	0.59	0.58	0.63	0.61	0.55	0.68	*0.54*
80	0.72	0.78	0.68	0.56	0.74	0.73	0.65	0.64	0.69	0.67	0.60	0.73	*0.61*
90	0.76	0.83	0.73	0.64	0.79	0.78	0.70	0.69	0.74	0.73	0.65	0.78	*0.69*
100	0.81	0.87	0.78	0.72	0.83	0.82	0.75	0.74	0.79	0.78	0.70	0.82	*0.76*
110	0.84	0.90	0.82	0.78	0.86	0.86	0.80	0.78	0.83	0.82	0.75	0.86	*0.83*
120	0.88	0.92	0.85	0.83	0.90	0.89	0.84	0.81	0.86	0.86	0.79	0.89	*0.89*
130	0.90	0.94	0.87	0.87	0.92	0.92	0.87	0.85	0.89	0.89	0.83	0.91	*0.92*
140	0.93	0.96	0.90	0.90	0.94	0.94	0.90	0.88	0.91	0.91	0.86	0.93	*0.94*
150	0.95	0.97	0.92	0.93	0.95	0.96	0.92	0.91	0.93	0.94	0.89	0.95	*0.97*
160	0.96	0.98	0.94	0.95	0.96	0.97	0.95	0.93	0.95	0.96	0.91	0.96	*0.98*
170	0.97	0.99	0.95	0.97	0.97	0.98	0.96	0.94	0.96	0.97	0.94	0.97	*0.99*
180	0.98	0.99	0.97	0.98	0.98	0.99	0.97	0.96	0.97	0.98	0.96	0.98	*1.00*
190	0.99	1.00	0.98	0.99	0.99	1.00	0.98	0.97	0.98	0.99	0.97	0.99	*1.00*
200	0.99	1.00	0.98	1.00	0.99	1.00	0.99	0.98	0.99	0.99	0.98	0.99	*1.00*
210	1.00	1.00	0.99	1.00	1.00	1.00	1.00	0.99	0.99	1.00	0.99	1.00	*1.00*
220	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
230	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.74: SYM-236 average $\hat{\mathcal{R}}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.02	0.03	0.03	0.03	0.02	0.03	0.02	0.02	0.00	0.00	0.00	0.02	0.01	*0.01*
10	0.13	0.19	0.18	0.19	0.17	0.18	0.14	0.18	0.01	0.02	0.04	0.11	0.13	*0.08*
20	0.23	0.32	0.32	0.32	0.31	0.33	0.26	0.30	0.03	0.06	0.09	0.21	0.25	*0.15*
30	0.32	0.44	0.44	0.44	0.42	0.43	0.38	0.41	0.04	0.10	0.15	0.31	0.35	*0.25*
40	0.38	0.53	0.54	0.53	0.51	0.53	0.46	0.50	0.05	0.15	0.20	0.41	0.45	*0.36*
50	0.43	0.61	0.62	0.61	0.59	0.62	0.54	0.58	0.07	0.20	0.27	0.50	0.53	*0.46*
60	0.48	0.69	0.69	0.69	0.66	0.70	0.60	0.65	0.09	0.26	0.34	0.57	0.60	*0.51*
70	0.53	0.74	0.76	0.75	0.72	0.75	0.67	0.71	0.11	0.33	0.40	0.64	0.67	*0.54*
80	0.56	0.80	0.81	0.80	0.77	0.80	0.73	0.76	0.14	0.40	0.47	0.70	0.72	*0.61*
90	0.61	0.84	0.85	0.84	0.82	0.85	0.77	0.81	0.17	0.47	0.55	0.76	0.76	*0.69*
100	0.65	0.88	0.88	0.88	0.86	0.88	0.82	0.85	0.20	0.54	0.63	0.81	0.81	*0.76*
110	0.67	0.91	0.91	0.90	0.89	0.90	0.86	0.89	0.24	0.62	0.69	0.85	0.84	*0.83*
120	0.69	0.93	0.93	0.92	0.91	0.91	0.90	0.91	0.28	0.69	0.76	0.89	0.88	*0.89*
130	0.72	0.95	0.95	0.93	0.92	0.94	0.92	0.93	0.32	0.77	0.82	0.92	0.90	*0.92*
140	0.74	0.96	0.97	0.95	0.94	0.96	0.94	0.94	0.36	0.83	0.87	0.94	0.93	*0.94*
150	0.77	0.98	0.98	0.97	0.97	0.97	0.97	0.97	0.41	0.88	0.92	0.97	0.95	*0.97*
160	0.79	0.98	0.98	0.98	0.98	0.99	0.98	0.98	0.47	0.93	0.96	0.98	0.96	*0.98*
170	0.80	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.53	0.97	0.98	0.99	0.97	*0.99*
180	0.83	1.00	1.00	0.99	0.99	1.00	0.99	1.00	0.60	0.99	1.00	0.99	0.98	*1.00*
190	0.86	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.67	1.00	1.00	1.00	0.99	*1.00*
200	0.88	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.74	1.00	1.00	1.00	0.99	*1.00*
210	0.89	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.81	1.00	1.00	1.00	1.00	*1.00*
220	0.90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.90	1.00	1.00	1.00	1.00	*1.00*
230	0.96	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97	1.00	1.00	1.00	1.00	*1.00*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.75: SYM-236 average $\mathcal{P}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.21	0.39	0.15	0.21	0.25	0.29	0.12	0.25	0.27	0.31	0.20	0.23	*0.78*
10	0.40	0.59	0.34	0.21	0.46	0.41	0.28	0.38	0.44	0.35	0.29	0.43	*0.73*
20	0.45	0.65	0.40	0.23	0.54	0.45	0.32	0.42	0.51	0.42	0.33	0.50	*0.72*
30	0.49	0.66	0.45	0.25	0.57	0.49	0.35	0.46	0.55	0.46	0.35	0.53	*0.71*
40	0.50	0.66	0.47	0.28	0.58	0.52	0.38	0.48	0.56	0.49	0.38	0.54	*0.72*
50	0.51	0.64	0.47	0.30	0.57	0.53	0.39	0.49	0.55	0.50	0.38	0.54	*0.69*
60	0.50	0.63	0.47	0.32	0.56	0.52	0.40	0.49	0.54	0.51	0.39	0.53	*0.63*
70	0.50	0.61	0.46	0.34	0.54	0.52	0.40	0.48	0.53	0.51	0.39	0.52	*0.57*
80	0.49	0.60	0.46	0.36	0.53	0.52	0.41	0.47	0.52	0.51	0.40	0.51	*0.58*
90	0.49	0.59	0.45	0.39	0.52	0.51	0.41	0.47	0.50	0.50	0.40	0.50	*0.59*
100	0.48	0.57	0.45	0.41	0.51	0.51	0.42	0.46	0.50	0.50	0.40	0.50	*0.59*
110	0.47	0.56	0.45	0.42	0.50	0.50	0.42	0.46	0.49	0.50	0.41	0.49	*0.59*
120	0.47	0.54	0.44	0.43	0.49	0.49	0.43	0.45	0.48	0.49	0.41	0.48	*0.59*
130	0.46	0.52	0.43	0.43	0.47	0.48	0.42	0.44	0.46	0.48	0.41	0.47	*0.56*
140	0.45	0.49	0.43	0.43	0.46	0.47	0.42	0.44	0.45	0.46	0.41	0.45	*0.53*
150	0.44	0.47	0.42	0.43	0.44	0.46	0.42	0.43	0.44	0.45	0.40	0.44	*0.50*
160	0.42	0.45	0.41	0.42	0.43	0.44	0.41	0.42	0.42	0.44	0.40	0.43	*0.47*
170	0.41	0.43	0.40	0.42	0.41	0.42	0.40	0.41	0.41	0.42	0.39	0.41	*0.45*
180	0.40	0.41	0.39	0.40	0.40	0.41	0.39	0.40	0.40	0.41	0.39	0.40	*0.42*
190	0.39	0.40	0.38	0.39	0.39	0.39	0.38	0.38	0.38	0.39	0.38	0.39	*0.40*
200	0.37	0.38	0.37	0.38	0.37	0.38	0.37	0.37	0.37	0.38	0.37	0.37	*0.38*
210	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36	*0.37*
220	0.35	0.35	0.34	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	*0.35*
230	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	*0.33*
236	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	*0.32*

Table D.76: SYM-236 average $\mathcal{P}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.79	0.90	0.50	0.85	0.85	0.85	0.81	0.85	0.09	0.07	0.07	0.55	0.21	*0.78*
10	0.76	0.83	0.56	0.83	0.81	0.82	0.76	0.82	0.10	0.11	0.11	0.61	0.40	*0.73*
20	0.76	0.83	0.59	0.83	0.81	0.81	0.76	0.81	0.09	0.15	0.15	0.59	0.45	*0.72*
30	0.74	0.81	0.62	0.81	0.79	0.81	0.75	0.79	0.09	0.18	0.18	0.60	0.49	*0.71*
40	0.73	0.79	0.63	0.78	0.76	0.79	0.73	0.76	0.09	0.21	0.21	0.60	0.50	*0.72*
50	0.70	0.78	0.63	0.76	0.74	0.78	0.70	0.74	0.10	0.25	0.25	0.61	0.51	*0.69*
60	0.68	0.76	0.63	0.74	0.71	0.76	0.68	0.71	0.11	0.28	0.29	0.60	0.50	*0.63*
70	0.67	0.74	0.62	0.72	0.69	0.74	0.66	0.68	0.12	0.32	0.32	0.59	0.50	*0.57*
80	0.65	0.72	0.61	0.69	0.67	0.71	0.65	0.66	0.13	0.35	0.35	0.58	0.49	*0.58*
90	0.63	0.69	0.60	0.67	0.65	0.69	0.64	0.65	0.15	0.37	0.38	0.58	0.49	*0.59*
100	0.61	0.66	0.59	0.64	0.63	0.66	0.62	0.63	0.17	0.40	0.40	0.57	0.48	*0.59*
110	0.58	0.63	0.57	0.62	0.61	0.63	0.60	0.61	0.19	0.41	0.42	0.57	0.47	*0.59*
120	0.56	0.60	0.55	0.59	0.58	0.60	0.58	0.58	0.21	0.43	0.44	0.56	0.47	*0.59*
130	0.53	0.56	0.53	0.56	0.56	0.56	0.55	0.56	0.23	0.44	0.45	0.55	0.46	*0.56*
140	0.51	0.53	0.51	0.53	0.53	0.53	0.53	0.53	0.25	0.45	0.45	0.53	0.45	*0.53*
150	0.48	0.50	0.49	0.50	0.50	0.50	0.50	0.50	0.27	0.45	0.45	0.50	0.44	*0.50*
160	0.46	0.47	0.47	0.47	0.47	0.48	0.47	0.47	0.29	0.44	0.44	0.48	0.42	*0.47*
170	0.44	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.30	0.43	0.43	0.45	0.41	*0.45*
180	0.42	0.43	0.42	0.43	0.43	0.43	0.43	0.43	0.32	0.42	0.42	0.43	0.40	*0.42*
190	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.33	0.40	0.40	0.40	0.39	*0.40*
200	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.34	0.38	0.38	0.38	0.37	*0.38*
210	0.36	0.37	0.37	0.37	0.37	0.37	0.37	0.37	0.34	0.37	0.37	0.37	0.36	*0.37*
220	0.34	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.34	0.35	0.35	0.35	0.35	*0.35*
230	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	*0.33*
236	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	*0.32*

Table D.77: SYM-236 average $\mathcal{P}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.37	0.55	0.29	0.34	0.39	0.46	0.36	0.29	0.28	0.35	0.34	0.37	*0.78*
10	0.49	0.66	0.40	0.29	0.54	0.50	0.40	0.37	0.42	0.37	0.37	0.53	*0.73*
20	0.53	0.71	0.49	0.32	0.61	0.53	0.42	0.42	0.50	0.41	0.40	0.57	*0.72*
30	0.56	0.71	0.53	0.32	0.64	0.55	0.43	0.44	0.54	0.44	0.41	0.60	*0.71*
40	0.57	0.69	0.54	0.34	0.63	0.57	0.46	0.45	0.55	0.47	0.42	0.61	*0.72*
50	0.57	0.68	0.53	0.36	0.62	0.58	0.46	0.46	0.54	0.48	0.43	0.60	*0.69*
60	0.57	0.67	0.53	0.38	0.61	0.58	0.47	0.46	0.53	0.49	0.43	0.59	*0.63*
70	0.56	0.65	0.53	0.40	0.60	0.58	0.47	0.46	0.53	0.50	0.42	0.58	*0.57*
80	0.55	0.64	0.52	0.42	0.59	0.56	0.47	0.46	0.53	0.50	0.42	0.57	*0.58*
90	0.54	0.62	0.51	0.44	0.57	0.55	0.47	0.46	0.52	0.49	0.42	0.55	*0.59*
100	0.53	0.61	0.50	0.45	0.56	0.55	0.47	0.45	0.51	0.49	0.42	0.54	*0.59*
110	0.52	0.58	0.49	0.46	0.54	0.53	0.47	0.45	0.50	0.49	0.42	0.53	*0.59*
120	0.50	0.56	0.48	0.46	0.52	0.52	0.46	0.45	0.49	0.48	0.42	0.51	*0.59*
130	0.49	0.53	0.46	0.46	0.50	0.50	0.46	0.44	0.48	0.47	0.42	0.49	*0.56*
140	0.47	0.51	0.45	0.46	0.48	0.49	0.45	0.43	0.46	0.46	0.41	0.47	*0.53*
150	0.45	0.48	0.43	0.45	0.46	0.47	0.44	0.42	0.44	0.45	0.41	0.46	*0.50*
160	0.44	0.46	0.42	0.44	0.44	0.45	0.43	0.41	0.43	0.43	0.40	0.44	*0.47*
170	0.42	0.44	0.41	0.42	0.42	0.43	0.41	0.40	0.41	0.42	0.40	0.42	*0.45*
180	0.41	0.42	0.39	0.41	0.41	0.42	0.40	0.39	0.40	0.41	0.39	0.41	*0.42*
190	0.39	0.40	0.38	0.40	0.39	0.40	0.39	0.38	0.39	0.39	0.38	0.39	*0.40*
200	0.38	0.38	0.37	0.38	0.38	0.38	0.37	0.37	0.37	0.38	0.37	0.38	*0.38*
210	0.36	0.37	0.36	0.37	0.36	0.37	0.36	0.36	0.36	0.36	0.36	0.36	*0.37*
220	0.35	0.35	0.34	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	*0.35*
230	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	*0.33*
236	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	*0.32*

Table D.78: SYM-236 average $\mathcal{P}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.66	0.89	0.84	0.90	0.81	0.89	0.83	0.84	0.06	0.22	0.32	0.78	0.37	*0.78*
10	0.67	0.84	0.80	0.86	0.81	0.86	0.79	0.83	0.07	0.25	0.33	0.75	0.49	*0.73*
20	0.64	0.82	0.79	0.84	0.81	0.85	0.78	0.82	0.09	0.30	0.39	0.75	0.53	*0.72*
30	0.65	0.81	0.78	0.82	0.79	0.84	0.77	0.80	0.09	0.33	0.43	0.73	0.56	*0.71*
40	0.62	0.79	0.77	0.80	0.77	0.81	0.74	0.77	0.10	0.36	0.45	0.72	0.57	*0.72*
50	0.58	0.77	0.76	0.78	0.75	0.79	0.71	0.75	0.11	0.39	0.48	0.71	0.57	*0.69*
60	0.57	0.75	0.74	0.76	0.73	0.78	0.68	0.72	0.12	0.42	0.49	0.69	0.57	*0.63*
70	0.55	0.73	0.72	0.74	0.71	0.75	0.66	0.70	0.12	0.44	0.50	0.67	0.56	*0.57*
80	0.52	0.71	0.70	0.71	0.69	0.73	0.65	0.68	0.13	0.46	0.51	0.65	0.55	*0.58*
90	0.52	0.68	0.68	0.69	0.67	0.71	0.63	0.66	0.15	0.47	0.52	0.64	0.54	*0.59*
100	0.50	0.66	0.65	0.66	0.65	0.67	0.62	0.64	0.16	0.48	0.52	0.62	0.53	*0.59*
110	0.48	0.63	0.62	0.63	0.62	0.63	0.60	0.62	0.18	0.49	0.52	0.60	0.52	*0.59*
120	0.46	0.60	0.59	0.60	0.59	0.60	0.58	0.59	0.19	0.49	0.52	0.58	0.50	*0.59*
130	0.44	0.57	0.56	0.56	0.56	0.56	0.55	0.56	0.21	0.49	0.51	0.56	0.49	*0.56*
140	0.43	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.22	0.49	0.50	0.53	0.47	*0.53*
150	0.41	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.24	0.48	0.49	0.50	0.45	*0.50*
160	0.39	0.47	0.47	0.47	0.47	0.48	0.47	0.47	0.25	0.46	0.47	0.48	0.44	*0.47*
170	0.38	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.26	0.44	0.45	0.45	0.42	*0.45*
180	0.37	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.27	0.42	0.43	0.43	0.41	*0.42*
190	0.35	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.29	0.40	0.40	0.40	0.39	*0.40*
200	0.35	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.30	0.38	0.38	0.38	0.38	*0.38*
210	0.33	0.37	0.37	0.37	0.37	0.37	0.37	0.37	0.31	0.37	0.37	0.37	0.36	*0.37*
220	0.32	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.32	0.35	0.35	0.35	0.35	*0.35*
230	0.32	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	*0.33*
236	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	*0.32*

Table D.79: SYM-236 average precision within the top 5 collections, and the number of queries (out of 100) for which the first ranked collection is correct, for the various configurations of the Doddle algorithm.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Q_s	Q_l	Q_s		Q_l	
Doddle	0.10	0.12	2	*(2%)*	6	*(6%)*
Doddle_RC	0.14	0.17	7	*(7%)*	12	*(12%)*
Doddle_RP	0.09	0.10	1	*(1%)*	6	*(6%)*
Doddle_RF	0.04	0.05	0	*(0%)*	0	*(0%)*
Doddle_RC+RP	0.11	0.13	3	*(3%)*	11	*(11%)*
Doddle_RC+RF	0.10	0.12	3	*(3%)*	5	*(5%)*
Doddle_RP+RF	0.07	0.10	0	*(0%)*	5	*(5%)*
Doddle_×	0.09	0.08	4	*(4%)*	2	*(2%)*
Doddle_RC×RP	0.10	0.10	6	*(6%)*	5	*(5%)*
Doddle_RC×RF	0.08	0.07	4	*(4%)*	2	*(2%)*
Doddle_RP×RF	0.08	0.08	3	*(3%)*	3	*(3%)*
Doddle_W	0.11	0.13	2	*(2%)*	7	*(7%)*
SBR	*0.03*	*0.03*	*2*	*(2%)*	*2*	*(2%)*

Table D.80: SYM-236 average precision within the top 5 collections, and the number of queries (out of 100) for which the first ranked collection is correct, for existing algorithms and Doddle.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Q_s	Q_l	Q_s		Q_l	
bGlOSS	0.18	0.11	8	*(8%)*	2	*(2%)*
CORI	0.17	0.18	9	*(9%)*	12	*(12%)*
Cosine Measure	0.09	0.17	3	*(3%)*	13	*(13%)*
Inner Product	0.14	0.17	8	*(8%)*	13	*(13%)*
Skew	0.14	0.16	8	*(8%)*	9	*(9%)*
Highest-available Similarity	0.14	0.17	6	*(6%)*	15	*(15%)*
CVV	0.09	0.10	3	*(3%)*	7	*(7%)*
DFPROP	0.15	0.16	8	*(8%)*	9	*(9%)*
Distribution of Informative Amt	0.00	0.01	0	*(0%)*	0	*(0%)*
SCS	0.00	0.01	0	*(0%)*	0	*(0%)*
AvICTF	0.00	0.01	0	*(0%)*	1	*(1%)*
NSCQ	0.03	0.06	3	*(3%)*	5	*(5%)*
Doddle	0.10	0.12	2	*(2%)*	6	*(6%)*
SBR	*0.03*	*0.03*	*2*	*(2%)*	*2*	*(2%)*

D.4 UDC-236

Table D.81: UDC-236 average Spearman rank correlations, for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		**SBR**	
Algorithms	Q_s	Q_l	Q_s	Q_l
Doddle	**0.39**	**0.43**	−0.04	−0.07
Doddle_RC	**0.42**	**0.48**	−0.12	−0.16
Doddle_RP	**0.41**	**0.45**	−0.15	−0.18
Doddle_RF	**0.21**	**0.31**	**0.24**	**0.20**
Doddle_RC+RP	**0.42**	**0.47**	−0.13	−0.17
Doddle_RC+RF	**0.36**	**0.42**	0.04	0.01
Doddle_RP+RF	**0.36**	**0.41**	−0.02	−0.04
Doddle_×	**0.35**	**0.37**	−0.02	−0.02
Doddle_RC×RP	**0.38**	**0.41**	−0.11	−0.12
Doddle_RC×RF	**0.36**	**0.39**	0.03	0.03
Doddle_RP×RF	**0.35**	**0.39**	−0.01	−0.02
Doddle_W	**0.40**	**0.45**	−0.07	−0.11
SBR	*−0.21*	*−0.21*	—	—

Table D.82: UDC-236 average Spearman rank correlations, for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		SBR	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.43**	**0.24**	−0.16	−0.07
CORI	**0.48**	**0.47**	−0.30	−0.20
Cosine Measure	**0.32**	**0.51**	−0.17	−0.22
Inner Product	**0.46**	**0.51**	−0.21	−0.23
Skew	**0.42**	**0.45**	−0.15	−0.17
Highest-available Similarity	**0.47**	**0.53**	−0.23	−0.28
CVV	**0.36**	**0.36**	−0.14	−0.15
DFPROP	**0.40**	**0.44**	−0.13	−0.17
Distribution of Informative Amt	−0.30	−0.26	**0.19**	**0.11**
SCS	−0.33	−0.16	0.10	0.08
AvICTF	−0.33	−0.03	0.10	0.01
NSCQ	0.04	**0.31**	**0.16**	0.05
Doddle	**0.39**	**0.43**	−0.04	−0.07
SBR	*−0.21*	*−0.21*	—	—

Table D.83: UDC-236 Z-Test results, showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Short queries.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AvICTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
Algorithms																									
bGlOSS										✓	✓	✓				✓									✓
CORI			✓							✓	✓	✓				✓									✓
Cosine Measure										✓	✓	✓				✓									✓
Inner Product										✓	✓	✓				✓									✓
Skew										✓	✓	✓				✓									✓
Highest-available Similarity										✓	✓	✓				✓									✓
CVV										✓	✓	✓				✓									✓
DFPROP										✓	✓	✓				✓									✓
Distribution of Informative Amt																									
SCS																									
AviCTF																									
NSCQ	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓				✓									✓
Doddle										✓	✓	✓				✓									✓
Doddle_RC										✓	✓	✓				✓									✓
Doddle_RP										✓	✓	✓				✓									✓
Doddle_RF	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓				✓									✓
Doddle_RC+RP										✓	✓	✓				✓									✓
Doddle_RC+RF										✓	✓	✓				✓									✓
Doddle_RP+RF										✓	✓	✓				✓									✓
Doddle_×										✓	✓	✓				✓									✓
Doddle_RC×RP										✓	✓	✓				✓									✓
Doddle_RC×RF										✓	✓	✓				✓									✓
Doddle_RP×RF										✓	✓	✓				✓									✓
Doddle_W										✓	✓	✓				✓									✓
SBR	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓				✓									✓

Table D.84: UDC-236 Z-Test results, showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Long queries.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AvICTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
bGlOSS		✓	✓	✓	✓	✓		✓			✓		✓	✓	✓		✓	✓	✓		✓			✓	✓
CORI												✓				✓				✓		✓	✓		
Cosine Measure							✓	✓				✓				✓				✓		✓	✓		
Inner Product							✓	✓				✓				✓				✓		✓	✓		
Skew												✓				✓				✓		✓	✓		
Highest-available Similarity							✓					✓				✓				✓		✓	✓		
CVV			✓	✓		✓																			
DFPROP	✓																								
Distribution of Informative Amt																									
SCS																									
AvICTF	✓	✓	✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NSCQ		✓	✓	✓		✓								✓		✓									
Doddle												✓													
Doddle_RC																									
Doddle_RP																									
Doddle_RF		✓	✓	✓		✓								✓		✓									
Doddle_RC+RP																									
Doddle_RC+RF																									
Doddle_RP+RF																									
Doddle_×						✓																			
Doddle_RC×RP																									
Doddle_RC×RF																									
Doddle_RP×RF						✓																			
Doddle_W																									
SBR	✓	✓	✓	✓	✓	✓	✓	✓			✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

Table D.85: UDC-236 average Blest and Da Costa weighted rank correlations, for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR.

	Blest		**Da Costa**	
Algorithms	Q_s	Q_l	Q_s	Q_l
Doddle	**0.45**	**0.50**	**0.43**	**0.47**
Doddle_RC	**0.46**	**0.52**	**0.46**	**0.52**
Doddle_RP	**0.48**	**0.52**	**0.45**	**0.49**
Doddle_RF	**0.27**	**0.37**	**0.25**	**0.33**
Doddle_RC+RP	**0.48**	**0.52**	**0.46**	**0.51**
Doddle_RC+RF	**0.42**	**0.48**	**0.40**	**0.45**
Doddle_RP+RF	**0.44**	**0.48**	**0.40**	**0.44**
Doddle_×	**0.41**	**0.42**	**0.40**	**0.41**
Doddle_RC×RP	**0.43**	**0.46**	**0.43**	**0.45**
Doddle_RC×RF	**0.41**	**0.45**	**0.40**	**0.43**
Doddle_RP×RF	**0.43**	**0.46**	**0.39**	**0.42**
Doddle_W	**0.47**	**0.51**	**0.44**	**0.49**
SBR	*−0.17*	*−0.17*	*−0.12*	*−0.12*

Table D.86: UDC-236 average Blest and Da Costa weighted rank correlations, for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR.

	Blest		**Da Costa**	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.54**	**0.47**	**0.50**	**0.52**
CORI	**0.52**	**0.50**	**0.52**	**0.51**
Cosine Measure	**0.34**	**0.55**	**0.36**	**0.55**
Inner Product	**0.52**	**0.57**	**0.50**	**0.55**
Skew	**0.49**	**0.52**	**0.46**	**0.49**
Highest-available Similarity	**0.46**	**0.53**	**0.50**	**0.57**
CVV	**0.44**	**0.45**	**0.39**	**0.40**
DFPROP	**0.47**	**0.51**	**0.44**	**0.48**
Distribution of Informative Amt	−0.33	−0.28	−0.22	−0.19
SCS	−0.36	−0.17	−0.28	−0.11
AvICTF	−0.35	−0.03	−0.27	0.01
NSCQ	0.04	**0.33**	0.08	**0.34**
Doddle	**0.45**	**0.50**	**0.43**	**0.47**
SBR	*−0.17*	*−0.17*	*−0.12*	*−0.12*

Table D.87: UDC-236 average $\mathcal{R}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.24	0.37	0.29	0.08	0.35	0.22	0.15	0.23	0.31	0.21	0.22	0.30	*0.08*
10	0.34	0.42	0.36	0.13	0.41	0.31	0.26	0.33	0.37	0.31	0.26	0.37	*0.07*
20	0.42	0.48	0.43	0.20	0.47	0.39	0.34	0.39	0.44	0.39	0.35	0.44	*0.11*
30	0.48	0.53	0.48	0.29	0.52	0.46	0.44	0.46	0.49	0.46	0.43	0.50	*0.18*
40	0.52	0.56	0.53	0.37	0.55	0.51	0.49	0.50	0.52	0.51	0.49	0.54	*0.22*
50	0.56	0.58	0.57	0.43	0.58	0.54	0.54	0.54	0.55	0.54	0.54	0.57	*0.26*
60	0.60	0.61	0.60	0.48	0.61	0.58	0.58	0.58	0.59	0.58	0.58	0.61	*0.29*
70	0.63	0.64	0.64	0.52	0.65	0.61	0.62	0.61	0.62	0.61	0.62	0.64	*0.33*
80	0.66	0.68	0.67	0.56	0.68	0.65	0.65	0.65	0.65	0.65	0.65	0.67	*0.36*
90	0.69	0.70	0.70	0.59	0.71	0.68	0.69	0.68	0.69	0.68	0.68	0.70	*0.37*
100	0.72	0.73	0.73	0.63	0.74	0.71	0.72	0.70	0.72	0.71	0.71	0.73	*0.37*
110	0.75	0.76	0.76	0.67	0.76	0.74	0.75	0.73	0.75	0.74	0.74	0.76	*0.39*
120	0.78	0.79	0.79	0.70	0.79	0.77	0.77	0.77	0.78	0.77	0.77	0.78	*0.41*
130	0.80	0.81	0.81	0.74	0.82	0.79	0.80	0.79	0.80	0.79	0.80	0.81	*0.44*
140	0.83	0.84	0.84	0.77	0.84	0.82	0.83	0.82	0.83	0.82	0.83	0.83	*0.47*
150	0.86	0.86	0.87	0.80	0.87	0.85	0.86	0.85	0.85	0.85	0.85	0.86	*0.50*
160	0.88	0.89	0.89	0.83	0.89	0.88	0.88	0.87	0.88	0.87	0.88	0.89	*0.58*
170	0.91	0.91	0.91	0.86	0.92	0.90	0.90	0.89	0.91	0.89	0.90	0.91	*0.66*
180	0.93	0.93	0.93	0.89	0.94	0.92	0.93	0.92	0.93	0.92	0.92	0.93	*0.71*
190	0.95	0.95	0.95	0.92	0.95	0.94	0.95	0.94	0.94	0.94	0.94	0.95	*0.76*
200	0.96	0.97	0.96	0.94	0.97	0.96	0.96	0.95	0.96	0.96	0.96	0.96	*0.80*
210	0.97	0.98	0.97	0.96	0.98	0.97	0.97	0.97	0.97	0.97	0.97	0.98	*0.85*
220	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.99	*0.90*
230	1.00	0.99	0.99	0.99	1.00	1.00	0.99	0.99	0.99	0.99	0.99	1.00	*0.96*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.88: UDC-236 average $\mathcal{R}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.36	0.35	0.22	0.30	0.27	0.35	0.26	0.26	0.02	0.02	0.02	0.10	0.24	*0.08*
10	0.38	0.42	0.31	0.38	0.36	0.42	0.33	0.34	0.04	0.03	0.03	0.15	0.34	*0.07*
20	0.45	0.48	0.36	0.45	0.43	0.48	0.38	0.42	0.06	0.05	0.05	0.20	0.42	*0.11*
30	0.51	0.52	0.40	0.51	0.49	0.52	0.43	0.48	0.07	0.06	0.06	0.25	0.48	*0.18*
40	0.55	0.56	0.43	0.55	0.53	0.56	0.48	0.52	0.09	0.07	0.07	0.28	0.52	*0.22*
50	0.58	0.60	0.47	0.58	0.57	0.59	0.52	0.56	0.11	0.09	0.09	0.31	0.56	*0.26*
60	0.62	0.62	0.51	0.62	0.61	0.62	0.56	0.60	0.13	0.11	0.11	0.34	0.60	*0.29*
70	0.65	0.66	0.55	0.66	0.64	0.65	0.60	0.63	0.16	0.14	0.14	0.37	0.63	*0.33*
80	0.69	0.70	0.58	0.69	0.67	0.68	0.63	0.67	0.19	0.16	0.16	0.40	0.66	*0.36*
90	0.72	0.73	0.62	0.72	0.71	0.71	0.67	0.70	0.22	0.20	0.20	0.44	0.69	*0.37*
100	0.75	0.76	0.65	0.75	0.73	0.74	0.70	0.73	0.26	0.23	0.23	0.47	0.72	*0.37*
110	0.77	0.79	0.68	0.78	0.76	0.77	0.73	0.76	0.30	0.27	0.27	0.51	0.75	*0.39*
120	0.80	0.82	0.71	0.81	0.79	0.80	0.76	0.79	0.34	0.32	0.32	0.54	0.78	*0.41*
130	0.83	0.84	0.74	0.85	0.82	0.83	0.79	0.81	0.38	0.36	0.36	0.58	0.80	*0.44*
140	0.86	0.87	0.77	0.87	0.85	0.86	0.82	0.84	0.42	0.40	0.40	0.62	0.83	*0.47*
150	0.88	0.89	0.80	0.89	0.88	0.88	0.85	0.87	0.46	0.45	0.45	0.66	0.86	*0.50*
160	0.91	0.91	0.84	0.91	0.90	0.90	0.88	0.89	0.51	0.50	0.50	0.70	0.88	*0.58*
170	0.93	0.93	0.87	0.93	0.92	0.92	0.90	0.91	0.56	0.55	0.56	0.74	0.91	*0.66*
180	0.94	0.95	0.90	0.95	0.94	0.94	0.93	0.93	0.61	0.61	0.61	0.78	0.93	*0.71*
190	0.96	0.96	0.92	0.96	0.96	0.95	0.94	0.95	0.67	0.66	0.67	0.82	0.95	*0.76*
200	0.97	0.97	0.95	0.97	0.97	0.97	0.96	0.96	0.74	0.73	0.73	0.86	0.96	*0.80*
210	0.98	0.98	0.97	0.98	0.98	0.97	0.97	0.97	0.81	0.80	0.81	0.91	0.97	*0.85*
220	0.98	0.99	0.98	0.99	0.99	0.98	0.98	0.99	0.89	0.88	0.88	0.95	0.99	*0.90*
230	0.99	0.99	1.00	1.00	1.00	0.99	0.99	0.99	0.95	0.95	0.96	0.98	1.00	*0.96*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.89: UDC-236 average $\mathcal{R}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.24	0.32	0.27	0.09	0.31	0.18	0.17	0.19	0.26	0.17	0.17	0.28	*0.08*
10	0.36	0.44	0.40	0.15	0.42	0.32	0.29	0.31	0.37	0.29	0.27	0.40	*0.07*
20	0.43	0.50	0.48	0.21	0.50	0.41	0.37	0.39	0.44	0.38	0.35	0.47	*0.11*
30	0.50	0.55	0.53	0.31	0.54	0.48	0.45	0.45	0.50	0.45	0.43	0.53	*0.18*
40	0.55	0.58	0.56	0.39	0.58	0.53	0.51	0.49	0.55	0.51	0.49	0.57	*0.22*
50	0.59	0.62	0.60	0.46	0.61	0.57	0.56	0.54	0.58	0.55	0.54	0.60	*0.26*
60	0.62	0.64	0.63	0.51	0.64	0.61	0.61	0.59	0.61	0.59	0.59	0.63	*0.29*
70	0.65	0.68	0.66	0.56	0.67	0.64	0.64	0.62	0.64	0.63	0.63	0.66	*0.33*
80	0.69	0.71	0.69	0.61	0.70	0.67	0.68	0.65	0.67	0.66	0.66	0.70	*0.36*
90	0.72	0.74	0.72	0.65	0.73	0.71	0.71	0.68	0.70	0.70	0.70	0.73	*0.37*
100	0.75	0.77	0.75	0.69	0.76	0.74	0.74	0.72	0.74	0.73	0.74	0.76	*0.37*
110	0.78	0.80	0.78	0.72	0.79	0.77	0.77	0.75	0.77	0.76	0.77	0.79	*0.39*
120	0.81	0.83	0.81	0.76	0.82	0.80	0.80	0.78	0.79	0.79	0.80	0.81	*0.41*
130	0.83	0.85	0.83	0.79	0.84	0.83	0.83	0.81	0.82	0.82	0.82	0.84	*0.44*
140	0.86	0.87	0.86	0.82	0.87	0.86	0.85	0.84	0.85	0.85	0.85	0.86	*0.47*
150	0.88	0.90	0.88	0.86	0.89	0.88	0.88	0.86	0.87	0.87	0.87	0.89	*0.50*
160	0.91	0.92	0.90	0.88	0.92	0.90	0.90	0.88	0.90	0.89	0.90	0.91	*0.58*
170	0.93	0.94	0.92	0.91	0.93	0.92	0.92	0.91	0.92	0.92	0.92	0.93	*0.66*
180	0.95	0.96	0.94	0.93	0.95	0.95	0.94	0.93	0.94	0.94	0.94	0.95	*0.71*
190	0.96	0.97	0.96	0.95	0.97	0.96	0.96	0.95	0.95	0.96	0.96	0.96	*0.76*
200	0.97	0.98	0.97	0.97	0.98	0.97	0.97	0.96	0.96	0.97	0.97	0.97	*0.80*
210	0.98	0.99	0.98	0.98	0.98	0.98	0.98	0.97	0.98	0.98	0.98	0.98	*0.85*
220	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	*0.90*
230	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	*0.96*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.90: UDC-236 average $\mathcal{R}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.27	0.35	0.38	0.41	0.26	0.39	0.30	0.27	0.04	0.03	0.06	0.15	0.24	*0.08*
10	0.33	0.43	0.44	0.45	0.38	0.46	0.32	0.39	0.07	0.06	0.09	0.24	0.36	*0.07*
20	0.37	0.47	0.49	0.52	0.46	0.52	0.39	0.47	0.09	0.08	0.13	0.31	0.43	*0.11*
30	0.40	0.52	0.54	0.56	0.52	0.56	0.45	0.52	0.11	0.11	0.17	0.36	0.50	*0.18*
40	0.45	0.55	0.58	0.60	0.57	0.60	0.49	0.56	0.13	0.14	0.20	0.40	0.55	*0.22*
50	0.48	0.59	0.62	0.63	0.60	0.64	0.52	0.59	0.15	0.17	0.24	0.45	0.59	*0.26*
60	0.51	0.62	0.65	0.66	0.63	0.67	0.55	0.63	0.17	0.20	0.27	0.50	0.62	*0.29*
70	0.55	0.66	0.69	0.69	0.66	0.70	0.59	0.66	0.20	0.23	0.30	0.53	0.65	*0.33*
80	0.58	0.69	0.72	0.72	0.69	0.73	0.63	0.69	0.22	0.26	0.34	0.57	0.69	*0.36*
90	0.62	0.72	0.76	0.75	0.72	0.76	0.67	0.72	0.26	0.30	0.37	0.61	0.72	*0.37*
100	0.65	0.75	0.79	0.78	0.76	0.79	0.70	0.75	0.29	0.34	0.41	0.64	0.75	*0.37*
110	0.68	0.77	0.81	0.80	0.78	0.81	0.73	0.78	0.33	0.38	0.45	0.68	0.78	*0.39*
120	0.72	0.80	0.83	0.83	0.81	0.84	0.77	0.81	0.37	0.42	0.50	0.72	0.81	*0.41*
130	0.76	0.83	0.86	0.86	0.84	0.87	0.80	0.83	0.40	0.46	0.54	0.75	0.83	*0.44*
140	0.78	0.86	0.88	0.88	0.86	0.89	0.82	0.86	0.44	0.50	0.58	0.79	0.86	*0.47*
150	0.81	0.88	0.90	0.91	0.89	0.91	0.85	0.88	0.49	0.55	0.62	0.82	0.88	*0.50*
160	0.85	0.91	0.92	0.93	0.91	0.92	0.88	0.90	0.53	0.59	0.66	0.85	0.91	*0.58*
170	0.87	0.93	0.94	0.94	0.93	0.94	0.91	0.92	0.58	0.64	0.71	0.88	0.93	*0.66*
180	0.89	0.95	0.95	0.96	0.95	0.96	0.93	0.94	0.63	0.69	0.74	0.90	0.95	*0.71*
190	0.91	0.96	0.97	0.97	0.96	0.97	0.95	0.96	0.68	0.74	0.79	0.92	0.96	*0.76*
200	0.93	0.97	0.98	0.98	0.97	0.98	0.96	0.97	0.74	0.79	0.83	0.94	0.97	*0.80*
210	0.94	0.98	0.98	0.99	0.98	0.98	0.97	0.98	0.81	0.85	0.88	0.96	0.98	*0.85*
220	0.96	0.99	0.99	0.99	0.99	0.99	0.98	0.99	0.88	0.91	0.94	0.98	0.99	*0.90*
230	0.98	1.00	1.00	1.00	1.00	1.00	0.99	1.00	0.95	0.96	0.98	0.99	1.00	*0.96*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.91: UDC-236 average $\hat{\mathcal{R}}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.01	0.02	0.01	0.00	0.02	0.01	0.01	0.01	0.02	0.01	0.01	0.02	*0.00*
10	0.10	0.13	0.10	0.04	0.12	0.09	0.07	0.09	0.11	0.09	0.08	0.11	*0.02*
20	0.19	0.22	0.20	0.09	0.22	0.18	0.15	0.18	0.20	0.18	0.16	0.21	*0.05*
30	0.28	0.31	0.28	0.17	0.31	0.27	0.26	0.27	0.29	0.27	0.25	0.29	*0.10*
40	0.36	0.38	0.36	0.26	0.38	0.35	0.34	0.35	0.36	0.35	0.34	0.37	*0.15*
50	0.43	0.44	0.43	0.33	0.44	0.41	0.41	0.41	0.42	0.41	0.41	0.43	*0.19*
60	0.49	0.50	0.49	0.39	0.50	0.47	0.48	0.47	0.48	0.47	0.47	0.49	*0.23*
70	0.54	0.55	0.55	0.45	0.55	0.53	0.53	0.53	0.53	0.53	0.53	0.55	*0.28*
80	0.59	0.61	0.60	0.50	0.60	0.58	0.58	0.58	0.58	0.58	0.58	0.60	*0.32*
90	0.64	0.65	0.65	0.55	0.65	0.63	0.63	0.62	0.63	0.63	0.63	0.65	*0.33*
100	0.68	0.69	0.69	0.60	0.69	0.67	0.68	0.66	0.67	0.67	0.67	0.69	*0.35*
110	0.72	0.73	0.73	0.65	0.73	0.71	0.72	0.70	0.72	0.71	0.71	0.73	*0.37*
120	0.76	0.77	0.77	0.69	0.77	0.75	0.75	0.75	0.76	0.75	0.75	0.76	*0.40*
130	0.79	0.80	0.80	0.72	0.80	0.78	0.79	0.78	0.79	0.78	0.78	0.79	*0.43*
140	0.82	0.83	0.84	0.76	0.83	0.81	0.82	0.81	0.82	0.81	0.82	0.83	*0.47*
150	0.85	0.86	0.87	0.80	0.86	0.85	0.85	0.84	0.85	0.84	0.85	0.86	*0.50*
160	0.88	0.89	0.89	0.83	0.89	0.87	0.88	0.87	0.88	0.87	0.88	0.88	*0.58*
170	0.91	0.91	0.91	0.86	0.92	0.90	0.90	0.89	0.91	0.89	0.90	0.91	*0.66*
180	0.93	0.93	0.93	0.89	0.94	0.92	0.93	0.92	0.93	0.92	0.92	0.93	*0.71*
190	0.95	0.95	0.95	0.92	0.95	0.94	0.95	0.94	0.94	0.94	0.94	0.95	*0.76*
200	0.96	0.97	0.96	0.94	0.97	0.96	0.96	0.95	0.96	0.96	0.96	0.96	*0.80*
210	0.97	0.98	0.97	0.96	0.98	0.97	0.97	0.97	0.97	0.97	0.97	0.98	*0.85*
220	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.99	*0.90*
230	1.00	0.99	0.99	0.99	1.00	1.00	0.99	0.99	0.99	0.99	0.99	1.00	*0.96*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.92: UDC-236 average $\hat{\mathcal{R}}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.02	0.02	0.01	0.02	0.01	0.02	0.01	0.01	0.00	0.00	0.00	0.00	0.01	*0.00*
10	0.11	0.12	0.09	0.11	0.10	0.12	0.10	0.10	0.01	0.01	0.01	0.04	0.10	*0.02*
20	0.21	0.22	0.17	0.21	0.20	0.22	0.18	0.19	0.03	0.02	0.02	0.09	0.19	*0.05*
30	0.29	0.31	0.23	0.30	0.28	0.31	0.25	0.28	0.04	0.03	0.03	0.15	0.28	*0.10*
40	0.37	0.38	0.29	0.37	0.36	0.38	0.33	0.35	0.06	0.05	0.05	0.19	0.36	*0.15*
50	0.44	0.45	0.35	0.44	0.43	0.45	0.39	0.43	0.08	0.06	0.06	0.24	0.43	*0.19*
60	0.50	0.51	0.41	0.51	0.49	0.51	0.45	0.49	0.11	0.09	0.09	0.27	0.49	*0.23*
70	0.56	0.57	0.47	0.56	0.55	0.56	0.51	0.54	0.13	0.11	0.12	0.31	0.54	*0.28*
80	0.61	0.62	0.52	0.61	0.60	0.61	0.56	0.60	0.16	0.14	0.15	0.36	0.59	*0.32*
90	0.66	0.67	0.57	0.66	0.65	0.65	0.61	0.64	0.20	0.18	0.18	0.40	0.64	*0.33*
100	0.70	0.71	0.61	0.71	0.69	0.70	0.66	0.69	0.24	0.22	0.22	0.44	0.68	*0.35*
110	0.74	0.76	0.65	0.75	0.73	0.74	0.70	0.73	0.28	0.26	0.26	0.49	0.72	*0.37*
120	0.78	0.80	0.69	0.79	0.77	0.78	0.74	0.77	0.33	0.31	0.31	0.53	0.76	*0.40*
130	0.82	0.83	0.73	0.83	0.81	0.82	0.77	0.80	0.37	0.35	0.35	0.57	0.79	*0.43*
140	0.85	0.86	0.76	0.86	0.84	0.85	0.81	0.83	0.41	0.40	0.40	0.61	0.82	*0.47*
150	0.88	0.89	0.80	0.89	0.87	0.88	0.84	0.86	0.46	0.45	0.45	0.65	0.85	*0.50*
160	0.90	0.91	0.84	0.91	0.90	0.90	0.87	0.89	0.51	0.50	0.50	0.69	0.88	*0.58*
170	0.93	0.93	0.87	0.93	0.92	0.92	0.90	0.91	0.56	0.55	0.56	0.74	0.91	*0.66*
180	0.94	0.95	0.90	0.95	0.94	0.94	0.93	0.93	0.61	0.61	0.61	0.78	0.93	*0.71*
190	0.96	0.96	0.92	0.96	0.96	0.95	0.94	0.95	0.67	0.66	0.67	0.82	0.95	*0.76*
200	0.97	0.97	0.95	0.97	0.97	0.97	0.96	0.96	0.74	0.73	0.73	0.86	0.96	*0.80*
210	0.98	0.98	0.97	0.98	0.98	0.97	0.97	0.97	0.81	0.80	0.81	0.91	0.97	*0.85*
220	0.98	0.99	0.98	0.99	0.99	0.98	0.98	0.99	0.89	0.88	0.88	0.95	0.99	*0.90*
230	0.99	0.99	1.00	1.00	1.00	0.99	0.99	0.99	0.95	0.95	0.96	0.98	1.00	*0.96*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.93: UDC-236 average $\hat{\mathcal{R}}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.01	0.02	0.01	0.00	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	*0.00*
10	0.11	0.13	0.12	0.04	0.12	0.10	0.09	0.09	0.11	0.08	0.08	0.12	*0.02*
20	0.20	0.23	0.22	0.10	0.23	0.19	0.17	0.18	0.20	0.18	0.17	0.22	*0.05*
30	0.29	0.32	0.31	0.19	0.32	0.28	0.27	0.26	0.29	0.27	0.25	0.31	*0.10*
40	0.38	0.40	0.38	0.26	0.40	0.36	0.35	0.34	0.37	0.35	0.34	0.39	*0.15*
50	0.45	0.47	0.45	0.35	0.46	0.44	0.43	0.41	0.44	0.42	0.41	0.46	*0.19*
60	0.50	0.52	0.51	0.42	0.52	0.50	0.49	0.48	0.49	0.48	0.48	0.51	*0.23*
70	0.56	0.58	0.56	0.48	0.58	0.55	0.55	0.53	0.55	0.54	0.54	0.57	*0.28*
80	0.62	0.63	0.62	0.54	0.63	0.60	0.61	0.58	0.60	0.59	0.60	0.62	*0.32*
90	0.66	0.68	0.67	0.60	0.68	0.65	0.66	0.63	0.65	0.64	0.65	0.67	*0.33*
100	0.71	0.73	0.71	0.65	0.72	0.70	0.70	0.68	0.70	0.69	0.69	0.72	*0.35*
110	0.75	0.77	0.75	0.70	0.76	0.74	0.74	0.72	0.74	0.73	0.74	0.76	*0.37*
120	0.79	0.80	0.79	0.74	0.80	0.78	0.78	0.76	0.77	0.77	0.78	0.79	*0.40*
130	0.82	0.84	0.82	0.78	0.83	0.82	0.81	0.80	0.81	0.81	0.81	0.82	*0.43*
140	0.85	0.87	0.85	0.82	0.86	0.85	0.84	0.83	0.84	0.84	0.84	0.85	*0.47*
150	0.88	0.90	0.88	0.85	0.89	0.88	0.87	0.86	0.87	0.87	0.87	0.88	*0.50*
160	0.90	0.92	0.90	0.88	0.91	0.90	0.90	0.88	0.89	0.89	0.89	0.91	*0.58*
170	0.93	0.94	0.92	0.91	0.93	0.92	0.92	0.91	0.92	0.92	0.92	0.93	*0.66*
180	0.95	0.96	0.94	0.93	0.95	0.95	0.94	0.93	0.94	0.94	0.94	0.95	*0.71*
190	0.96	0.97	0.96	0.95	0.97	0.96	0.96	0.95	0.95	0.96	0.96	0.96	*0.76*
200	0.97	0.98	0.97	0.97	0.98	0.97	0.97	0.96	0.96	0.97	0.97	0.97	*0.80*
210	0.98	0.99	0.98	0.98	0.98	0.98	0.98	0.97	0.98	0.98	0.98	0.98	*0.85*
220	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	*0.90*
230	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	*0.96*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.94: UDC-236 average $\hat{\mathcal{R}}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.02	0.02	0.02	0.02	0.01	0.02	0.02	0.01	0.00	0.00	0.00	0.01	0.01	*0.00*
10	0.10	0.13	0.13	0.13	0.11	0.14	0.09	0.12	0.02	0.02	0.03	0.07	0.11	*0.02*
20	0.17	0.22	0.23	0.24	0.21	0.24	0.18	0.22	0.04	0.04	0.06	0.14	0.20	*0.05*
30	0.23	0.30	0.31	0.33	0.30	0.33	0.26	0.30	0.06	0.07	0.10	0.21	0.29	*0.10*
40	0.30	0.38	0.40	0.41	0.38	0.41	0.33	0.38	0.09	0.10	0.13	0.27	0.38	*0.15*
50	0.36	0.44	0.47	0.48	0.45	0.48	0.39	0.45	0.11	0.13	0.18	0.34	0.45	*0.19*
60	0.41	0.50	0.53	0.54	0.51	0.54	0.45	0.51	0.14	0.16	0.22	0.40	0.50	*0.23*
70	0.47	0.56	0.59	0.59	0.57	0.60	0.51	0.56	0.17	0.20	0.26	0.46	0.56	*0.28*
80	0.52	0.62	0.65	0.64	0.62	0.65	0.56	0.62	0.20	0.23	0.30	0.51	0.62	*0.32*
90	0.57	0.67	0.70	0.69	0.67	0.70	0.62	0.66	0.24	0.27	0.34	0.56	0.66	*0.33*
100	0.61	0.71	0.74	0.73	0.71	0.74	0.66	0.71	0.28	0.32	0.39	0.61	0.71	*0.35*
110	0.66	0.74	0.78	0.77	0.75	0.78	0.71	0.75	0.31	0.36	0.43	0.65	0.75	*0.37*
120	0.70	0.78	0.81	0.81	0.79	0.82	0.75	0.78	0.36	0.41	0.49	0.70	0.79	*0.40*
130	0.74	0.82	0.84	0.84	0.82	0.85	0.78	0.82	0.39	0.46	0.53	0.74	0.82	*0.43*
140	0.78	0.85	0.87	0.87	0.86	0.88	0.81	0.85	0.44	0.50	0.58	0.78	0.85	*0.47*
150	0.81	0.88	0.90	0.90	0.88	0.90	0.85	0.88	0.48	0.54	0.62	0.81	0.88	*0.50*
160	0.84	0.91	0.92	0.92	0.91	0.92	0.88	0.90	0.53	0.59	0.66	0.85	0.90	*0.58*
170	0.87	0.93	0.94	0.94	0.93	0.94	0.91	0.92	0.58	0.64	0.71	0.88	0.93	*0.66*
180	0.89	0.95	0.95	0.96	0.95	0.96	0.93	0.94	0.63	0.69	0.74	0.90	0.95	*0.71*
190	0.91	0.96	0.97	0.97	0.96	0.97	0.95	0.96	0.68	0.74	0.79	0.92	0.96	*0.76*
200	0.93	0.97	0.98	0.98	0.97	0.98	0.96	0.97	0.74	0.79	0.83	0.94	0.97	*0.80*
210	0.94	0.98	0.98	0.99	0.98	0.98	0.97	0.98	0.81	0.85	0.88	0.96	0.98	*0.85*
220	0.96	0.99	0.99	0.99	0.99	0.99	0.98	0.99	0.88	0.91	0.94	0.98	0.99	*0.90*
230	0.98	1.00	1.00	1.00	1.00	1.00	0.99	1.00	0.95	0.96	0.98	0.99	1.00	*0.96*
236	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.95: UDC-236 average $\mathcal{P}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.53	0.78	0.61	0.32	0.72	0.50	0.43	0.55	0.69	0.55	0.47	0.61	*0.18*
10	0.60	0.72	0.64	0.33	0.70	0.57	0.48	0.58	0.65	0.58	0.48	0.65	*0.20*
20	0.63	0.71	0.67	0.37	0.70	0.59	0.52	0.59	0.66	0.59	0.53	0.66	*0.26*
30	0.65	0.71	0.68	0.45	0.71	0.62	0.59	0.62	0.66	0.62	0.58	0.67	*0.38*
40	0.66	0.70	0.68	0.50	0.70	0.63	0.62	0.63	0.66	0.63	0.61	0.68	*0.43*
50	0.66	0.69	0.67	0.52	0.69	0.63	0.63	0.63	0.65	0.63	0.62	0.67	*0.46*
60	0.65	0.68	0.66	0.54	0.67	0.63	0.63	0.63	0.64	0.63	0.62	0.66	*0.48*
70	0.64	0.66	0.66	0.55	0.66	0.63	0.63	0.62	0.63	0.63	0.62	0.65	*0.49*
80	0.64	0.65	0.65	0.55	0.65	0.62	0.62	0.62	0.63	0.62	0.62	0.64	*0.49*
90	0.63	0.64	0.64	0.55	0.64	0.62	0.62	0.61	0.62	0.62	0.61	0.63	*0.45*
100	0.62	0.63	0.63	0.56	0.63	0.61	0.61	0.60	0.62	0.61	0.61	0.63	*0.43*
110	0.61	0.63	0.62	0.56	0.63	0.60	0.61	0.60	0.61	0.60	0.60	0.62	*0.41*
120	0.60	0.62	0.61	0.55	0.61	0.60	0.60	0.59	0.60	0.60	0.59	0.61	*0.39*
130	0.59	0.61	0.60	0.55	0.61	0.59	0.59	0.58	0.59	0.59	0.59	0.60	*0.40*
140	0.58	0.60	0.59	0.55	0.60	0.58	0.58	0.58	0.58	0.58	0.58	0.59	*0.40*
150	0.58	0.59	0.58	0.54	0.58	0.57	0.57	0.57	0.58	0.57	0.57	0.58	*0.41*
160	0.57	0.58	0.57	0.54	0.57	0.56	0.56	0.56	0.57	0.56	0.56	0.57	*0.42*
170	0.56	0.57	0.56	0.53	0.57	0.55	0.55	0.55	0.56	0.55	0.55	0.56	*0.44*
180	0.55	0.55	0.55	0.52	0.55	0.54	0.54	0.54	0.54	0.54	0.54	0.55	*0.44*
190	0.53	0.54	0.54	0.52	0.54	0.53	0.53	0.53	0.53	0.53	0.53	0.54	*0.45*
200	0.52	0.52	0.52	0.51	0.52	0.52	0.52	0.51	0.52	0.52	0.52	0.52	*0.45*
210	0.51	0.51	0.51	0.50	0.51	0.50	0.50	0.50	0.50	0.50	0.50	0.51	*0.46*
220	0.49	0.49	0.49	0.49	0.49	0.49	0.49	0.49	0.49	0.49	0.49	0.49	*0.46*
230	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	*0.47*
236	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	*0.47*

Table D.96: UDC-236 average $\mathcal{P}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.71	0.78	0.61	0.67	0.62	0.78	0.61	0.57	0.08	0.09	0.09	0.38	0.53	*0.18*
10	0.69	0.76	0.61	0.69	0.65	0.75	0.62	0.62	0.18	0.12	0.12	0.40	0.60	*0.20*
20	0.71	0.76	0.61	0.71	0.68	0.75	0.63	0.66	0.19	0.15	0.15	0.43	0.63	*0.26*
30	0.72	0.75	0.61	0.71	0.68	0.74	0.63	0.67	0.21	0.17	0.17	0.46	0.65	*0.38*
40	0.71	0.74	0.60	0.71	0.68	0.74	0.63	0.67	0.23	0.18	0.18	0.46	0.66	*0.43*
50	0.71	0.72	0.60	0.70	0.68	0.72	0.63	0.67	0.24	0.20	0.20	0.47	0.66	*0.46*
60	0.69	0.71	0.60	0.69	0.67	0.71	0.63	0.66	0.26	0.22	0.22	0.47	0.65	*0.48*
70	0.68	0.70	0.59	0.68	0.66	0.70	0.63	0.65	0.28	0.25	0.25	0.47	0.64	*0.49*
80	0.67	0.69	0.59	0.67	0.65	0.68	0.62	0.64	0.30	0.28	0.28	0.48	0.64	*0.49*
90	0.66	0.68	0.59	0.66	0.64	0.67	0.62	0.64	0.32	0.30	0.30	0.49	0.63	*0.45*
100	0.65	0.67	0.59	0.65	0.63	0.66	0.61	0.62	0.34	0.32	0.32	0.49	0.62	*0.43*
110	0.64	0.66	0.58	0.64	0.62	0.65	0.60	0.62	0.36	0.34	0.34	0.49	0.61	*0.41*
120	0.63	0.65	0.58	0.63	0.62	0.64	0.60	0.61	0.37	0.36	0.36	0.50	0.60	*0.39*
130	0.62	0.63	0.57	0.63	0.61	0.63	0.59	0.60	0.38	0.37	0.37	0.50	0.59	*0.40*
140	0.61	0.62	0.57	0.62	0.60	0.62	0.58	0.59	0.40	0.39	0.39	0.50	0.58	*0.40*
150	0.60	0.61	0.56	0.60	0.59	0.61	0.57	0.58	0.41	0.40	0.40	0.50	0.58	*0.41*
160	0.59	0.60	0.56	0.59	0.58	0.60	0.57	0.57	0.42	0.41	0.41	0.50	0.57	*0.42*
170	0.57	0.58	0.55	0.58	0.57	0.58	0.56	0.56	0.43	0.42	0.42	0.50	0.56	*0.44*
180	0.56	0.57	0.54	0.56	0.55	0.57	0.55	0.55	0.43	0.43	0.43	0.50	0.55	*0.44*
190	0.54	0.55	0.53	0.55	0.54	0.55	0.53	0.53	0.44	0.44	0.44	0.49	0.53	*0.45*
200	0.53	0.53	0.52	0.53	0.52	0.53	0.52	0.52	0.45	0.45	0.45	0.49	0.52	*0.45*
210	0.51	0.51	0.51	0.51	0.51	0.51	0.50	0.50	0.46	0.46	0.46	0.49	0.51	*0.46*
220	0.49	0.50	0.49	0.49	0.49	0.49	0.49	0.49	0.47	0.47	0.47	0.48	0.49	*0.46*
230	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.47	0.47	0.47	0.48	0.48	*0.47*
236	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	*0.47*

Table D.97: UDC-236 average $\mathcal{P}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.61	0.78	0.64	0.40	0.74	0.56	0.49	0.60	0.65	0.58	0.48	0.71	*0.18*
10	0.63	0.74	0.68	0.36	0.71	0.58	0.53	0.58	0.65	0.57	0.52	0.68	*0.20*
20	0.64	0.74	0.69	0.40	0.72	0.61	0.55	0.60	0.66	0.60	0.54	0.69	*0.26*
30	0.67	0.73	0.71	0.49	0.73	0.64	0.61	0.62	0.67	0.62	0.59	0.70	*0.38*
40	0.68	0.73	0.70	0.54	0.73	0.66	0.63	0.62	0.68	0.64	0.61	0.70	*0.43*
50	0.68	0.72	0.69	0.57	0.71	0.66	0.65	0.63	0.67	0.64	0.63	0.69	*0.46*
60	0.67	0.71	0.68	0.59	0.70	0.66	0.65	0.63	0.66	0.64	0.63	0.68	*0.48*
70	0.66	0.70	0.67	0.60	0.69	0.66	0.65	0.62	0.65	0.64	0.63	0.68	*0.49*
80	0.66	0.69	0.67	0.60	0.68	0.65	0.65	0.62	0.64	0.64	0.63	0.67	*0.49*
90	0.65	0.67	0.65	0.60	0.67	0.64	0.64	0.62	0.63	0.63	0.63	0.66	*0.45*
100	0.64	0.66	0.64	0.60	0.66	0.64	0.63	0.61	0.63	0.63	0.62	0.65	*0.43*
110	0.63	0.65	0.63	0.60	0.65	0.63	0.62	0.61	0.62	0.62	0.62	0.64	*0.41*
120	0.62	0.64	0.63	0.59	0.64	0.62	0.62	0.60	0.61	0.61	0.61	0.63	*0.39*
130	0.61	0.63	0.62	0.58	0.62	0.61	0.61	0.59	0.60	0.60	0.60	0.62	*0.40*
140	0.60	0.62	0.61	0.58	0.61	0.60	0.60	0.58	0.59	0.59	0.59	0.61	*0.40*
150	0.59	0.61	0.59	0.57	0.60	0.59	0.59	0.57	0.58	0.58	0.58	0.60	*0.41*
160	0.58	0.59	0.58	0.56	0.59	0.58	0.58	0.56	0.57	0.57	0.57	0.58	*0.42*
170	0.57	0.58	0.57	0.55	0.58	0.57	0.57	0.55	0.56	0.56	0.56	0.57	*0.44*
180	0.56	0.57	0.56	0.55	0.56	0.56	0.55	0.54	0.55	0.55	0.55	0.56	*0.44*
190	0.54	0.55	0.54	0.53	0.55	0.54	0.54	0.53	0.54	0.54	0.54	0.55	*0.45*
200	0.53	0.53	0.52	0.52	0.53	0.53	0.53	0.52	0.52	0.53	0.52	0.53	*0.45*
210	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.50	0.50	0.51	0.51	0.51	*0.46*
220	0.49	0.50	0.49	0.49	0.49	0.50	0.49	0.49	0.49	0.49	0.49	0.49	*0.46*
230	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	*0.47*
236	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	*0.47*

Table D.98: UDC-236 average $\mathcal{P}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.64	0.85	0.83	0.81	0.62	0.85	0.61	0.60	0.21	0.15	0.26	0.46	0.61	*0.18*
10	0.66	0.79	0.77	0.76	0.67	0.82	0.59	0.68	0.24	0.24	0.31	0.55	0.63	*0.20*
20	0.64	0.76	0.76	0.76	0.68	0.80	0.60	0.68	0.25	0.26	0.35	0.58	0.64	*0.26*
30	0.63	0.76	0.75	0.76	0.70	0.79	0.63	0.70	0.27	0.29	0.39	0.60	0.67	*0.38*
40	0.64	0.74	0.75	0.75	0.71	0.78	0.63	0.70	0.28	0.31	0.40	0.61	0.68	*0.43*
50	0.63	0.73	0.74	0.74	0.70	0.76	0.62	0.69	0.30	0.33	0.42	0.61	0.68	*0.46*
60	0.62	0.71	0.73	0.72	0.68	0.75	0.62	0.68	0.31	0.35	0.43	0.62	0.67	*0.48*
70	0.62	0.70	0.72	0.71	0.68	0.73	0.62	0.67	0.33	0.37	0.44	0.62	0.66	*0.49*
80	0.61	0.69	0.70	0.69	0.67	0.72	0.62	0.66	0.34	0.38	0.45	0.61	0.66	*0.49*
90	0.61	0.68	0.69	0.68	0.66	0.71	0.61	0.65	0.35	0.40	0.46	0.61	0.65	*0.45*
100	0.60	0.67	0.68	0.67	0.65	0.69	0.61	0.64	0.37	0.41	0.46	0.60	0.64	*0.43*
110	0.60	0.65	0.67	0.66	0.64	0.68	0.60	0.63	0.38	0.42	0.47	0.60	0.63	*0.41*
120	0.60	0.64	0.65	0.65	0.63	0.66	0.60	0.62	0.39	0.42	0.47	0.60	0.62	*0.39*
130	0.59	0.63	0.64	0.64	0.62	0.65	0.59	0.61	0.39	0.43	0.47	0.59	0.61	*0.40*
140	0.58	0.62	0.63	0.63	0.61	0.64	0.58	0.60	0.40	0.44	0.48	0.58	0.60	*0.40*
150	0.57	0.61	0.62	0.62	0.60	0.62	0.57	0.59	0.41	0.44	0.48	0.58	0.59	*0.41*
160	0.56	0.59	0.60	0.60	0.59	0.61	0.56	0.58	0.41	0.45	0.48	0.57	0.58	*0.42*
170	0.56	0.58	0.59	0.59	0.57	0.59	0.55	0.57	0.42	0.45	0.48	0.56	0.57	*0.44*
180	0.54	0.57	0.57	0.57	0.56	0.58	0.55	0.56	0.43	0.46	0.48	0.55	0.56	*0.44*
190	0.53	0.55	0.55	0.55	0.55	0.56	0.53	0.54	0.44	0.46	0.48	0.54	0.54	*0.45*
200	0.52	0.53	0.54	0.53	0.53	0.54	0.52	0.52	0.44	0.47	0.48	0.52	0.53	*0.45*
210	0.50	0.51	0.52	0.52	0.51	0.52	0.50	0.51	0.45	0.47	0.48	0.51	0.51	*0.46*
220	0.48	0.50	0.50	0.50	0.49	0.50	0.49	0.49	0.46	0.47	0.48	0.49	0.49	*0.46*
230	0.47	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.47	0.47	0.47	0.48	0.48	*0.47*
236	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	*0.47*

Table D.99: UDC-236 average precision within the top 5 collections, and the number of queries (out of 100) for which the first ranked collection is correct, for the various configurations of the Doddle algorithm.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Q_s	Q_l	Q_s		Q_l	
Doddle	0.12	0.12	11	*(11%)*	5	*(5%)*
Doddle_RC	0.17	0.16	15	*(15%)*	8	*(8%)*
Doddle_RP	0.14	0.14	11	*(11%)*	8	*(8%)*
Doddle_RF	0.02	0.04	1	*(1%)*	0	*(0%)*
Doddle_RC+RP	0.17	0.15	15	*(15%)*	8	*(8%)*
Doddle_RC+RF	0.12	0.10	9	*(9%)*	2	*(2%)*
Doddle_RP+RF	0.09	0.09	4	*(4%)*	2	*(2%)*
Doddle_×	0.11	0.08	9	*(9%)*	2	*(2%)*
Doddle_RC×RP	0.15	0.12	12	*(12%)*	5	*(5%)*
Doddle_RC×RF	0.10	0.08	6	*(6%)*	2	*(2%)*
Doddle_RP×RF	0.09	0.08	10	*(10%)*	1	*(1%)*
Doddle_W	0.14	0.14	15	*(15%)*	6	*(6%)*
SBR	*0.02*	*0.02*	*3*	*(3%)*	*3*	*(3%)*

Table D.100: UDC-236 average precision within the top 5 collections, and the number of queries (out of 100) for which the first ranked collection is correct, for existing algorithms and Doddle.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Q_s	Q_l	Q_s		Q_l	
bGlOSS	0.15	0.09	15	*(15%)*	7	*(7%)*
CORI	0.16	0.15	10	*(10%)*	10	*(10%)*
Cosine Measure	0.09	0.17	5	*(5%)*	11	*(11%)*
Inner Product	0.16	0.17	11	*(11%)*	16	*(16%)*
Skew	0.14	0.13	10	*(10%)*	7	*(7%)*
Highest-available Similarity	0.16	0.17	11	*(11%)*	13	*(13%)*
CVV	0.14	0.12	10	*(10%)*	12	*(12%)*
DFPROP	0.13	0.14	10	*(10%)*	8	*(8%)*
Distribution of Informative Amt	0.00	0.01	0	*(0%)*	0	*(0%)*
SCS	0.00	0.00	0	*(0%)*	0	*(0%)*
AvICTF	0.00	0.01	0	*(0%)*	0	*(0%)*
NSCQ	0.02	0.06	2	*(2%)*	1	*(1%)*
Doddle	0.12	0.12	11	*(11%)*	5	*(5%)*
SBR	*0.02*	*0.02*	*3*	*(3%)*	*3*	*(3%)*

D.5 UBC-100

Table D.101: UBC-100 average Spearman rank correlations, for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		SBR	
Algorithms	Q_s	Q_l	Q_s	Q_l
Doddle	**0.42**	**0.47**	−0.15	−0.16
Doddle_RC	**0.52**	**0.59**	0.16	0.15
Doddle_RP	**0.45**	**0.51**	−0.17	−0.18
Doddle_RF	0.14	**0.22**	−0.37	−0.37
Doddle_RC+RP	**0.50**	**0.56**	−0.02	−0.04
Doddle_RC+RF	**0.38**	**0.45**	−0.12	−0.14
Doddle_RP+RF	**0.34**	**0.40**	−0.30	−0.30
Doddle_×	**0.38**	**0.38**	−0.17	−0.23
Doddle_RC×RP	**0.46**	**0.49**	−0.04	−0.08
Doddle_RC×RF	**0.39**	**0.41**	−0.11	−0.17
Doddle_RP×RF	**0.32**	**0.35**	−0.33	−0.35
Doddle_W	**0.45**	**0.51**	−0.10	−0.11
SBR	***0.18***	***0.18***	—	—

Table D.102: UBC-100 average Spearman rank correlations, for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		SBR	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.56**	**0.35**	0.12	0.06
CORI	**0.62**	**0.61**	**0.39**	**0.32**
Cosine Measure	**0.40**	**0.63**	**0.18**	**0.25**
Inner Product	**0.61**	**0.65**	**0.44**	**0.41**
Skew	**0.56**	**0.59**	**0.47**	**0.46**
Highest-available Similarity	**0.47**	**0.52**	**0.65**	**0.66**
CVV	**0.50**	**0.54**	**0.48**	**0.52**
DFPROP	**0.55**	**0.59**	**0.50**	**0.51**
Distribution of Informative Amt	−0.45	−0.43	−0.28	−0.19
SCS	−0.40	−0.25	−0.09	0.04
AvICTF	−0.39	−0.09	−0.08	0.10
NSCQ	0.09	**0.35**	**0.55**	**0.61**
Doddle	**0.42**	**0.47**	−0.15	−0.16
SBR	***0.18***	*0.18*	—	—

Table D.103: UBC-100 Z-Test results, showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Short queries.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AvICTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
bGlOSS																									
CORI			✓																						
Cosine Measure				✓																					
Inner Product																									
Skew																									
Highest-available Similarity																									
CVV																									
DFPROP																									
Distribution of Informative Amt																									
SCS																									
AvICTF																									
NSCQ																									
Doddle																									
Doddle_RC																									
Doddle_RP																									
Doddle_RF																									
Doddle_RC+RP																									
Doddle_RC+RF																									
Doddle_RP+RF																									
Doddle_×																									
Doddle_RC×RP																									
Doddle_RC×RF																									
Doddle_RP×RF																									
Doddle_W																									
SBR.																									

Table D.104: UBC-100 Z-Test results, showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Long queries.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AvICTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
bGlOSS		✓	✓	✓	✓			✓			✓			✓											
CORI											✓	✓								✓		✓	✓		✓
Cosine Measure											✓	✓								✓		✓	✓		✓
Inner Product											✓	✓								✓		✓	✓		✓
Skew											✓	✓								✓			✓		✓
Highest-available Similarity											✓														
CVV											✓														
DFPROP											✓	✓		✓									✓		✓
Distribution of Informative Amt											✓														
SCS	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
AvICTF	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NSCQ		✓	✓	✓	✓			✓			✓			✓											
Doddle											✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Doddle_RC											✓	✓													
Doddle_RP											✓	✓													
Doddle_RF		✓	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓		✓				✓			✓	
Doddle_RC+RP											✓	✓													
Doddle_RC+RF				✓							✓	✓													
Doddle_RP+RF		✓	✓	✓							✓	✓													
Doddle_×		✓	✓	✓							✓	✓													
Doddle_RC×RP											✓	✓													
Doddle_RC×RF			✓	✓							✓	✓													
Doddle_RP×RF		✓	✓	✓	✓			✓			✓	✓		✓											
Doddle_W											✓	✓													
SBR		✓	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓		✓				✓			✓	

Table D.105: UBC-100 average Blest and Da Costa weighted rank correlations, for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR.

	Blest		**Da Costa**	
Algorithms	Q_s	Q_l	Q_s	Q_l
Doddle	**0.39**	**0.43**	**0.44**	**0.49**
Doddle_RC	**0.50**	**0.56**	**0.54**	**0.61**
Doddle_RP	**0.42**	**0.46**	**0.48**	**0.53**
Doddle_RF	0.13	**0.20**	0.14	**0.22**
Doddle_RC+RP	**0.47**	**0.52**	**0.52**	**0.58**
Doddle_RC+RF	**0.36**	**0.41**	**0.40**	**0.46**
Doddle_RP+RF	**0.31**	**0.36**	**0.35**	**0.41**
Doddle_×	**0.36**	**0.35**	**0.40**	**0.40**
Doddle_RC×RP	**0.44**	**0.45**	**0.48**	**0.51**
Doddle_RC×RF	**0.37**	**0.37**	**0.40**	**0.42**
Doddle_RP×RF	**0.29**	**0.31**	**0.33**	**0.37**
Doddle_W	**0.43**	**0.47**	**0.48**	**0.53**
SBR	***0.17***	***0.17***	*0.16*	*0.16*

Table D.106: UBC-100 average Blest and Da Costa weighted rank correlations, for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR.

	Blest		**Da Costa**	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.53**	**0.25**	**0.60**	**0.55**
CORI	**0.59**	**0.59**	**0.63**	**0.63**
Cosine Measure	**0.38**	**0.60**	**0.41**	**0.64**
Inner Product	**0.58**	**0.63**	**0.62**	**0.67**
Skew	**0.54**	**0.56**	**0.58**	**0.61**
Highest-available Similarity	**0.44**	**0.49**	**0.46**	**0.52**
CVV	**0.48**	**0.52**	**0.52**	**0.56**
DFPROP	**0.53**	**0.56**	**0.57**	**0.60**
Distribution of Informative Amt	−0.42	−0.41	−0.38	−0.38
SCS	−0.37	−0.23	−0.37	−0.24
AvICTF	−0.37	−0.08	−0.37	−0.09
NSCQ	0.09	**0.33**	0.08	**0.33**
Doddle	**0.39**	**0.43**	**0.44**	**0.49**
SBR	***0.17***	***0.17***	*0.16*	*0.16*

Table D.107: UBC-100 average $\mathcal{R}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.21	0.34	0.31	0.06	0.33	0.18	0.15	0.25	0.34	0.20	0.18	0.33	*0.01*
10	0.41	0.51	0.45	0.16	0.50	0.38	0.32	0.40	0.49	0.36	0.30	0.46	*0.09*
20	0.54	0.62	0.57	0.25	0.61	0.51	0.44	0.52	0.59	0.50	0.42	0.57	*0.23*
30	0.65	0.70	0.68	0.42	0.71	0.63	0.59	0.64	0.68	0.63	0.59	0.67	*0.35*
40	0.74	0.78	0.77	0.56	0.79	0.72	0.70	0.73	0.76	0.72	0.70	0.76	*0.53*
50	0.82	0.84	0.84	0.67	0.85	0.80	0.80	0.80	0.83	0.80	0.79	0.83	*0.62*
60	0.88	0.89	0.89	0.78	0.90	0.87	0.87	0.86	0.88	0.87	0.87	0.89	*0.71*
70	0.92	0.93	0.93	0.86	0.94	0.91	0.92	0.91	0.93	0.92	0.92	0.93	*0.82*
80	0.96	0.96	0.96	0.93	0.96	0.95	0.96	0.95	0.96	0.95	0.95	0.96	*0.93*
90	0.99	0.98	0.99	0.98	0.98	0.98	0.99	0.98	0.98	0.98	0.99	0.99	*0.96*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.108: UBC-100 average $\mathcal{R}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.41	0.36	0.21	0.41	0.35	0.09	0.31	0.35	0.03	0.02	0.02	0.06	0.21	*0.01*
10	0.53	0.52	0.35	0.50	0.50	0.27	0.44	0.49	0.05	0.05	0.05	0.15	0.41	*0.09*
20	0.63	0.65	0.46	0.64	0.63	0.48	0.57	0.63	0.09	0.07	0.07	0.23	0.54	*0.23*
30	0.71	0.75	0.57	0.74	0.72	0.63	0.67	0.72	0.12	0.11	0.11	0.31	0.65	*0.35*
40	0.79	0.83	0.66	0.82	0.79	0.73	0.75	0.79	0.16	0.15	0.15	0.40	0.74	*0.53*
50	0.84	0.88	0.74	0.87	0.85	0.81	0.81	0.84	0.23	0.23	0.23	0.51	0.82	*0.62*
60	0.89	0.91	0.82	0.91	0.89	0.88	0.87	0.88	0.32	0.34	0.35	0.61	0.88	*0.71*
70	0.93	0.94	0.89	0.94	0.92	0.91	0.91	0.92	0.45	0.49	0.50	0.72	0.92	*0.82*
80	0.95	0.95	0.94	0.96	0.95	0.94	0.95	0.95	0.61	0.65	0.66	0.85	0.96	*0.93*
90	0.98	0.97	0.98	0.98	0.97	0.97	0.97	0.97	0.80	0.84	0.84	0.95	0.99	*0.96*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.109: UBC-100 average $\mathcal{R}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.27	0.34	0.28	0.12	0.35	0.21	0.20	0.18	0.27	0.19	0.17	0.33	*0.01*
10	0.46	0.54	0.50	0.19	0.55	0.43	0.38	0.40	0.49	0.38	0.35	0.51	*0.09*
20	0.58	0.64	0.61	0.26	0.64	0.55	0.48	0.51	0.60	0.51	0.44	0.61	*0.23*
30	0.67	0.74	0.71	0.46	0.73	0.65	0.63	0.62	0.69	0.62	0.60	0.71	*0.35*
40	0.75	0.82	0.79	0.61	0.81	0.74	0.73	0.71	0.77	0.72	0.70	0.78	*0.53*
50	0.83	0.88	0.85	0.74	0.87	0.83	0.81	0.80	0.84	0.81	0.80	0.85	*0.62*
60	0.90	0.92	0.91	0.83	0.92	0.89	0.89	0.87	0.90	0.89	0.88	0.91	*0.71*
70	0.95	0.96	0.95	0.91	0.96	0.94	0.94	0.93	0.94	0.94	0.94	0.95	*0.82*
80	0.97	0.98	0.97	0.96	0.97	0.97	0.97	0.96	0.97	0.97	0.97	0.97	*0.93*
90	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	*0.96*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.110: UBC-100 average $\mathcal{R}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.29	0.35	0.37	0.42	0.32	0.16	0.30	0.36	0.04	0.03	0.04	0.11	0.27	*0.01*
10	0.37	0.53	0.53	0.55	0.50	0.34	0.46	0.52	0.06	0.06	0.09	0.23	0.46	*0.09*
20	0.44	0.63	0.64	0.67	0.63	0.53	0.59	0.64	0.10	0.10	0.15	0.36	0.58	*0.23*
30	0.51	0.74	0.75	0.76	0.72	0.67	0.69	0.73	0.14	0.15	0.23	0.47	0.67	*0.35*
40	0.55	0.80	0.83	0.83	0.80	0.77	0.77	0.80	0.19	0.23	0.31	0.58	0.75	*0.53*
50	0.63	0.87	0.88	0.89	0.86	0.85	0.83	0.86	0.26	0.32	0.41	0.69	0.83	*0.62*
60	0.69	0.91	0.92	0.92	0.91	0.89	0.88	0.90	0.36	0.43	0.52	0.78	0.90	*0.71*
70	0.74	0.94	0.95	0.95	0.94	0.92	0.92	0.93	0.47	0.57	0.65	0.85	0.95	*0.82*
80	0.82	0.97	0.97	0.97	0.96	0.93	0.95	0.96	0.60	0.72	0.78	0.92	0.97	*0.93*
90	0.88	0.99	0.99	0.99	0.98	0.97	0.98	0.98	0.78	0.87	0.91	0.96	0.99	*0.96*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.111: UBC-100 average $\hat{\mathcal{R}}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.02	0.03	0.03	0.01	0.03	0.02	0.02	0.02	0.03	0.02	0.02	0.03	*0.00*
10	0.21	0.27	0.23	0.08	0.26	0.20	0.17	0.21	0.26	0.19	0.15	0.24	*0.04*
20	0.41	0.46	0.42	0.18	0.46	0.38	0.33	0.39	0.45	0.37	0.32	0.43	*0.17*
30	0.57	0.62	0.60	0.37	0.62	0.55	0.52	0.56	0.60	0.55	0.52	0.59	*0.31*
40	0.70	0.74	0.73	0.53	0.75	0.69	0.66	0.70	0.72	0.69	0.66	0.72	*0.50*
50	0.80	0.82	0.82	0.66	0.83	0.78	0.78	0.79	0.81	0.79	0.77	0.82	*0.61*
60	0.87	0.88	0.89	0.78	0.89	0.86	0.87	0.86	0.88	0.87	0.86	0.88	*0.71*
70	0.92	0.93	0.93	0.86	0.94	0.91	0.92	0.91	0.93	0.91	0.92	0.93	*0.82*
80	0.96	0.96	0.96	0.93	0.96	0.95	0.96	0.95	0.96	0.95	0.95	0.96	*0.93*
90	0.99	0.98	0.99	0.98	0.98	0.98	0.99	0.98	0.98	0.98	0.99	0.99	*0.96*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.112: UBC-100 average $\hat{\mathcal{R}}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.04	0.03	0.02	0.04	0.03	0.01	0.03	0.03	0.00	0.00	0.00	0.01	0.02	*0.00*
10	0.28	0.27	0.18	0.26	0.26	0.14	0.23	0.26	0.03	0.02	0.02	0.08	0.21	*0.04*
20	0.47	0.49	0.34	0.49	0.48	0.37	0.43	0.48	0.06	0.05	0.05	0.17	0.41	*0.17*
30	0.63	0.66	0.49	0.65	0.64	0.56	0.59	0.63	0.10	0.09	0.09	0.27	0.57	*0.31*
40	0.75	0.78	0.62	0.77	0.75	0.69	0.71	0.75	0.15	0.14	0.14	0.38	0.70	*0.50*
50	0.83	0.86	0.72	0.85	0.83	0.80	0.80	0.82	0.22	0.22	0.23	0.50	0.80	*0.61*
60	0.89	0.90	0.81	0.90	0.88	0.87	0.86	0.87	0.31	0.34	0.34	0.61	0.87	*0.71*
70	0.92	0.93	0.88	0.94	0.92	0.91	0.91	0.91	0.44	0.49	0.50	0.72	0.92	*0.82*
80	0.95	0.95	0.94	0.96	0.95	0.94	0.95	0.95	0.61	0.65	0.66	0.85	0.96	*0.93*
90	0.98	0.97	0.98	0.98	0.97	0.97	0.97	0.97	0.80	0.84	0.84	0.95	0.99	*0.96*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.113: UBC-100 average $\hat{\mathcal{R}}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.03	0.03	0.03	0.01	0.04	0.02	0.02	0.02	0.03	0.02	0.02	0.03	*0.00*
10	0.24	0.28	0.26	0.10	0.28	0.22	0.20	0.21	0.25	0.20	0.18	0.26	*0.04*
20	0.43	0.48	0.46	0.20	0.48	0.41	0.36	0.39	0.45	0.38	0.33	0.46	*0.17*
30	0.59	0.65	0.62	0.40	0.64	0.57	0.55	0.54	0.61	0.55	0.52	0.62	*0.31*
40	0.71	0.77	0.74	0.58	0.77	0.70	0.69	0.67	0.73	0.68	0.66	0.74	*0.50*
50	0.82	0.86	0.83	0.72	0.86	0.81	0.79	0.78	0.83	0.79	0.78	0.83	*0.61*
60	0.89	0.92	0.90	0.83	0.92	0.89	0.88	0.87	0.89	0.88	0.87	0.90	*0.71*
70	0.95	0.96	0.95	0.90	0.95	0.94	0.94	0.93	0.94	0.94	0.94	0.95	*0.82*
80	0.97	0.98	0.97	0.96	0.97	0.97	0.97	0.96	0.97	0.97	0.97	0.97	*0.93*
90	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	*0.96*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.114: UBC-100 average $\hat{\mathcal{R}}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.03	0.03	0.04	0.04	0.03	0.02	0.03	0.04	0.00	0.00	0.00	0.01	0.03	*0.00*
10	0.19	0.27	0.28	0.29	0.26	0.19	0.24	0.27	0.03	0.03	0.05	0.12	0.24	*0.04*
20	0.32	0.48	0.48	0.50	0.47	0.40	0.44	0.48	0.07	0.07	0.11	0.27	0.43	*0.17*
30	0.44	0.65	0.66	0.67	0.64	0.59	0.61	0.64	0.12	0.13	0.20	0.42	0.59	*0.31*
40	0.52	0.76	0.79	0.78	0.76	0.73	0.73	0.76	0.18	0.22	0.29	0.55	0.71	*0.50*
50	0.61	0.85	0.86	0.87	0.85	0.83	0.82	0.84	0.26	0.31	0.40	0.68	0.82	*0.61*
60	0.68	0.90	0.91	0.92	0.90	0.89	0.88	0.90	0.36	0.43	0.52	0.77	0.89	*0.71*
70	0.74	0.94	0.95	0.95	0.94	0.92	0.92	0.93	0.46	0.57	0.65	0.85	0.95	*0.82*
80	0.82	0.97	0.97	0.97	0.96	0.93	0.95	0.96	0.60	0.72	0.78	0.92	0.97	*0.93*
90	0.88	0.99	0.99	0.99	0.98	0.97	0.98	0.98	0.78	0.87	0.91	0.96	0.99	*0.96*
100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.115: UBC-100 average $\mathcal{P}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.54	0.77	0.63	0.26	0.69	0.49	0.41	0.57	0.68	0.53	0.45	0.66	*0.09*
10	0.62	0.81	0.66	0.29	0.76	0.59	0.47	0.59	0.74	0.56	0.44	0.69	*0.36*
20	0.66	0.79	0.70	0.35	0.76	0.63	0.53	0.63	0.73	0.62	0.50	0.71	*0.53*
30	0.68	0.77	0.72	0.46	0.75	0.66	0.60	0.66	0.72	0.66	0.59	0.71	*0.59*
40	0.68	0.74	0.70	0.52	0.73	0.66	0.62	0.66	0.70	0.67	0.61	0.70	*0.63*
50	0.67	0.72	0.68	0.55	0.70	0.65	0.64	0.65	0.68	0.66	0.62	0.69	*0.61*
60	0.65	0.68	0.65	0.57	0.67	0.64	0.63	0.63	0.65	0.65	0.63	0.66	*0.59*
70	0.62	0.64	0.62	0.58	0.63	0.61	0.61	0.61	0.62	0.61	0.61	0.62	*0.60*
80	0.58	0.59	0.58	0.57	0.59	0.58	0.58	0.58	0.58	0.58	0.58	0.58	*0.60*
90	0.55	0.56	0.55	0.55	0.55	0.55	0.55	0.55	0.55	0.55	0.55	0.55	*0.56*
100	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	*0.52*

Table D.116: UBC-100 average $\mathcal{P}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.83	0.90	0.58	0.88	0.85	0.35	0.85	0.87	0.13	0.10	0.10	0.37	0.54	*0.09*
10	0.83	0.88	0.63	0.88	0.87	0.54	0.84	0.87	0.17	0.16	0.16	0.46	0.62	*0.36*
20	0.81	0.87	0.66	0.88	0.85	0.70	0.82	0.86	0.22	0.22	0.22	0.51	0.66	*0.53*
30	0.80	0.86	0.67	0.85	0.83	0.74	0.80	0.83	0.25	0.26	0.26	0.53	0.68	*0.59*
40	0.77	0.83	0.68	0.82	0.79	0.75	0.78	0.80	0.28	0.30	0.30	0.56	0.68	*0.63*
50	0.74	0.79	0.68	0.78	0.75	0.74	0.74	0.75	0.32	0.36	0.36	0.58	0.67	*0.61*
60	0.69	0.72	0.65	0.72	0.70	0.71	0.68	0.70	0.37	0.41	0.42	0.59	0.65	*0.59*
70	0.64	0.67	0.62	0.66	0.65	0.66	0.64	0.65	0.42	0.46	0.46	0.58	0.62	*0.60*
80	0.59	0.61	0.59	0.61	0.60	0.61	0.60	0.60	0.46	0.49	0.49	0.58	0.58	*0.60*
90	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.49	0.51	0.51	0.56	0.55	*0.56*
100	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	*0.52*

Table D.117: UBC-100 average $\mathcal{P}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.64	0.81	0.62	0.38	0.76	0.62	0.53	0.54	0.66	0.55	0.46	0.72	*0.09*
10	0.66	0.83	0.69	0.34	0.79	0.63	0.53	0.59	0.72	0.58	0.50	0.73	*0.36*
20	0.70	0.81	0.74	0.37	0.79	0.68	0.57	0.62	0.73	0.63	0.53	0.75	*0.53*
30	0.70	0.80	0.73	0.49	0.78	0.68	0.63	0.64	0.73	0.65	0.60	0.74	*0.59*
40	0.69	0.78	0.72	0.55	0.76	0.68	0.64	0.64	0.71	0.65	0.61	0.73	*0.63*
50	0.68	0.76	0.70	0.59	0.74	0.68	0.65	0.64	0.69	0.66	0.62	0.71	*0.61*
60	0.66	0.71	0.67	0.61	0.69	0.66	0.65	0.63	0.67	0.65	0.63	0.67	*0.59*
70	0.64	0.66	0.63	0.60	0.65	0.63	0.63	0.62	0.63	0.63	0.62	0.64	*0.60*
80	0.59	0.61	0.59	0.58	0.60	0.59	0.59	0.58	0.59	0.59	0.59	0.59	*0.60*
90	0.56	0.56	0.56	0.55	0.56	0.56	0.56	0.55	0.55	0.55	0.55	0.56	*0.56*
100	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	*0.52*

Table D.118: UBC-100 average $\mathcal{P}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AviCTF	NSCQ	Doddle	*SBR*
1	0.66	0.87	0.87	0.94	0.85	0.48	0.91	0.88	0.20	0.26	0.31	0.56	0.64	*0.09*
10	0.61	0.87	0.85	0.91	0.85	0.62	0.86	0.87	0.22	0.26	0.33	0.61	0.66	*0.36*
20	0.60	0.85	0.84	0.89	0.85	0.74	0.84	0.86	0.26	0.30	0.39	0.68	0.70	*0.53*
30	0.61	0.84	0.83	0.87	0.83	0.77	0.83	0.85	0.29	0.34	0.43	0.70	0.70	*0.59*
40	0.58	0.82	0.81	0.84	0.81	0.78	0.79	0.81	0.31	0.39	0.47	0.71	0.69	*0.63*
50	0.58	0.78	0.78	0.80	0.78	0.77	0.75	0.77	0.35	0.43	0.50	0.69	0.68	*0.61*
60	0.58	0.72	0.72	0.73	0.72	0.72	0.69	0.71	0.39	0.47	0.52	0.67	0.66	*0.59*
70	0.54	0.67	0.67	0.67	0.66	0.66	0.65	0.66	0.42	0.50	0.54	0.63	0.64	*0.60*
80	0.53	0.62	0.61	0.62	0.61	0.61	0.60	0.61	0.45	0.51	0.54	0.60	0.59	*0.60*
90	0.52	0.57	0.56	0.57	0.56	0.56	0.56	0.56	0.49	0.52	0.54	0.56	0.56	*0.56*
100	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	*0.52*

Table D.119: UBC-100 average precision within the top 5 collections, and the number of queries (out of 100) for which the first ranked collection is correct, for the various configurations of the Doddle algorithm.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Q_s	Q_l	Q_s		Q_l	
Doddle	0.18	0.22	5	*(5%)*	6	*(6%)*
Doddle_RC	0.23	0.26	8	*(8%)*	10	*(10%)*
Doddle_RP	0.19	0.24	12	*(12%)*	8	*(8%)*
Doddle_RF	0.05	0.08	0	*(0%)*	2	*(2%)*
Doddle_RC+RP	0.21	0.27	10	*(10%)*	10	*(10%)*
Doddle_RC+RF	0.16	0.20	3	*(3%)*	3	*(3%)*
Doddle_RP+RF	0.12	0.17	4	*(4%)*	2	*(2%)*
Doddle_×	0.16	0.17	8	*(8%)*	0	*(0%)*
Doddle_RC×RP	0.20	0.21	13	*(13%)*	6	*(6%)*
Doddle_RC×RF	0.14	0.18	4	*(4%)*	1	*(1%)*
Doddle_RP×RF	0.12	0.16	4	*(4%)*	2	*(2%)*
Doddle_W	0.19	0.24	12	*(12%)*	10	*(10%)*
SBR	*0.00*	*0.00*	0	*(0%)*	0	*(0%)*

Table D.120: UBC-100 average precision within the top 5 collections, and the number of queries (out of 100) for which the first ranked collection is correct, for existing algorithms and Doddle.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Q_s	Q_l	Q_s		Q_l	
bGlOSS	0.26	0.16	15	*(15%)*	7	*(7%)*
CORI	0.23	0.21	11	*(11%)*	8	*(8%)*
Cosine Measure	0.14	0.25	4	*(4%)*	11	*(11%)*
Inner Product	0.21	0.25	13	*(13%)*	14	*(14%)*
Skew	0.19	0.20	9	*(9%)*	7	*(7%)*
Highest-available Similarity	0.07	0.13	2	*(2%)*	5	*(5%)*
CVV	0.16	0.16	9	*(9%)*	5	*(5%)*
DFPROP	0.19	0.22	9	*(9%)*	10	*(10%)*
Distribution of Informative Amt	0.02	0.01	0	*(0%)*	0	*(0%)*
SCS	0.02	0.01	0	*(0%)*	0	*(0%)*
AvICTF	0.01	0.02	0	*(0%)*	0	*(0%)*
NSCQ	0.03	0.06	1	*(1%)*	4	*(4%)*
Doddle	0.18	0.22	5	*(5%)*	6	*(6%)*
SBR	*0.00*	*0.00*	0	*(0%)*	0	*(0%)*

D.6 2LDB-60COL

Table D.121: 2LDB-60COL average Spearman rank correlations, for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		SBR	
Algorithms	Q_s	Q_l	Q_s	Q_l
Doddle	**0.43**	**0.49**	−0.09	−0.08
Doddle_RC	**0.54**	**0.60**	**0.21**	**0.22**
Doddle_RP	**0.45**	**0.51**	−0.13	−0.13
Doddle_RF	0.15	**0.23**	−0.28	−0.26
Doddle_RC+RP	**0.51**	**0.57**	0.04	0.03
Doddle_RC+RF	**0.40**	**0.47**	−0.05	−0.04
Doddle_RP+RF	**0.34**	**0.41**	−0.23	−0.21
Doddle_×	**0.38**	**0.38**	−0.11	−0.17
Doddle_RC×RP	**0.46**	**0.49**	0.01	−0.04
Doddle_RC×RF	**0.40**	**0.42**	−0.03	−0.09
Doddle_RP×RF	**0.32**	**0.36**	−0.27	−0.27
Doddle_W	**0.46**	**0.52**	−0.04	−0.04
SBR	***0.24***	***0.24***	—	—

Table D.122: 2LDB-60COL average Spearman rank correlations, for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		SBR	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.56**	**0.37**	**0.25**	0.15
CORI	**0.63**	**0.63**	**0.48**	**0.43**
Cosine Measure	**0.40**	**0.63**	**0.27**	**0.37**
Inner Product	**0.61**	**0.65**	**0.55**	**0.53**
Skew	**0.57**	**0.60**	**0.59**	**0.58**
Highest-available Similarity	**0.51**	**0.56**	**0.67**	**0.68**
CVV	**0.51**	**0.55**	**0.58**	**0.61**
DFPROP	**0.56**	**0.59**	**0.61**	**0.62**
Distribution of Informative Amt	−0.45	−0.43	−0.36	−0.27
SCS	−0.41	−0.25	−0.10	0.06
AvICTF	−0.40	−0.09	−0.10	0.15
NSCQ	0.13	**0.37**	**0.60**	**0.67**
Doddle	**0.43**	**0.49**	−0.09	−0.08
SBR	***0.24***	***0.24***	—	—

Table D.123: 2LDB-60COL Z-Test results, showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Short queries.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AvICTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
bGlOSS																									
CORI																									
Cosine Measure																									
Inner Product																									
Skew																									
Highest-available Similarity																									
CVV																									
DFPROP																									
Distribution of Informative Amt																									
SCS																									
AvICTF																									
NSCQ																									
Doddle																									
Doddle_RC																									
Doddle_RP																									
Doddle_RF																									
Doddle_RC+RP																									
Doddle_RC+RF																									
Doddle_RP+RF																									
Doddle_×																									
Doddle_RC×RP																									
Doddle_RC×RF																									
Doddle_RP×RF																									
Doddle_W																									
SBR																									

Table D.124: 2LDB-60COL Z-Test results, showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Long queries.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AvICTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
bGlOSS				✓																					
CORI																									
Cosine Measure																									
Inner Product	✓																								
Skew																									
Highest-available Similarity																									
CVV																									
DFPROP																									
Distribution of Informative Amt																									
SCS																									
AvICTF										✓															
NSCQ										✓	✓														
Doddle										✓	✓	✓													
Doddle_RC										✓	✓	✓	✓												
Doddle_RP										✓	✓	✓	✓	✓											
Doddle_RF										✓	✓	✓	✓	✓	✓										
Doddle_RC+RP										✓	✓	✓	✓	✓	✓	✓									
Doddle_RC+RF										✓	✓	✓	✓	✓	✓	✓	✓								
Doddle_RP+RF										✓	✓	✓	✓	✓	✓	✓	✓	✓							
Doddle_×										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓						
Doddle_RC×RP										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓					
Doddle_RC×RF										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				
Doddle_RP×RF										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
Doddle_W										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
SBR										✓	✓														

Table D.125: 2LDB-60COL average Blest and Da Costa weighted rank correlations, for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR.

	Blest		Da Costa	
Algorithms	Q_s	Q_l	Q_s	Q_l
Doddle	**0.38**	**0.43**	**0.45**	**0.51**
Doddle_RC	**0.50**	**0.55**	**0.56**	**0.62**
Doddle_RP	**0.41**	**0.45**	**0.47**	**0.53**
Doddle_RF	0.11	**0.18**	0.14	**0.23**
Doddle_RC+RP	**0.47**	**0.52**	**0.53**	**0.59**
Doddle_RC+RF	**0.36**	**0.41**	**0.42**	**0.48**
Doddle_RP+RF	**0.30**	**0.35**	**0.36**	**0.42**
Doddle_×	**0.35**	**0.34**	**0.41**	**0.40**
Doddle_RC×RP	**0.43**	**0.44**	**0.48**	**0.51**
Doddle_RC×RF	**0.36**	**0.37**	**0.42**	**0.44**
Doddle_RP×RF	**0.28**	**0.31**	**0.34**	**0.37**
Doddle_W	**0.42**	**0.47**	**0.49**	**0.54**
SBR	***0.25***	***0.25***	***0.22***	***0.22***

Table D.126: 2LDB-60COL average Blest and Da Costa weighted rank correlations, for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR.

	Blest		Da Costa	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.50**	**0.22**	**0.60**	**0.55**
CORI	**0.58**	**0.58**	**0.64**	**0.64**
Cosine Measure	**0.38**	**0.58**	**0.41**	**0.64**
Inner Product	**0.56**	**0.61**	**0.61**	**0.66**
Skew	**0.53**	**0.56**	**0.58**	**0.60**
Highest-available Similarity	**0.47**	**0.52**	**0.50**	**0.55**
CVV	**0.46**	**0.51**	**0.52**	**0.56**
DFPROP	**0.52**	**0.55**	**0.57**	**0.60**
Distribution of Informative Amt	−0.42	−0.40	−0.38	−0.38
SCS	−0.38	−0.22	−0.39	−0.24
AvICTF	−0.37	−0.08	−0.38	−0.09
NSCQ	0.12	**0.34**	0.12	**0.35**
Doddle	**0.38**	**0.43**	**0.45**	**0.51**
SBR	***0.25***	***0.25***	***0.22***	***0.22***

Table D.127: 2LDB-60COL average $\mathcal{R}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.25	0.37	0.35	0.07	0.34	0.24	0.19	0.25	0.30	0.21	0.22	0.30	*0.12*
10	0.51	0.59	0.53	0.20	0.59	0.47	0.41	0.49	0.57	0.46	0.39	0.55	*0.19*
20	0.69	0.74	0.72	0.46	0.75	0.67	0.63	0.68	0.72	0.66	0.63	0.72	*0.42*
30	0.82	0.83	0.84	0.66	0.84	0.80	0.79	0.80	0.82	0.80	0.78	0.84	*0.63*
40	0.91	0.91	0.91	0.82	0.92	0.89	0.90	0.89	0.91	0.89	0.90	0.91	*0.76*
50	0.96	0.96	0.96	0.93	0.97	0.95	0.96	0.95	0.96	0.96	0.95	0.96	*0.93*
60	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.128: 2LDB-60COL average $\mathcal{R}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.25	0.39	0.20	0.13	0.13	0.09	0.12	0.13	0.03	0.02	0.02	0.12	0.25	*0.12*
10	0.55	0.60	0.40	0.53	0.52	0.40	0.48	0.53	0.08	0.06	0.06	0.20	0.51	*0.19*
20	0.73	0.77	0.58	0.75	0.74	0.66	0.69	0.73	0.13	0.12	0.12	0.32	0.69	*0.42*
30	0.83	0.87	0.72	0.86	0.84	0.81	0.80	0.83	0.22	0.21	0.22	0.48	0.82	*0.63*
40	0.91	0.92	0.85	0.92	0.90	0.90	0.89	0.90	0.37	0.40	0.40	0.66	0.91	*0.76*
50	0.95	0.95	0.94	0.96	0.95	0.94	0.95	0.95	0.63	0.66	0.66	0.86	0.96	*0.93*
60	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.96	0.95	0.95	0.99	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.129: 2LDB-60COL average $\mathcal{R}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.30	0.36	0.32	0.12	0.38	0.28	0.23	0.19	0.28	0.19	0.18	0.34	*0.12*
10	0.55	0.61	0.58	0.24	0.62	0.53	0.45	0.47	0.56	0.47	0.42	0.59	*0.19*
20	0.71	0.76	0.74	0.49	0.77	0.68	0.66	0.64	0.73	0.66	0.63	0.74	*0.42*
30	0.83	0.88	0.85	0.71	0.87	0.82	0.80	0.79	0.83	0.80	0.79	0.84	*0.63*
40	0.93	0.94	0.93	0.86	0.94	0.92	0.92	0.90	0.92	0.91	0.91	0.93	*0.76*
50	0.98	0.98	0.97	0.95	0.98	0.97	0.97	0.97	0.97	0.97	0.97	0.97	*0.93*
60	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.130: 2LDB-60COL average $\mathcal{R}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.35	0.34	0.17	0.13	0.13	0.12	0.13	0.13	0.03	0.03	0.04	0.12	0.30	*0.12*
10	0.42	0.57	0.56	0.58	0.54	0.44	0.49	0.54	0.10	0.09	0.12	0.29	0.55	*0.19*
20	0.51	0.76	0.77	0.77	0.73	0.71	0.71	0.75	0.14	0.17	0.25	0.50	0.71	*0.42*
30	0.59	0.86	0.87	0.88	0.85	0.84	0.82	0.85	0.26	0.31	0.39	0.67	0.83	*0.63*
40	0.71	0.93	0.93	0.94	0.93	0.91	0.90	0.91	0.41	0.48	0.58	0.80	0.93	*0.76*
50	0.80	0.97	0.97	0.97	0.96	0.94	0.95	0.96	0.64	0.71	0.78	0.92	0.98	*0.93*
60	0.97	0.99	1.00	1.00	1.00	0.99	0.99	0.99	0.95	0.96	0.97	0.98	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.131: 2LDB-60COL average $\hat{\mathcal{R}}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.03	0.05	0.05	0.01	0.05	0.03	0.03	0.04	0.04	0.03	0.03	0.04	*0.02*
10	0.34	0.40	0.36	0.13	0.40	0.32	0.28	0.33	0.38	0.31	0.27	0.37	*0.13*
20	0.62	0.66	0.64	0.41	0.67	0.60	0.56	0.60	0.64	0.59	0.56	0.64	*0.37*
30	0.80	0.81	0.82	0.65	0.82	0.78	0.77	0.78	0.80	0.78	0.76	0.82	*0.62*
40	0.90	0.91	0.91	0.81	0.92	0.89	0.90	0.89	0.90	0.89	0.89	0.91	*0.76*
50	0.96	0.96	0.96	0.93	0.97	0.95	0.96	0.95	0.96	0.96	0.95	0.96	*0.93*
60	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.132: 2LDB-60COL average $\hat{\mathcal{R}}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.04	0.06	0.03	0.02	0.02	0.01	0.02	0.02	0.00	0.00	0.00	0.01	0.03	*0.02*
10	0.37	0.40	0.26	0.36	0.36	0.27	0.33	0.36	0.05	0.04	0.04	0.13	0.34	*0.13*
20	0.65	0.68	0.51	0.67	0.66	0.59	0.63	0.66	0.11	0.10	0.10	0.29	0.62	*0.37*
30	0.81	0.85	0.70	0.84	0.82	0.79	0.78	0.81	0.21	0.20	0.21	0.47	0.80	*0.62*
40	0.91	0.92	0.84	0.92	0.90	0.90	0.88	0.89	0.37	0.39	0.40	0.65	0.90	*0.76*
50	0.95	0.95	0.94	0.96	0.95	0.94	0.95	0.95	0.63	0.66	0.66	0.86	0.96	*0.93*
60	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.96	0.95	0.95	0.99	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.133: 2LDB-60COL average $\hat{\mathcal{R}}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.04	0.05	0.05	0.02	0.05	0.04	0.03	0.02	0.04	0.02	0.02	0.05	*0.02*
10	0.37	0.41	0.39	0.16	0.42	0.36	0.31	0.32	0.38	0.32	0.28	0.39	*0.13*
20	0.63	0.68	0.66	0.44	0.69	0.61	0.59	0.57	0.65	0.59	0.56	0.66	*0.37*
30	0.81	0.85	0.83	0.69	0.84	0.80	0.78	0.77	0.81	0.78	0.77	0.82	*0.62*
40	0.93	0.94	0.93	0.86	0.94	0.92	0.91	0.90	0.92	0.91	0.91	0.93	*0.76*
50	0.97	0.98	0.97	0.95	0.98	0.97	0.97	0.97	0.97	0.97	0.97	0.97	*0.93*
60	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.134: 2LDB-60COL average $\hat{\mathcal{R}}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.05	0.05	0.02	0.02	0.02	0.02	0.02	0.02	0.00	0.00	0.00	0.02	0.04	*0.02*
10	0.27	0.39	0.38	0.39	0.37	0.30	0.33	0.37	0.06	0.06	0.08	0.19	0.37	*0.13*
20	0.45	0.68	0.68	0.69	0.66	0.63	0.64	0.67	0.12	0.15	0.22	0.45	0.63	*0.37*
30	0.58	0.84	0.85	0.86	0.83	0.82	0.80	0.83	0.25	0.30	0.38	0.65	0.81	*0.62*
40	0.70	0.92	0.92	0.93	0.92	0.90	0.89	0.91	0.41	0.48	0.57	0.80	0.93	*0.76*
50	0.80	0.97	0.97	0.97	0.96	0.93	0.95	0.96	0.64	0.71	0.78	0.92	0.97	*0.93*
60	0.97	0.99	1.00	1.00	1.00	0.99	0.99	0.99	0.95	0.96	0.97	0.98	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.135: 2LDB-60COL average $\mathcal{P}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.56	0.80	0.64	0.28	0.72	0.55	0.45	0.58	0.67	0.55	0.47	0.68	*1.00*
10	0.65	0.80	0.68	0.33	0.77	0.62	0.51	0.61	0.73	0.60	0.48	0.71	*0.60*
20	0.69	0.78	0.71	0.50	0.75	0.67	0.61	0.66	0.72	0.68	0.59	0.72	*0.67*
30	0.68	0.74	0.68	0.57	0.72	0.68	0.64	0.66	0.69	0.68	0.63	0.70	*0.65*
40	0.65	0.67	0.65	0.58	0.66	0.64	0.64	0.64	0.65	0.65	0.63	0.65	*0.61*
50	0.59	0.60	0.59	0.57	0.59	0.59	0.59	0.58	0.59	0.59	0.58	0.59	*0.61*
60	0.53	0.54	0.53	0.54	0.54	0.54	0.54	0.53	0.54	0.54	0.54	0.54	*0.54*
62	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	*0.52*

Table D.136: 2LDB-60COL average $\mathcal{P}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.89	0.95	0.61	1.00	1.00	0.76	1.00	1.00	0.14	0.11	0.12	0.95	0.56	*1.00*
10	0.85	0.89	0.67	0.90	0.88	0.75	0.86	0.88	0.22	0.20	0.20	0.60	0.65	*0.60*
20	0.81	0.87	0.69	0.86	0.84	0.79	0.82	0.84	0.26	0.27	0.27	0.58	0.69	*0.67*
30	0.75	0.81	0.69	0.80	0.78	0.77	0.76	0.77	0.32	0.35	0.35	0.60	0.68	*0.65*
40	0.67	0.71	0.65	0.70	0.69	0.70	0.67	0.69	0.39	0.44	0.44	0.60	0.65	*0.61*
50	0.60	0.62	0.60	0.61	0.61	0.61	0.61	0.61	0.46	0.50	0.50	0.59	0.59	*0.61*
60	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.51	0.53	0.53	0.54	0.53	*0.54*
62	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	*0.52*

Table D.137: 2LDB-60COL average $\mathcal{P}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.66	0.80	0.62	0.44	0.76	0.65	0.52	0.51	0.63	0.53	0.48	0.70	*1.00*
10	0.70	0.83	0.73	0.39	0.80	0.69	0.57	0.61	0.73	0.62	0.52	0.75	*0.60*
20	0.72	0.81	0.74	0.54	0.79	0.71	0.65	0.64	0.73	0.67	0.61	0.75	*0.67*
30	0.70	0.78	0.71	0.60	0.75	0.70	0.66	0.65	0.70	0.67	0.63	0.72	*0.65*
40	0.67	0.69	0.66	0.61	0.68	0.66	0.65	0.64	0.66	0.66	0.64	0.67	*0.61*
50	0.60	0.61	0.59	0.59	0.60	0.60	0.60	0.59	0.59	0.60	0.59	0.60	*0.61*
60	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	*0.54*
62	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	*0.52*

Table D.138: 2LDB-60COL average $\mathcal{P}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.70	0.92	0.92	1.00	1.00	0.96	1.00	1.00	0.19	0.30	0.36	1.00	0.66	*1.00*
10	0.65	0.89	0.87	0.92	0.88	0.79	0.89	0.90	0.27	0.31	0.39	0.72	0.70	*0.60*
20	0.61	0.86	0.84	0.88	0.85	0.82	0.84	0.86	0.30	0.36	0.46	0.73	0.72	*0.67*
30	0.57	0.81	0.80	0.82	0.79	0.80	0.77	0.79	0.35	0.44	0.50	0.71	0.70	*0.65*
40	0.56	0.71	0.70	0.71	0.70	0.71	0.68	0.70	0.41	0.49	0.54	0.67	0.67	*0.61*
50	0.53	0.62	0.62	0.62	0.61	0.61	0.61	0.61	0.46	0.52	0.55	0.61	0.60	*0.61*
60	0.52	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.51	0.53	0.53	0.54	0.54	*0.54*
62	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	*0.52*

Table D.139: 2LDB-60COL average precision within the top 5 collections, and the number of queries (out of 100) for which the first ranked collection is correct, for the various configurations of the Doddle algorithm.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Q_s	Q_l	Q_s		Q_l	
Doddle	0.22	0.29	8	*(8%)*	9	*(9%)*
Doddle_RC	0.31	0.32	10	*(10%)*	11	*(11%)*
Doddle_RP	0.26	0.31	15	*(15%)*	12	*(12%)*
Doddle_RF	0.06	0.08	0	*(0%)*	3	*(3%)*
Doddle_RC+RP	0.30	0.33	14	*(14%)*	13	*(13%)*
Doddle_RC+RF	0.21	0.23	7	*(7%)*	8	*(8%)*
Doddle_RP+RF	0.16	0.22	6	*(6%)*	7	*(7%)*
Doddle_×	0.22	0.22	9	*(9%)*	3	*(3%)*
Doddle_RC×RP	0.30	0.28	12	*(12%)*	8	*(8%)*
Doddle_RC×RF	0.19	0.20	5	*(5%)*	3	*(3%)*
Doddle_RP×RF	0.15	0.18	9	*(9%)*	3	*(3%)*
Doddle_W	0.26	0.31	10	*(10%)*	11	*(11%)*
SBR	*0.01*	*0.01*	*0*	*(0%)*	*0*	*(0%)*

Table D.140: 2LDB-60COL average precision within the top 5 collections, and the number of queries (out of 100) for which the first ranked collection is correct, for existing algorithms and Doddle.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Q_s	Q_l	Q_s		Q_l	
bGlOSS	0.23	0.21	10	*(10%)*	14	*(14%)*
CORI	0.27	0.26	11	*(11%)*	6	*(6%)*
Cosine Measure	0.16	0.22	5	*(5%)*	2	*(2%)*
Inner Product	0.18	0.22	0	*(0%)*	0	*(0%)*
Skew	0.18	0.19	0	*(0%)*	0	*(0%)*
Highest-available Similarity	0.08	0.12	0	*(0%)*	0	*(0%)*
CVV	0.15	0.15	0	*(0%)*	0	*(0%)*
DFPROP	0.17	0.20	0	*(0%)*	0	*(0%)*
Distribution of Informative Amt	0.01	0.01	0	*(0%)*	0	*(0%)*
SCS	0.01	0.01	0	*(0%)*	0	*(0%)*
AvICTF	0.01	0.02	0	*(0%)*	0	*(0%)*
NSCQ	0.03	0.05	0	*(0%)*	0	*(0%)*
Doddle	0.22	0.29	8	*(8%)*	9	*(9%)*
SBR	*0.01*	*0.01*	*0*	*(0%)*	*0*	*(0%)*

D.7 AP-WSJ-60COL

Table D.141: AP-WSJ-60COL average Spearman rank correlations, for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		SBR	
Algorithms	Q_s	Q_l	Q_s	Q_l
Doddle	**0.35**	**0.41**	−0.33	−0.36
Doddle_RC	**0.45**	**0.52**	0.12	0.09
Doddle_RP	**0.35**	**0.41**	−0.43	−0.46
Doddle_RF	0.07	0.15	−0.55	−0.55
Doddle_RC+RP	**0.42**	**0.48**	−0.19	−0.24
Doddle_RC+RF	**0.34**	**0.41**	−0.23	−0.25
Doddle_RP+RF	**0.25**	**0.33**	−0.53	−0.53
Doddle_×	**0.33**	**0.37**	−0.36	−0.42
Doddle_RC×RP	**0.41**	**0.45**	−0.18	−0.27
Doddle_RC×RF	**0.33**	**0.38**	−0.24	−0.31
Doddle_RP×RF	**0.23**	**0.29**	−0.56	−0.59
Doddle_W	**0.38**	**0.45**	−0.28	−0.31
SBR	*0.13*	*0.13*	—	—

Table D.142: AP-WSJ-60COL average Spearman rank correlations, for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		SBR	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.44**	**0.35**	−0.02	0.04
CORI	**0.50**	**0.53**	**0.43**	**0.32**
Cosine Measure	**0.36**	**0.53**	0.18	**0.22**
Inner Product	**0.48**	**0.54**	**0.42**	**0.37**
Skew	**0.41**	**0.46**	**0.46**	**0.45**
Highest-available Similarity	**0.36**	**0.38**	**0.80**	**0.81**
CVV	**0.34**	**0.36**	**0.46**	**0.49**
DFPROP	**0.39**	**0.45**	**0.48**	**0.49**
Distribution of Informative Amt	−0.36	−0.37	−0.25	−0.17
SCS	−0.29	−0.15	−0.01	0.14
AvICTF	−0.28	−0.03	−0.01	0.17
NSCQ	**0.22**	**0.34**	**0.71**	**0.69**
Doddle	**0.35**	**0.41**	−0.33	−0.36
SBR	*0.13*	*0.13*	—	—

Table D.143: AP-WSJ-60COL Z-Test results, showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Short queries.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AvICTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
bGlOSS																✓									✓
CORI																✓									✓
Cosine Measure																									
Inner Product																✓									✓
Skew																✓									
Highest-available Similarity																									
CVV																									
DFPROP																									
Distribution of Informative Amt																									
SCS																									
AvICTF	✓	✓	✓	✓	✓	✓	✓	✓																	
NSCQ	✓	✓	✓	✓	✓	✓	✓	✓																	
Doddle	✓	✓	✓	✓	✓	✓	✓	✓																	
Doddle_RC	✓	✓	✓	✓	✓	✓	✓	✓								✓									
Doddle_RP	✓	✓	✓	✓	✓	✓	✓	✓																	
Doddle_RF	✓	✓	✓	✓	✓	✓	✓	✓						✓											
Doddle_RC+RP	✓	✓	✓	✓	✓	✓	✓	✓								✓									
Doddle_RC+RF	✓	✓	✓	✓	✓	✓	✓	✓																	
Doddle_RP+RF	✓	✓	✓	✓	✓	✓	✓	✓																	
Doddle_×	✓	✓	✓	✓	✓	✓	✓	✓																	
Doddle_RC×RP	✓	✓	✓	✓	✓	✓	✓	✓								✓									
Doddle_RC×RF	✓	✓	✓	✓	✓	✓	✓	✓																	
Doddle_RP×RF	✓	✓	✓	✓	✓	✓	✓	✓																	
Doddle_W	✓	✓	✓	✓	✓	✓	✓	✓																	
SBR	✓	✓	✓	✓												✓									

Table D.144: AP-WSJ-60COL Z-Test results, showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Long queries.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AvICTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
bGlOSS																									
CORI																									
Cosine Measure																									
Inner Product																									
Skew																									
Highest-available Similarity																									
CVV																									
DFPROP																									
Distribution of Informative Amt																									
SCS																									
AvICTF																									
NSCQ																									
Doddle																									
Doddle_RC																									
Doddle_RP																									
Doddle_RF																									
Doddle_RC+RP																									
Doddle_RC+RF																									
Doddle_RP+RF																									
Doddle_×																									
Doddle_RC×RP																									
Doddle_RC×RF																									
Doddle_RP×RF																									
Doddle_W																									
SBR																									

Table D.145: AP-WSJ-60COL average Blest and Da Costa weighted rank correlations, for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR.

	Blest		Da Costa	
Algorithms	Q_s	Q_l	Q_s	Q_l
Doddle	**0.21**	**0.24**	**0.45**	**0.50**
Doddle_RC	**0.36**	**0.39**	**0.54**	**0.60**
Doddle_RP	**0.23**	**0.24**	**0.44**	**0.50**
Doddle_RF	−0.04	0.01	0.18	**0.24**
Doddle_RC+RP	**0.30**	**0.32**	**0.51**	**0.57**
Doddle_RC+RF	0.20	**0.24**	**0.43**	**0.49**
Doddle_RP+RF	0.11	0.16	**0.34**	**0.41**
Doddle_×	0.19	0.19	**0.42**	**0.45**
Doddle_RC×RP	**0.30**	**0.29**	**0.50**	**0.54**
Doddle_RC×RF	0.20	0.20	**0.42**	**0.46**
Doddle_RP×RF	0.08	0.11	**0.33**	**0.38**
Doddle_W	**0.25**	**0.28**	**0.47**	**0.53**
SBR	***0.27***	***0.27***	***0.21***	***0.21***

Table D.146: AP-WSJ-60COL average Blest and Da Costa weighted rank correlations, for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR.

	Blest		Da Costa	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.39**	**0.35**	**0.54**	**0.61**
CORI	**0.50**	**0.52**	**0.58**	**0.61**
Cosine Measure	**0.36**	**0.50**	**0.46**	**0.61**
Inner Product	**0.49**	**0.54**	**0.56**	**0.62**
Skew	**0.43**	**0.47**	**0.49**	**0.54**
Highest-available Similarity	**0.37**	**0.40**	**0.42**	**0.45**
CVV	**0.34**	**0.38**	**0.42**	**0.44**
DFPROP	**0.40**	**0.47**	**0.47**	**0.52**
Distribution of Informative Amt	−0.33	−0.32	−0.15	−0.18
SCS	−0.20	−0.04	−0.13	−0.02
AvICTF	−0.20	0.05	−0.12	0.08
NSCQ	**0.27**	**0.37**	**0.30**	**0.41**
Doddle	**0.21**	**0.24**	**0.45**	**0.50**
SBR	***0.27***	***0.27***	***0.21***	***0.21***

Table D.147: AP-WSJ-60COL average $\mathcal{R}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.39	0.47	0.42	0.09	0.45	0.33	0.27	0.37	0.47	0.27	0.26	0.44	*0.50*
10	0.52	0.62	0.52	0.22	0.61	0.48	0.40	0.48	0.60	0.46	0.36	0.56	*0.27*
20	0.71	0.78	0.71	0.41	0.77	0.69	0.58	0.68	0.76	0.67	0.55	0.74	*0.32*
30	0.86	0.87	0.85	0.62	0.88	0.85	0.81	0.84	0.87	0.84	0.80	0.86	*0.52*
40	0.92	0.92	0.91	0.78	0.93	0.92	0.90	0.92	0.93	0.92	0.89	0.93	*0.81*
50	0.97	0.96	0.96	0.92	0.97	0.97	0.96	0.97	0.97	0.96	0.96	0.97	*0.90*
60	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.148: AP-WSJ-60COL average $\mathcal{R}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.55	0.52	0.28	0.52	0.51	0.20	0.52	0.52	0.04	0.02	0.02	0.31	0.39	*0.50*
10	0.58	0.64	0.46	0.58	0.52	0.39	0.47	0.50	0.07	0.07	0.07	0.37	0.52	*0.27*
20	0.74	0.78	0.62	0.75	0.69	0.70	0.62	0.67	0.14	0.13	0.13	0.51	0.71	*0.32*
30	0.83	0.86	0.78	0.86	0.81	0.79	0.73	0.78	0.20	0.25	0.25	0.62	0.86	*0.52*
40	0.90	0.91	0.88	0.91	0.89	0.84	0.84	0.88	0.32	0.38	0.39	0.78	0.92	*0.81*
50	0.95	0.94	0.95	0.95	0.94	0.92	0.93	0.93	0.51	0.64	0.65	0.91	0.97	*0.90*
60	0.99	0.99	1.00	1.00	0.99	0.99	0.99	0.99	0.82	0.92	0.93	0.99	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.149: AP-WSJ-60COL average $\mathcal{R}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.46	0.53	0.47	0.18	0.52	0.45	0.38	0.33	0.47	0.30	0.31	0.51	*0.50*
10	0.60	0.66	0.60	0.29	0.67	0.56	0.49	0.53	0.64	0.50	0.44	0.63	*0.27*
20	0.75	0.82	0.76	0.45	0.79	0.74	0.67	0.71	0.78	0.71	0.62	0.77	*0.32*
30	0.87	0.91	0.88	0.71	0.91	0.88	0.84	0.85	0.90	0.86	0.83	0.89	*0.52*
40	0.94	0.97	0.94	0.87	0.96	0.94	0.92	0.94	0.95	0.94	0.92	0.95	*0.81*
50	0.98	0.99	0.98	0.96	0.98	0.98	0.98	0.97	0.98	0.98	0.97	0.98	*0.90*
60	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.150: AP-WSJ-60COL average $\mathcal{R}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.42	0.53	0.53	0.56	0.54	0.37	0.52	0.53	0.04	0.03	0.06	0.39	0.46	*0.50*
10	0.43	0.66	0.65	0.64	0.57	0.47	0.46	0.57	0.06	0.12	0.18	0.46	0.60	*0.27*
20	0.55	0.79	0.80	0.79	0.73	0.71	0.63	0.72	0.13	0.22	0.31	0.64	0.75	*0.32*
30	0.61	0.87	0.90	0.88	0.83	0.78	0.76	0.82	0.21	0.33	0.44	0.73	0.87	*0.52*
40	0.68	0.93	0.95	0.94	0.90	0.83	0.87	0.90	0.34	0.52	0.62	0.84	0.94	*0.81.*
50	0.72	0.97	0.97	0.98	0.97	0.92	0.94	0.96	0.52	0.76	0.82	0.92	0.98	*0.90*
60	0.95	1.00	1.00	1.00	1.00	1.00	0.99	1.00	0.82	0.95	0.97	0.99	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.151: AP-WSJ-60COL average $\hat{\mathcal{R}}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.09	0.12	0.09	0.02	0.11	0.08	0.07	0.09	0.11	0.07	0.06	0.11	*0.13*
10	0.44	0.53	0.44	0.18	0.51	0.41	0.33	0.40	0.51	0.39	0.30	0.47	*0.24*
20	0.68	0.75	0.68	0.39	0.73	0.66	0.56	0.65	0.73	0.65	0.53	0.71	*0.31*
30	0.85	0.86	0.84	0.61	0.88	0.84	0.80	0.83	0.86	0.83	0.79	0.85	*0.51*
40	0.92	0.92	0.91	0.78	0.93	0.92	0.90	0.92	0.93	0.92	0.89	0.93	*0.81*
50	0.97	0.96	0.96	0.92	0.97	0.97	0.96	0.97	0.97	0.96	0.96	0.97	*0.90*
60	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.152: AP-WSJ-60COL average $\hat{\mathcal{R}}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.13	0.12	0.06	0.12	0.13	0.05	0.13	0.13	0.00	0.00	0.00	0.08	0.09	*0.13*
10	0.49	0.54	0.39	0.50	0.45	0.35	0.40	0.43	0.05	0.05	0.05	0.33	0.44	*0.24*
20	0.70	0.75	0.59	0.72	0.67	0.67	0.59	0.65	0.12	0.12	0.12	0.49	0.68	*0.31*
30	0.82	0.85	0.77	0.85	0.80	0.78	0.72	0.77	0.19	0.24	0.25	0.62	0.85	*0.51*
40	0.90	0.90	0.88	0.91	0.89	0.84	0.84	0.88	0.32	0.38	0.38	0.78	0.92	*0.81*
50	0.95	0.94	0.95	0.95	0.94	0.92	0.93	0.93	0.51	0.64	0.65	0.91	0.97	*0.90*
60	0.99	0.99	1.00	1.00	0.99	0.99	0.99	0.99	0.82	0.92	0.93	0.99	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.153: AP-WSJ-60COL average $\hat{\mathcal{R}}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.12	0.13	0.11	0.04	0.12	0.11	0.09	0.08	0.11	0.08	0.08	0.13	*0.13*
10	0.51	0.56	0.51	0.24	0.56	0.48	0.41	0.45	0.54	0.42	0.36	0.53	*0.24*
20	0.72	0.78	0.73	0.43	0.76	0.71	0.65	0.69	0.75	0.68	0.60	0.74	*0.31*
30	0.86	0.90	0.87	0.71	0.90	0.87	0.83	0.84	0.89	0.85	0.82	0.88	*0.51*
40	0.94	0.96	0.93	0.87	0.96	0.94	0.92	0.94	0.95	0.94	0.92	0.94	*0.81*
50	0.98	0.99	0.98	0.96	0.98	0.98	0.98	0.97	0.98	0.98	0.97	0.98	*0.90*
60	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.154: AP-WSJ-60COL average $\hat{\mathcal{R}}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.09	0.12	0.13	0.13	0.13	0.10	0.13	0.13	0.01	0.01	0.01	0.11	0.12	*0.13*
10	0.35	0.56	0.55	0.55	0.49	0.42	0.39	0.49	0.05	0.10	0.15	0.41	0.51	*0.24*
20	0.52	0.76	0.77	0.76	0.70	0.69	0.60	0.69	0.12	0.21	0.30	0.62	0.72	*0.31*
30	0.60	0.86	0.89	0.87	0.82	0.78	0.75	0.81	0.21	0.32	0.43	0.73	0.86	*0.51*
40	0.67	0.93	0.95	0.93	0.90	0.83	0.86	0.90	0.33	0.52	0.62	0.84	0.94	*0.81*
50	0.72	0.97	0.97	0.98	0.97	0.92	0.94	0.96	0.52	0.76	0.82	0.92	0.98	*0.90*
60	0.95	1.00	1.00	1.00	1.00	1.00	0.99	1.00	0.82	0.95	0.97	0.99	1.00	*0.98*
62	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.155: AP-WSJ-60COL average $\mathcal{P}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.65	0.81	0.63	0.26	0.74	0.53	0.44	0.63	0.73	0.51	0.45	0.70	*0.99*
10	0.53	0.68	0.52	0.26	0.63	0.51	0.39	0.49	0.62	0.48	0.37	0.57	*0.39*
20	0.48	0.58	0.48	0.29	0.54	0.48	0.38	0.46	0.53	0.47	0.36	0.51	*0.34*
30	0.45	0.49	0.45	0.34	0.48	0.45	0.42	0.45	0.47	0.46	0.41	0.46	*0.37*
40	0.41	0.43	0.40	0.36	0.42	0.41	0.39	0.41	0.41	0.41	0.39	0.41	*0.42*
50	0.37	0.38	0.37	0.36	0.37	0.37	0.37	0.37	0.37	0.37	0.37	0.37	*0.39*
60	0.35	0.35	0.35	0.35	0.35	0.34	0.35	0.35	0.35	0.35	0.35	0.35	*0.35*
62	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	*0.34*

Table D.156: AP-WSJ-60COL average $\mathcal{P}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.91	0.94	0.59	0.99	0.99	0.44	0.99	0.99	0.12	0.12	0.12	0.81	0.65	*0.99*
10	0.65	0.75	0.57	0.71	0.64	0.51	0.62	0.63	0.16	0.17	0.17	0.51	0.53	*0.39*
20	0.55	0.61	0.50	0.60	0.56	0.57	0.52	0.55	0.19	0.21	0.21	0.46	0.48	*0.34*
30	0.48	0.52	0.47	0.51	0.49	0.49	0.46	0.48	0.21	0.24	0.24	0.42	0.45	*0.37*
40	0.42	0.45	0.42	0.44	0.44	0.43	0.42	0.43	0.24	0.27	0.27	0.40	0.41	*0.42*
50	0.38	0.39	0.39	0.39	0.39	0.39	0.38	0.38	0.27	0.31	0.31	0.39	0.37	*0.39*
60	0.34	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.32	0.34	0.34	0.35	0.35	*0.35*
62	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	*0.34*

Table D.157: AP-WSJ-60COL average $\mathcal{P}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.73	0.88	0.72	0.42	0.81	0.72	0.60	0.62	0.78	0.60	0.55	0.80	*0.99*
10	0.58	0.72	0.59	0.32	0.67	0.57	0.47	0.52	0.64	0.51	0.43	0.63	*0.39*
20	0.51	0.60	0.51	0.30	0.56	0.52	0.44	0.48	0.55	0.49	0.41	0.54	*0.34*
30	0.47	0.52	0.46	0.36	0.50	0.48	0.43	0.45	0.49	0.46	0.42	0.49	*0.37*
40	0.42	0.45	0.42	0.38	0.43	0.43	0.41	0.41	0.43	0.42	0.40	0.43	*0.42*
50	0.38	0.39	0.38	0.37	0.38	0.38	0.38	0.38	0.38	0.38	0.37	0.38	*0.39*
60	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	*0.35*
62	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	*0.34*

Table D.158: AP-WSJ-60COL average $\mathcal{P}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.67	0.95	0.92	1.00	1.00	0.68	0.99	0.99	0.15	0.28	0.38	0.97	0.73	*0.99*
10	0.50	0.76	0.73	0.77	0.69	0.56	0.60	0.68	0.16	0.24	0.29	0.58	0.58	*0.39*
20	0.45	0.62	0.61	0.63	0.58	0.58	0.52	0.58	0.20	0.26	0.31	0.53	0.51	*0.34*
30	0.40	0.53	0.52	0.53	0.51	0.49	0.46	0.50	0.21	0.28	0.33	0.47	0.47	*0.37*
40	0.35	0.46	0.46	0.46	0.44	0.43	0.42	0.44	0.24	0.31	0.34	0.43	0.42	*0.42*
50	0.32	0.40	0.40	0.40	0.40	0.39	0.39	0.39	0.28	0.33	0.35	0.39	0.38	*0.39*
60	0.33	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.32	0.34	0.34	0.35	0.35	*0.35*
62	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	*0.34*

Table D.159: AP-WSJ-60COL average precision within the top 5 collections, and the number of queries (out of 100) for which the first ranked collection is correct, for the various configurations of the Doddle algorithm.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Q_s	Q_l	Q_s		Q_l	
Doddle	0.28	0.34	19	*(19%)*	25	*(25%)*
Doddle_RC	0.34	0.38	25	*(25%)*	28	*(28%)*
Doddle_RP	0.28	0.34	22	*(22%)*	23	*(23%)*
Doddle_RF	0.09	0.15	3	*(3%)*	7	*(7%)*
Doddle_RC+RP	0.33	0.39	26	*(26%)*	25	*(25%)*
Doddle_RC+RF	0.25	0.32	17	*(17%)*	25	*(25%)*
Doddle_RP+RF	0.21	0.26	13	*(13%)*	19	*(19%)*
Doddle_×	0.25	0.30	16	*(16%)*	14	*(14%)*
Doddle_RC×RP	0.33	0.36	26	*(26%)*	24	*(24%)*
Doddle_RC×RF	0.23	0.29	12	*(12%)*	14	*(14%)*
Doddle_RP×RF	0.20	0.26	11	*(11%)*	16	*(16%)*
Doddle_W	0.31	0.36	24	*(24%)*	26	*(26%)*
SBR	*0.22*	*0.22*	*19*	*(19%)*	*19*	*(19%)*

Table D.160: AP-WSJ-60COL average precision within the top 5 collections, and the number of queries (out of 100) for which the first ranked collection is correct, for existing algorithms and Doddle.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Q_s	Q_l	Q_s		Q_l	
bGlOSS	0.35	0.22	25	*(25%)*	22	*(22%)*
CORI	0.36	0.36	22	*(22%)*	23	*(23%)*
Cosine Measure	0.23	0.38	10	*(10%)*	24	*(24%)*
Inner Product	0.34	0.36	21	*(21%)*	24	*(24%)*
Skew	0.32	0.33	20	*(20%)*	23	*(23%)*
Highest-available Similarity	0.18	0.25	9	*(9%)*	20	*(20%)*
CVV	0.29	0.28	21	*(21%)*	21	*(21%)*
DFPROP	0.31	0.32	19	*(19%)*	22	*(22%)*
Distribution of Informative Amt	0.02	0.01	0	*(0%)*	1	*(1%)*
SCS	0.02	0.05	0	*(0%)*	0	*(0%)*
AvICTF	0.02	0.08	0	*(0%)*	0	*(0%)*
NSCQ	0.18	0.27	8	*(8%)*	11	*(11%)*
Doddle	0.28	0.34	19	*(19%)*	25	*(25%)*
SBR	*0.22*	*0.22*	*19*	*(19%)*	*19*	*(19%)*

D.8 FR-DOE-81COL

Table D.161: FR-DOE-81COL average Spearman rank correlations, for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		**SBR**	
Algorithms	Q_s	Q_l	Q_s	Q_l
Doddle	**0.49**	**0.54**	0.05	0.05
Doddle_RC	**0.54**	**0.61**	**0.20**	**0.19**
Doddle_RP	**0.49**	**0.54**	0.00	−0.01
Doddle_RF	**0.24**	**0.35**	−0.04	−0.02
Doddle_RC+RP	**0.53**	**0.58**	0.09	0.08
Doddle_RC+RF	**0.46**	**0.53**	0.09	0.08
Doddle_RP+RF	**0.44**	**0.49**	−0.01	−0.01
Doddle_×	**0.45**	**0.44**	0.04	−0.03
Doddle_RC×RP	**0.49**	**0.51**	0.08	0.03
Doddle_RC×RF	**0.46**	**0.49**	0.10	0.04
Doddle_RP×RF	**0.42**	**0.44**	−0.04	−0.07
Doddle_W	**0.51**	**0.56**	0.07	0.06
SBR	*0.13*	*0.13*	—	—

Table D.162: FR-DOE-81COL average Spearman rank correlations, for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR and SBR.

	FsBR		SBR	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.57**	**0.45**	**0.25**	0.12
CORI	**0.63**	**0.61**	**0.31**	**0.29**
Cosine Measure	**0.39**	**0.63**	0.15	**0.27**
Inner Product	**0.59**	**0.63**	**0.45**	**0.43**
Skew	**0.55**	**0.57**	**0.48**	**0.47**
Highest-available Similarity	**0.51**	**0.57**	**0.42**	**0.44**
CVV	**0.48**	**0.51**	**0.52**	**0.54**
DFPROP	**0.53**	**0.56**	**0.51**	**0.51**
Distribution of Informative Amt	−0.44	−0.43	−0.27	−0.19
SCS	−0.43	−0.28	−0.10	0.04
AvICTF	−0.42	−0.13	−0.10	0.12
NSCQ	0.03	**0.32**	**0.45**	**0.54**
Doddle	**0.49**	**0.54**	0.05	0.05
SBR	*0.13*	*0.13*	—	—

Table D.163: FR-DOE-81COL Z-Test results, showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Short queries.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AviCTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
bGlOSS																									
CORI			✓																						
Cosine Measure		✓																							
Inner Product																									
Skew																									
Highest-available Similarity																									
CVV																									
DFPROP																									
Distribution of Informative Amt																									
SCS																									
AviCTF																									
NSCQ																									
Doddle																									
Doddle_RC																									
Doddle_RP																									
Doddle_RF																									
Doddle_RC+RP																									
Doddle_RC+RF																									
Doddle_RP+RF																									
Doddle_×																									
Doddle_RC×RP																									
Doddle_RC×RF																									
Doddle_RP×RF																									
Doddle_W																									
SBR																									

Table D.164: FR-DOE-81COL Z-Test results, showing whether the differences between Spearman correlations are significant (a ✓ shows there is significant difference in performance), executed over Long queries.

Algorithms	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-available Similarity	CVV	DFPROP	Distribution of Informative Amt	SCS	AviCTF	NSCQ	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	SBR
bGlOSS																									
CORI																✓									✓
Cosine Measure																✓									✓
Inner Product																									✓
Skew																✓									✓
Highest-available Similarity																									✓
CVV																									✓
DFPROP																									✓
Distribution of Informative Amt											✓														
SCS												✓	✓	✓											✓
AviCTF										✓		✓	✓	✓											✓
NSCQ		✓	✓	✓	✓	✓				✓			✓	✓											✓
Doddle										✓		✓		✓											✓
Doddle_RC										✓		✓	✓			✓									✓
Doddle_RP										✓		✓	✓												✓
Doddle_RF		✓	✓	✓						✓		✓	✓	✓											✓
Doddle_RC+RP										✓		✓	✓												✓
Doddle_RC+RF										✓		✓	✓												✓
Doddle_RP+RF										✓		✓	✓												✓
Doddle_×										✓		✓	✓												✓
Doddle_RC×RP										✓		✓	✓												✓
Doddle_RC×RF										✓		✓	✓												✓
Doddle_RP×RF										✓		✓	✓												✓
Doddle_W										✓		✓	✓												✓
SBR	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓		✓	✓	✓									

Table D.165: FR-DOE-81COL average Blest and Da Costa weighted rank correlations, for the various configurations of the Doddle algorithm, comparing algorithm-produced rankings to FsBR.

	Blest		**Da Costa**	
Algorithms	Q_s	Q_l	Q_s	Q_l
Doddle	**0.47**	**0.51**	**0.50**	**0.54**
Doddle_RC	**0.51**	**0.58**	**0.55**	**0.62**
Doddle_RP	**0.47**	**0.51**	**0.50**	**0.55**
Doddle_RF	**0.25**	**0.35**	**0.23**	**0.34**
Doddle_RC+RP	**0.51**	**0.55**	**0.54**	**0.59**
Doddle_RC+RF	**0.45**	**0.50**	**0.47**	**0.53**
Doddle_RP+RF	**0.43**	**0.47**	**0.45**	**0.49**
Doddle_×	**0.44**	**0.43**	**0.46**	**0.45**
Doddle_RC×RP	**0.48**	**0.49**	**0.50**	**0.52**
Doddle_RC×RF	**0.45**	**0.47**	**0.48**	**0.49**
Doddle_RP×RF	**0.41**	**0.43**	**0.43**	**0.44**
Doddle_W	**0.49**	**0.53**	**0.52**	**0.57**
SBR	*0.11*	*0.11*	*0.11*	*0.11*

Table D.166: FR-DOE-81COL average Blest and Da Costa weighted rank correlations, for existing algorithms and Doddle, comparing algorithm-produced rankings to FsBR.

	Blest		**Da Costa**	
Algorithms	Q_s	Q_l	Q_s	Q_l
bGlOSS	**0.52**	**0.20**	**0.60**	**0.56**
CORI	**0.57**	**0.57**	**0.63**	**0.61**
Cosine Measure	**0.36**	**0.60**	**0.39**	**0.63**
Inner Product	**0.55**	**0.58**	**0.59**	**0.63**
Skew	**0.51**	**0.54**	**0.55**	**0.57**
Highest-available Similarity	**0.45**	**0.51**	**0.50**	**0.56**
CVV	**0.44**	**0.47**	**0.47**	**0.51**
DFPROP	**0.50**	**0.53**	**0.53**	**0.56**
Distribution of Informative Amt	−0.41	−0.40	−0.39	−0.40
SCS	−0.40	−0.26	−0.42	−0.29
AvICTF	−0.40	−0.13	−0.41	−0.14
NSCQ	0.00	**0.28**	0.01	**0.29**
Doddle	**0.47**	**0.51**	**0.50**	**0.54**
SBR	*0.11*	*0.11*	*0.11*	*0.11*

Table D.167: FR-DOE-81COL average $\mathcal{R}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.23	0.34	0.29	0.04	0.34	0.21	0.19	0.26	0.30	0.22	0.17	0.28	*0.03*
10	0.50	0.54	0.50	0.28	0.54	0.47	0.43	0.49	0.52	0.49	0.41	0.52	*0.18*
20	0.64	0.65	0.64	0.45	0.67	0.62	0.61	0.64	0.65	0.61	0.61	0.65	*0.29*
30	0.75	0.74	0.75	0.59	0.77	0.72	0.73	0.73	0.74	0.72	0.73	0.75	*0.45*
40	0.82	0.83	0.83	0.69	0.84	0.80	0.81	0.81	0.82	0.81	0.81	0.83	*0.59*
50	0.89	0.89	0.89	0.79	0.90	0.87	0.88	0.87	0.88	0.88	0.88	0.89	*0.68*
60	0.93	0.93	0.94	0.88	0.94	0.93	0.93	0.92	0.93	0.93	0.93	0.94	*0.78*
70	0.97	0.96	0.97	0.95	0.97	0.97	0.97	0.96	0.96	0.96	0.96	0.97	*0.91*
80	1.00	0.99	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.99	1.00	1.00	*0.98*
83	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.168: FR-DOE-81COL average $\mathcal{R}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.33	0.34	0.20	0.18	0.12	0.06	0.07	0.09	0.03	0.02	0.02	0.03	0.23	*0.03*
10	0.55	0.56	0.38	0.50	0.49	0.41	0.43	0.48	0.07	0.06	0.06	0.18	0.50	*0.18*
20	0.65	0.67	0.50	0.65	0.64	0.58	0.57	0.64	0.12	0.10	0.10	0.26	0.64	*0.29*
30	0.75	0.78	0.61	0.76	0.75	0.70	0.69	0.73	0.16	0.15	0.15	0.35	0.75	*0.45*
40	0.84	0.86	0.71	0.85	0.82	0.79	0.79	0.82	0.25	0.23	0.23	0.46	0.82	*0.59*
50	0.89	0.91	0.80	0.90	0.88	0.88	0.86	0.87	0.37	0.36	0.36	0.58	0.89	*0.68*
60	0.93	0.94	0.88	0.93	0.92	0.92	0.90	0.91	0.52	0.53	0.54	0.71	0.93	*0.78*
70	0.97	0.96	0.95	0.96	0.96	0.95	0.95	0.95	0.72	0.74	0.74	0.87	0.97	*0.91*
80	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.94	0.95	0.95	0.98	1.00	*0.98*
83	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.169: FR-DOE-81COL average $\mathcal{R}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.26	0.32	0.29	0.12	0.33	0.23	0.20	0.19	0.26	0.18	0.17	0.30	*0.03*
10	0.53	0.59	0.56	0.29	0.60	0.50	0.46	0.43	0.53	0.46	0.40	0.56	*0.18*
20	0.65	0.69	0.67	0.50	0.69	0.64	0.62	0.59	0.65	0.61	0.59	0.67	*0.29*
30	0.76	0.79	0.76	0.65	0.78	0.75	0.74	0.70	0.74	0.73	0.72	0.77	*0.45*
40	0.84	0.86	0.84	0.77	0.86	0.83	0.83	0.81	0.83	0.82	0.81	0.85	*0.59*
50	0.91	0.93	0.91	0.86	0.92	0.91	0.90	0.89	0.90	0.90	0.89	0.92	*0.68*
60	0.96	0.96	0.96	0.92	0.96	0.96	0.95	0.94	0.95	0.95	0.95	0.96	*0.78*
70	0.98	0.98	0.97	0.97	0.98	0.98	0.97	0.97	0.97	0.98	0.97	0.98	*0.91*
80	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.98*
83	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.170: FR-DOE-81COL average $\mathcal{R}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.32	0.36	0.34	0.23	0.15	0.12	0.06	0.14	0.03	0.03	0.04	0.03	0.26	*0.03*
10	0.41	0.54	0.55	0.54	0.49	0.46	0.43	0.50	0.08	0.07	0.10	0.27	0.53	*0.18*
20	0.46	0.66	0.67	0.66	0.63	0.62	0.59	0.64	0.13	0.13	0.17	0.40	0.65	*0.29*
30	0.53	0.76	0.78	0.77	0.74	0.74	0.70	0.74	0.19	0.20	0.28	0.52	0.76	*0.45*
40	0.60	0.84	0.86	0.85	0.83	0.83	0.80	0.83	0.28	0.31	0.38	0.64	0.84	*0.59*
50	0.68	0.91	0.92	0.92	0.90	0.90	0.88	0.89	0.39	0.44	0.52	0.76	0.91	*0.68*
60	0.77	0.95	0.95	0.95	0.94	0.94	0.92	0.93	0.53	0.60	0.66	0.84	0.96	*0.78*
70	0.88	0.98	0.98	0.98	0.97	0.96	0.96	0.97	0.70	0.78	0.82	0.93	0.98	*0.91*
80	0.97	0.99	1.00	1.00	1.00	0.99	0.99	0.99	0.93	0.95	0.97	0.99	1.00	*0.98*
83	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.171: FR-DOE-81COL average $\hat{\mathcal{R}}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.02	0.03	0.03	0.00	0.03	0.02	0.02	0.03	0.03	0.02	0.02	0.03	*0.00*
10	0.27	0.29	0.26	0.15	0.29	0.26	0.23	0.26	0.28	0.27	0.22	0.28	*0.10*
20	0.49	0.50	0.49	0.35	0.51	0.48	0.47	0.49	0.50	0.48	0.47	0.50	*0.23*
30	0.67	0.67	0.67	0.53	0.69	0.65	0.65	0.66	0.66	0.65	0.65	0.68	*0.41*
40	0.79	0.80	0.80	0.67	0.81	0.77	0.78	0.78	0.79	0.78	0.78	0.80	*0.57*
50	0.88	0.88	0.88	0.79	0.89	0.86	0.87	0.86	0.87	0.87	0.87	0.88	*0.68*
60	0.93	0.93	0.93	0.87	0.94	0.92	0.93	0.92	0.93	0.93	0.93	0.93	*0.78*
70	0.97	0.96	0.97	0.95	0.97	0.96	0.97	0.96	0.96	0.96	0.96	0.97	*0.91*
80	1.00	0.99	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.99	1.00	1.00	*0.98*
83	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.172: FR-DOE-81COL average $\hat{\mathcal{R}}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.03	0.04	0.02	0.02	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.02	*0.00*
10	0.29	0.30	0.20	0.27	0.26	0.22	0.23	0.26	0.04	0.03	0.03	0.10	0.27	*0.10*
20	0.50	0.52	0.38	0.50	0.50	0.45	0.44	0.49	0.08	0.07	0.07	0.20	0.49	*0.23*
30	0.68	0.70	0.54	0.68	0.67	0.63	0.62	0.66	0.14	0.13	0.13	0.31	0.67	*0.41*
40	0.80	0.83	0.68	0.81	0.79	0.76	0.76	0.79	0.23	0.21	0.21	0.44	0.79	*0.57*
50	0.88	0.90	0.79	0.89	0.87	0.87	0.85	0.87	0.36	0.35	0.36	0.58	0.88	*0.68*
60	0.92	0.93	0.88	0.93	0.92	0.92	0.90	0.91	0.52	0.53	0.53	0.71	0.93	*0.78*
70	0.97	0.96	0.95	0.96	0.96	0.95	0.95	0.95	0.72	0.74	0.74	0.87	0.97	*0.91*
80	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.94	0.95	0.95	0.98	1.00	*0.98*
83	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.173: FR-DOE-81COL average $\hat{\mathcal{R}}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.03	0.03	0.03	0.01	0.04	0.02	0.02	0.02	0.03	0.02	0.02	0.03	*0.00*
10	0.29	0.32	0.30	0.16	0.32	0.27	0.25	0.23	0.29	0.25	0.22	0.30	*0.10*
20	0.50	0.53	0.52	0.39	0.53	0.49	0.48	0.46	0.50	0.47	0.45	0.52	*0.23*
30	0.68	0.70	0.68	0.59	0.70	0.67	0.66	0.63	0.66	0.66	0.64	0.69	*0.41*
40	0.81	0.83	0.81	0.74	0.83	0.80	0.80	0.78	0.80	0.79	0.78	0.82	*0.57*
50	0.90	0.92	0.90	0.85	0.92	0.90	0.89	0.88	0.89	0.89	0.88	0.91	*0.68*
60	0.96	0.96	0.95	0.92	0.96	0.95	0.95	0.94	0.94	0.95	0.95	0.96	*0.78*
70	0.98	0.98	0.97	0.97	0.98	0.98	0.97	0.97	0.97	0.98	0.97	0.98	*0.91*
80	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.98*
83	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.174: FR-DOE-81COL average $\hat{\mathcal{R}}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.03	0.04	0.03	0.02	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.03	*0.00*
10	0.21	0.29	0.30	0.29	0.26	0.25	0.23	0.27	0.04	0.03	0.06	0.14	0.29	*0.10*
20	0.35	0.51	0.51	0.51	0.49	0.48	0.45	0.50	0.09	0.09	0.13	0.31	0.50	*0.23*
30	0.47	0.68	0.70	0.69	0.66	0.66	0.63	0.66	0.17	0.17	0.25	0.46	0.68	*0.41*
40	0.57	0.81	0.83	0.82	0.80	0.80	0.77	0.80	0.27	0.29	0.37	0.61	0.81	*0.57*
50	0.67	0.90	0.91	0.91	0.89	0.89	0.87	0.88	0.38	0.44	0.51	0.75	0.90	*0.68*
60	0.76	0.95	0.95	0.95	0.94	0.94	0.92	0.93	0.53	0.60	0.66	0.84	0.96	*0.78*
70	0.88	0.98	0.98	0.98	0.97	0.96	0.96	0.97	0.70	0.78	0.82	0.93	0.98	*0.91*
80	0.97	0.99	1.00	1.00	1.00	0.99	0.99	0.99	0.93	0.95	0.97	0.99	1.00	*0.98*
83	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*1.00*

Table D.175: FR-DOE-81COL average $\mathcal{P}_n$ scores, for the various configurations of the Doddle algorithm, on short queries (Q_s).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.58	0.78	0.62	0.27	0.71	0.55	0.49	0.61	0.66	0.62	0.45	0.66	*0.42*
10	0.72	0.82	0.70	0.46	0.77	0.71	0.60	0.69	0.75	0.72	0.56	0.74	*0.59*
20	0.77	0.82	0.78	0.58	0.81	0.76	0.71	0.75	0.78	0.76	0.70	0.79	*0.64*
30	0.78	0.81	0.78	0.64	0.81	0.76	0.75	0.75	0.77	0.76	0.73	0.79	*0.68*
40	0.76	0.79	0.76	0.66	0.78	0.75	0.74	0.74	0.76	0.75	0.74	0.77	*0.67*
50	0.73	0.76	0.74	0.66	0.75	0.73	0.72	0.72	0.73	0.73	0.72	0.74	*0.64*
60	0.69	0.70	0.69	0.65	0.69	0.68	0.68	0.68	0.68	0.68	0.68	0.69	*0.62*
70	0.64	0.65	0.64	0.63	0.64	0.64	0.64	0.63	0.64	0.64	0.63	0.64	*0.63*
80	0.59	0.60	0.60	0.60	0.59	0.59	0.60	0.60	0.59	0.59	0.60	0.59	*0.60*
83	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	*0.58*

Table D.176: FR-DOE-81COL average $\mathcal{P}_n$ scores, for existing algorithms and Doddle, on short queries (Q_s).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.80	0.91	0.62	0.69	0.61	0.38	0.55	0.58	0.17	0.16	0.16	0.43	0.58	*0.42*
10	0.84	0.89	0.68	0.86	0.84	0.73	0.81	0.84	0.21	0.22	0.22	0.54	0.72	*0.59*
20	0.84	0.88	0.71	0.87	0.85	0.80	0.81	0.85	0.28	0.28	0.28	0.56	0.77	*0.64*
30	0.82	0.86	0.71	0.85	0.84	0.80	0.80	0.83	0.31	0.33	0.33	0.58	0.78	*0.68*
40	0.80	0.84	0.72	0.83	0.81	0.79	0.79	0.81	0.39	0.41	0.41	0.61	0.76	*0.67*
50	0.76	0.80	0.71	0.78	0.77	0.77	0.76	0.77	0.45	0.48	0.48	0.63	0.73	*0.64*
60	0.70	0.72	0.68	0.72	0.71	0.72	0.70	0.70	0.50	0.53	0.54	0.63	0.69	*0.62*
70	0.64	0.66	0.64	0.66	0.65	0.66	0.65	0.65	0.55	0.57	0.57	0.63	0.64	*0.63*
80	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.58	0.58	0.58	0.60	0.59	*0.60*
83	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	*0.58*

Table D.177: FR-DOE-81COL average $\mathcal{P}_n$ scores, for the various configurations of the Doddle algorithm, on long queries (Q_l).

	Algorithms												
n	Doddle	Doddle_RC	Doddle_RP	Doddle_RF	Doddle_RC+RP	Doddle_RC+RF	Doddle_RP+RF	Doddle_×	Doddle_RC×RP	Doddle_RC×RF	Doddle_RP×RF	Doddle_W	*SBR*
1	0.68	0.85	0.67	0.46	0.79	0.67	0.58	0.63	0.70	0.60	0.52	0.74	*0.42*
10	0.74	0.85	0.75	0.48	0.82	0.73	0.64	0.63	0.75	0.69	0.57	0.79	*0.59*
20	0.78	0.85	0.79	0.64	0.83	0.77	0.73	0.71	0.77	0.74	0.69	0.80	*0.64*
30	0.78	0.83	0.79	0.69	0.81	0.78	0.75	0.72	0.77	0.76	0.72	0.80	*0.68*
40	0.78	0.81	0.78	0.71	0.80	0.77	0.76	0.73	0.76	0.75	0.73	0.79	*0.67*
50	0.76	0.79	0.75	0.70	0.77	0.76	0.74	0.72	0.74	0.75	0.73	0.76	*0.64*
60	0.71	0.72	0.70	0.68	0.71	0.71	0.70	0.69	0.69	0.70	0.69	0.71	*0.62*
70	0.65	0.65	0.64	0.64	0.65	0.65	0.64	0.64	0.64	0.64	0.64	0.65	*0.63*
80	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	*0.60*
83	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	*0.58*

Table D.178: FR-DOE-81COL average $\mathcal{P}_n$ scores, for existing algorithms and Doddle, on long queries (Q_l).

	Algorithms													
n	bGlOSS	CORI	Cosine Measure	Inner Product	Skew	Highest-avail. Sim.	CVV	DFPROP	Dist. of Inf. Amt	SCS	AvICTF	NSCQ	Doddle	*SBR*
1	0.73	0.89	0.90	0.76	0.68	0.52	0.55	0.64	0.19	0.30	0.36	0.44	0.68	*0.42*
10	0.65	0.88	0.88	0.89	0.83	0.76	0.82	0.83	0.25	0.30	0.37	0.70	0.74	*0.59*
20	0.64	0.86	0.86	0.88	0.85	0.82	0.83	0.86	0.31	0.36	0.44	0.73	0.78	*0.64*
30	0.60	0.85	0.85	0.86	0.83	0.82	0.82	0.84	0.36	0.42	0.51	0.74	0.78	*0.68*
40	0.60	0.83	0.83	0.84	0.82	0.82	0.80	0.82	0.42	0.49	0.55	0.74	0.78	*0.67*
50	0.61	0.79	0.80	0.81	0.79	0.79	0.77	0.78	0.46	0.53	0.58	0.73	0.76	*0.64*
60	0.60	0.73	0.73	0.73	0.72	0.73	0.70	0.72	0.50	0.57	0.60	0.69	0.71	*0.62*
70	0.59	0.67	0.66	0.66	0.66	0.66	0.65	0.66	0.54	0.59	0.61	0.65	0.65	*0.63*
80	0.58	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.57	0.59	0.59	0.60	0.60	*0.60*
83	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	*0.58*

Table D.179: FR-DOE-81COL average precision within the top 5 collections, and the number of queries (out of 100) for which the first ranked collection is correct, for the various configurations of the Doddle algorithm.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Q_s	Q_l	Q_s		Q_l	
Doddle	0.21	0.26	6	*(6%)*	7	*(7%)*
Doddle_RC	0.24	0.29	7	*(7%)*	9	*(9%)*
Doddle_RP	0.20	0.26	12	*(12%)*	9	*(9%)*
Doddle_RF	0.07	0.10	0	*(0%)*	3	*(3%)*
Doddle_RC+RP	0.23	0.29	10	*(10%)*	9	*(9%)*
Doddle_RC+RF	0.21	0.24	4	*(4%)*	6	*(6%)*
Doddle_RP+RF	0.16	0.20	6	*(6%)*	4	*(4%)*
Doddle_×	0.18	0.17	8	*(8%)*	2	*(2%)*
Doddle_RC×RP	0.22	0.22	11	*(11%)*	5	*(5%)*
Doddle_RC×RF	0.20	0.20	4	*(4%)*	3	*(3%)*
Doddle_RP×RF	0.14	0.17	4	*(4%)*	2	*(2%)*
Doddle_W	0.22	0.28	8	*(8%)*	9	*(9%)*
SBR	*0.01*	*0.01*	*0*	*(0%)*	*0*	*(0%)*

Table D.180: FR-DOE-81COL average precision within the top 5 collections, and the number of queries (out of 100) for which the first ranked collection is correct, for existing algorithms and Doddle.

	Precision @ 5		**Correct @ Pos. 1**			
Algorithms	Q_s	Q_l	Q_s		Q_l	
bGlOSS	0.25	0.18	12	*(12%)*	9	*(9%)*
CORI	0.24	0.22	10	*(10%)*	8	*(8%)*
Cosine Measure	0.14	0.26	3	*(3%)*	9	*(9%)*
Inner Product	0.19	0.23	8	*(8%)*	5	*(5%)*
Skew	0.17	0.19	4	*(4%)*	2	*(2%)*
Highest-available Similarity	0.13	0.18	0	*(0%)*	2	*(2%)*
CVV	0.13	0.14	1	*(1%)*	0	*(0%)*
DFPROP	0.16	0.20	2	*(2%)*	3	*(3%)*
Distribution of Informative Amt	0.02	0.02	0	*(0%)*	0	*(0%)*
SCS	0.02	0.02	0	*(0%)*	0	*(0%)*
AvICTF	0.01	0.02	0	*(0%)*	0	*(0%)*
NSCQ	0.04	0.05	0	*(0%)*	0	*(0%)*
Doddle	0.21	0.26	6	*(6%)*	7	*(7%)*
SBR	*0.01*	*0.01*	*0*	*(0%)*	*0*	*(0%)*

Bibliography

[1] American National Standards Institute. *Information Retrieval (Z39.50): Application Service Definition and Protocol Specification*, 2003.

[2] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, 1999.

[3] David C. Blest. Rank correlation – an alternative measure. *Australian & New Zealand Journal of Statistics*, 42(1):101–111, 2000.

[4] George Buchanan, Sally Jo Cunningham, Ann Blandford, Jon Rimmer, and Claire Warwick. Information seeking by humanities scholars. In *Proceedings of the 9th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, volume 3652 of *LNCS*, pages 218–229. Springer, 2005.

[5] Chris Buckley and Ellen M. Voorhees. Retrieval system evaluation. In *TREC: Experiment and Evaluation in Information Retrieval*, chapter 3, pages 53–75. MIT Press, 2005.

[6] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching distributed collections with inference networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28. ACM, 1995.

[7] C.W. Cleverdon. The Cranfield tests on index language devices. In Karen Sparck Jones and Peter Willett, editors, *Readings in Information Retrieval*, pages 47–59. Morgan Kaufmann Publishers Inc., 1997.

[8] Joaquim Pinto Da Costa and Carlos Soares. A weighted rank measure of correlation. *Australian & New Zealand Journal of Statistics*, 47(4):515–529, 2005.

[9] Helen Dodd, George Buchanan, and Matt Jones. A new perspective on collection selection. In *Proceedings of the 14th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, volume 6273 of *LNCS*, pages 433–436. Springer, 2010.

[10] Daryl D'Souza, James A. Thom, and Justin Zobel. Collection selection for managed distributed document databases. *Information Processing and Management*, 40(3):527–546, 2004.

[11] Daryl J. D'Souza and James A. Thom. Collection selection using n-term indexing. In *Proceedings of the Second International Symposium on Cooperative Database Systems for Advanced Applications (CODAS)*, pages 52–63, 1999.

[12] Daryl J. D'Souza, James A. Thom, and Justin Zobel. A comparison of techniques for selecting text collections. In *Proceedings of the 2000 Australasian Database Conference (ADC)*, 2000.

[13] Daryl J. D'Souza, Justin Zobel, and James A. Thom. Is CORI effective for collection selection? An exploration of parameters, queries, and data. In *Proceedings of the Ninth Australasian Document Computing Symposium (ADCS)*, pages 41–46, 2004.

[14] David Ellis. A behavioural approach to information retrieval system design. *The Journal of Documentation*, 45(3):171–212, 1989.

[15] The Apache Software Foundation. Lucene search algorithm. Available at: `http://lucene.apache.org/core/old_versioned_docs/versions/3_0_0/api/core/org/apache/lucene/search/Similarity.html`, November 2009.

[16] James C. French and Allison L. Powell. Metrics for evaluating database selection techniques. *World Wide Web*, 3(3):153–163, 2000.

[17] James C. French, Allison L. Powell, Jamie Callan, Charles L. Viles, Travis Emmitt, Kevin J. Prey, and Yun Mou. Comparing the performance of database selection algorithms. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 238–245. ACM, 1999.

[18] James C. French, Allison L. Powell, Charles L. Viles, Travis Emmitt, and Kevin J. Prey. Evaluating database selection techniques: a testbed and experiment. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 121–129. ACM, 1998.

[19] Norbert Fuhr. Optimum database selection in networked IR. In *Proceedings of the SIGIR'96 Workshop on Networked Information Retrieval*, volume 7 of *CEUR Workshop Proceedings*. CEUR-WS.org, 1996.

[20] Norbert Fuhr. A decision-theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems*, 17(3):229–249, 1999.

[21] Norbert Fuhr. Resource discovery in distributed digital libraries. In *Proceedings of the 1999 Russian Conference on Digital Libraries (RCDL)*, pages 35–45. St. Petersburg State University, 1999.

[22] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. The effectiveness of GlOSS for the text database discovery problem. In *Proceedings of the 1994 ACM SIGMOD Conference on Management of Data*, pages 126–137. ACM, 1994.

[23] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. GlOSS: text-source discovery over the internet. *ACM Transactions on Database Systems*, 24(2):229–264, 1999.

[24] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *Proceedings of the 14th International World Wide Web Conferences (WWW)*, pages 902–903, 2005.

[25] Donna K. Harman. The TREC test collections. In Ellen M. Voorhees and Donna K. Harman, editors, *TREC: Experiment and Evaluation in Information Retrieval*, chapter 2, pages 21–52. MIT Press, 2005.

[26] Claudia Hauff. *Predicting the Effectiveness of Queries and Retrieval Systems.* PhD thesis, University of Twente, 2007.

[27] Claudia Hauff, Djoerd Hiemstra, and Franciska de Jong. A survey of pre-retrieval query performance predictors. In *Proceedings of the 2008 ACM Conference on Information and Knowledge Management (CIKM)*, pages 1419–1420. ACM, 2008.

[28] Ben He and Iadh Ounis. Query performance prediction. *Information Systems*, 31(7):585–594, 2006.

[29] Bin He, Mitesh Patel, Zhen Zhang, and Kevin Chen-Chuan Chang. Accessing the deep web. *Communications of the ACM*, 50(5):94–101, 2007.

[30] David Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 329–338. ACM, 1993.

[31] Dublin Core Metadata Initiative. Dublin core metadata element set, version 1.1. Available at: `http://dublincore.org/documents/2012/06/14/dces/`, June 2012.

[32] Gopal K. Kanji. *100 Statistical Tests.* SAGE Publications, 1993.

[33] I. E. Kuralenok and I. S. Nekrestyanov. Evaluation of text retrieval systems. *Programming and Computer Software*, 28(4):226–242, 2002.

[34] K. L. Kwok. A new method of weighting query terms for ad-hoc retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 187–195. ACM, 1996.

[35] Carl Lagoze, Herbert Van de Sompel, Michael Nelson, and Simeon Warner, editors. *The Open Archives Initiative Protocol for Metadata Harvesting v2.0.* Open Archives Initiative, 2008. Available at `http://www.openarchives.org/OAI/openarchivesprotocol.html`.

[36] Steve Lawrence and C. Lee Giles. Accessibility of information on the web. *Nature*, 400:107–109, July 1999.

[37] John Parry Lewis and Alasdair Traill. *Statistics Explained.* Addison Wesley, 1999.

[38] Weiyi Meng, Clement Yu, and King-Lup Liu. Building efficient and effective metasearch engines. *ACM Computing Surveys*, 34(1):48–89, 2002.

[39] Henrik Nottelmann and Norbert Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 290–297. ACM, 2003.

[40] Henrik Nottelmann and Norbert Fuhr. Combining CORI and the decision-theoretic approach for advanced resource selection. In *Proceedings of the 26th European Conference on IR Research (ECIR)*, volume 2997 of *LNCS*, pages 138–153. Springer, 2004.

[41] Henrik Nottelmann and Norbert Fuhr. A decision-theoretic model for decentralised query routing in hierarchical peer-to-peer networks. In *Proceedings of the 29th European Conference on IR Research (ECIR)*, volume 4425 of *LNCS*, pages 148–159. Springer, 2007.

[42] Joaquín Pérez-Iglesias, José R. Pérez-Agüera, Víctor Fresno, and Yuval Z. Feinstein. Integrating the probabilistic models BM25/BM25F into Lucene. *Computing Research Repository*, 2009.

[43] Allison L. Powell. *Database Selection in Distributed Information Retrieval: A Study of Multi-Collection Information Retrieval.* PhD thesis, School of Engineering and Applied Science, University of Virginia, 2001.

[44] Allison L. Powell and James C. French. Comparing the performance of collection selection algorithms. *ACM Transactions on Information Systems*, 21(4):412–456, 2003.

[45] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[46] Jangwon Seo and W. Bruce Croft. Blog site search using resource selection. In *Proceedings of the 2008 ACM Conference on Information and Knowledge Management (CIKM)*, pages 1053–1062. ACM, 2008.

[47] Milad Shokouhi. Central-rank-based collection selection in uncooperative distributed information retrieval. In *Proceedings of the 29th European Conference on IR Research (ECIR)*, volume 4425 of *LNCS*, pages 160–172. Springer, 2007.

[48] Milad Shokouhi and Luo Si. Federated search. *Foundations and Trends in Information Retrieval*, 5(1):1–102, 2011.

[49] Luo Si and Jamie Callan. Relevant document distribution estimation method for resource selection. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 298–305. ACM, 2003.

[50] Luo Si and Jamie Callan. Unified utility maximization framework for resource selection. In *Proceedings of the 2004 ACM Conference on Information and Knowledge Management (CIKM)*, pages 32–41. ACM, 2004.

[51] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[52] Mikhail Sogrine and Ahmed Patel. Evaluating database selection algorithms for distributed search. In *Proceedings of the 2003 ACM Symposium on Applied Computing (SAC)*, pages 817–822, 2003.

[53] Yang Song, Nam Nguyen, and Li wei He. Searchable web sites recommendations. In *Proceedings of the Fourth International Conference on Web Search and Data Mining (WSDM)*, pages 405–414. ACM, 2011.

[54] Rashmi Srinivasa, Tram Phan, Nisanti Mohanraj, Allison L. Powell, and Jim French. Database selection using document and collection term frequencies. Technical Report CS-2000-32, University of Virginia, May 2000.

[55] Jean Tague-Sutcliffe and James Blustein. A statistical analysis of the TREC-3 data. In *Overview of the Third Text REtrieval Conference TREC-3*, pages 385–398, 1994.

[56] Paul Thomas and David Hawking. Server selection methods in personal metasearch: a comparative empirical study. *Information Retrieval*, 12(5):581–604, 2009.

[57] Paul Thomas and Milad Shokouhi. SUSHI: Scoring scaled samples for server selection. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 419–426. ACM, 2009.

[58] C. J. van Rijsbergen. *Information Retrieval.* Butterworths, second edition, 1979.

[59] Ellen M. Voorhees and Donna K. Harman. The text REtrieval conference. In Ellen M. Voorhees and Donna K. Harman, editors, *TREC: Experiment and Evaluation in Information Retrieval*, chapter 1, pages 3–19. MIT Press, 2005.

[60] Iris Xie and Colleen Cool. Understanding help seeking within the context of searching digital libraries. *Journal of the American Society for Information Science and Technology*, 60(3):477–494, 2009.

[61] Elad Yom-Tov, Shai Fine, David Carmel, and Adam Darlow. Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 512–519. ACM, 2005.

[62] Elad Yom-Tov, Shai Fine, David Carmel, and Adam Darlow. Metasearch and federation using query difficulty prediction. In *ACM SIGIR 2005 Workshop: Predicting Query Difficulty - Methods and Applications*. ACM, 2005.

[63] Budi Yuwono and Dik L. Lee. Server ranking for distributed text retrieval systems on the internet. In *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 41–50. World Scientific Press, 1997.

[64] Ying Zhao, Falk Scholer, and Yohannes Tsegay. Effective pre-retrieval query performance prediction using similarity and variability evidence. In *Proceedings of the 30th European Conference on IR Research (ECIR)*, volume 4956 of *LNCS*, pages 52–64. Springer, 2008.

[65] Yun Zhou and W. Bruce Croft. Query performance prediction in web search environments. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 543–550. ACM, 2007.

[66] Justin Zobel. Collection selection via lexicon inspection. In *Proceedings of the Second Australasian Document Computing Symposium (ADCS)*, pages 74–80, 1997.