POLIS: a probabilistic summarisation logic for structured documents

Forst, Jan Frederik

POLIS: A Probabilistic Summarisation Logic for Structured Documents

Jan Frederik Forst

University of London

Thesis submitted for the degree of Doctor of Philosophy

at Queen Mary, University of London

June 2009

Declaration of originality

I hereby declare that this thesis, and the research to which it refers, are the product of my own work, and that any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

The material contained in this thesis has not been submitted, either in whole or in part, for a degree or diploma or other qualification at the University of London or any other University.

Some parts of this work have been previously published as:

- Jan Frederik Forst, *A Summarisation Logic for Structured Documents*, SIGIR 2007 Doctoral Consortium (Forst (2007))

- Jan Frederik Forst, Thomas Roelleke, and Anastasios Tombros, *Modelling a Summarisation Logic in Probabilistic Datalog*, LWA 2007 Workshop Information Retrieval (Forst et al. (2007a))

- Jan Frederik Forst, Anastasios Tombros, and Thomas Roelleke, *POLIS: A Probabilistic Logic for Document Summarisation*, ICTIR 2007 (Forst et al. (2007b))

- Jan Frederik Forst, Anastasios Tombros, and Thomas Roelleke, *Solving the Enterprise TREC Task with Probabilistic Data Models*, TREC 2006 (Forst et al. (2006))

Jan Frederik Forst

Abstract

As the availability of structured documents, formatted in markup languages such as SGML, RDF, or XML, increases, retrieval systems increasingly focus on the retrieval of document-elements, rather than entire documents. Additionally, abstraction layers in the form of formalised retrieval logics have allowed developers to include search facilities into numerous applications, without the need of having detailed knowledge of retrieval models.

Although automatic document summarisation has been recognised as a useful tool for reducing the workload of information system users, very few such abstraction layers have been developed for the task of automatic document summarisation. This thesis describes the development of an abstraction logic for summarisation, called POLIS, which provides users (such as developers or knowledge engineers) with a high-level access to summarisation facilities. Furthermore, POLIS allows users to exploit the hierarchical information provided by structured documents.

The development of POLIS is carried out in a step-by-step way. We start by defining a series of probabilistic summarisation models, which provide weights to document-elements at a user selected level. These summarisation models are those accessible through POLIS. The formal definition of POLIS is performed in three steps. We start by providing a syntax for POLIS, through which users/knowledge engineers interact with the logic. This is followed by a definition of the logics semantics. Finally, we provide details of an implementation of POLIS.

The final chapters of this dissertation are concerned with the evaluation of POLIS, which is conducted in two stages. Firstly, we evaluate the performance of the summarisation models by applying POLIS to two test collections, the DUC AQUAINT corpus, and the INEX IEEE corpus. This is followed by application scenarios for POLIS, in which we discuss how POLIS can be used in specific IR tasks.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank the following people:

My supervisors Thomas Rölleke, Anastasios Tombros, and Dudley Stark, who first drew my attention to the field of IR, and with their comments, instructions, and constructive feedback made it possible for me to complete this work.

The members of the QMIR research group (both past and present) for providing a stimulating research environment. Special thanks to Zoltán Szlávik, Hany Azzam, and Jun Wang, with whom I endlessly discussed the meaning of PhD, life, and the ”-hashFunction” operator in hy_mds2pri.

David Harper and Mark Sanderson for the feedback provided at the SIGIR 2007 Doctoral Consortium. Their comments helped me realise that POLIS needs to be used by actual people.

My examiners, Norbert Fuhr and Wolfgang Emmerich, for their helpful comments.

My parents for their love and support.

Eva. Without her love and support I would not have been able to complete my studies.

Chapter 1

Introduction

To mention the ever increasing amount of data, triggered by the ubiquity of computers, has become a mainstay of publications in information retrieval. Although the statement “data is ever increasing” can safely be considered a truism, the large amounts of data held on personal computers, and accessible via the internet, require advanced processing methods to not be overwhelming, but useful for searchers (Wurman et al. (2000)).

Automatic document summarisation has been recognised as a useful means to reduce the information load of information systems. Tombros and Sanderson showed that users of a retrieval system that presents query-based summaries of retrieved documents were able to judge the relevance of documents more quickly and more reliable than users of a “typical” IR systems, which only display document titles and leading text (Tombros and Sanderson (1998)).

Wolf *et al.* confirmed these findings in a later study, showing that users of a technical support system were able to determine the relevance of retrieved documents more accurately when presented with short summaries, compared to being shown only the title (Wolf et al. (2004)).

In addition to reducing the information load of users, summarisation systems have also been considered for reducing the data load of information systems. Sakai and Sparck-Jones showed that – given a summariser of sufficiently high quality – indexing summaries instead of full texts does not adversely affect the retrieval quality of an IR system (Sakai and Sparck-Jones (2001)).

1.1 Motivation

Prior research on automatic summarisation has established the usefulness of automatic summarisation methods as a means for reducing information overload. A multitude of strategies and models for the generation of summaries have been developed, using diverse methods such as term frequencies for determining the most salient sentences (e.g. Luhn (1958) and Edmundson (1969)), machine learning techniques (e.g. Kupiec et al. (1995)), functional classification of sentences (e.g. Teufel and Moens (2002) and Teufel and Moens (1998a)), or form-filling approaches (Radev and McKeown (1998); McKeown and Radev (1995)).

All of these methodologies were either developed in the context of a summarisation system, or were implemented in a summarisation system after their formal definition. This, consequently, has led to the development of a large number of automatic summarisation systems, such as SUMMARIST (Hovy and Lin (1998)), SUMMONS (Radev and McKeown (1998)), OCELOT (Berger and Mittal (2000)), SUSY (Fum et al. (1985)), TOPIC (Reimer and Hahn (1988)), or SCISORS (Rau et al. (1989)). However, these systems provide a *black box* approach to summarisation: it is not clear to users of such systems how the summaries are generated, or how (and if) the process of generating summaries could be tailored to individual needs. Interaction with such summarisation systems is usually achieved through either a Graphical User Interface (GUI), or by means of a configuration file.

However, both interaction strategies have drawbacks: the expressive power of summarisation systems controlled through GUIs is limit by screen estate, while the expressiveness of configuration files is limited by the set of options available (we discuss examples of both GUIs and configuration files in Section 3.7). A careful comparison of the expressive powers of these approaches with logical retrieval approaches further motivates the development of POLIS.

While the expressiveness of both GUIs and configuration files is limited by the predefined number of options, POLIS in comparison has a higher expressiveness, as it allows the definition of entirely new summarisation frameworks, customised to specific scenarios. POLIS is not intended to be a replacement of GUIs or configuration files, but instead provides an intermediate layer between the actual summariser implementation, and the user interaction layer. In this tiered architecture, POLIS itself could be controlled through GUIs, or could be used by other applications as a summarisation service (Figure 1.1).

Ideally, it should be possible for users of a summarisation system (the issue is whether "users" here refers to end-users of a system, or to e.g. knowledge engineers who use a summariser as a component for developing their own information system) to tailor the summarisation process to their individual needs without any knowledge of how the actual summary generation is performed. Users would issue a command to a system, detailing the textual components that should make up the summary (that is, whether it should be composed of individual sentences, or larger text fragments), query terms to be considered in the formation of the summary, and a quantifier limiting the size of the summary to be generated, which would be processed by the system to generate an appropriate summary.

Such a "summarisation command" would provide an abstraction layer from the actual summarisation system, hiding technical details such as the model used, or the actual implementation of the model, and instead provide a high level view of the summarisation process, consisting of the summarisation command as input, and the generated summary as output.

Similar abstraction layers have been proposed and developed for the information retrieval task of ad-hoc retrieval, where *abstraction logics* allow users to develop their own retrieval strategies expressed as logical statements. Meghini *et al.* proposed to model ad-hoc retrieval as a terminological logic, where a query consists of a logical expression, and all documents subsumed by the query are considered relevant (Meghini et al. (1993)). Roelleke developed an abstraction logic for the retrieval of structured documents, called *POOL*. In POOL, the syntax mirrors the actual structure of documents to be retrieved (Rölleke and Fuhr (1996); Roelleke (1999)). Furthermore, Fuhr *et al.* used POOL to develop a multimedia retrieval system (called DOLORES), highlighting how abstraction logics can be used to quickly adapt retrieval systems to new tasks (Fuhr et al. (1998)).

1.2 Summarisation Logic

The aim of the research reported here is the development of an abstraction logic for summarisation, similar to abstraction logics for ad-hoc retrieval. While such a summarisation logic will need to use some models for the generation of summaries, the development of (probabilistic) models is not the main focus of the work reported here. We will develop probabilistic models for the process of content extraction and summary generation, however, the actual models used do not determine the summarisation logic, and could be substituted by other summarisation methodologies.

Instead, the work reported here will focus on ways in which users in need of a summarisation facility could be able to express their need in an abstract way, and how such abstract summarisation expressions could be processed by an information retrieval framework.

To achieve this level of abstraction, we propose a summarisation logic with a syntax based on XPath (Clark and DeRose (1999)), with additional quantification operators inspired by terminological logics. Summarisation expressions mirror the structure of documents to be summarised, thus specifying those structural elements which should form the summary. Quantification operators limit the size of the generated summaries, limiting either the maximum total number of words in a summary, or the maximum number of textual elements (such as sentences) in a summary.

In addition, we develop a possible worlds based semantics for the summarisation logic, to give meaning to the logical expressions, and acquire a solid foundation for the probabilistic summarisation models developed.

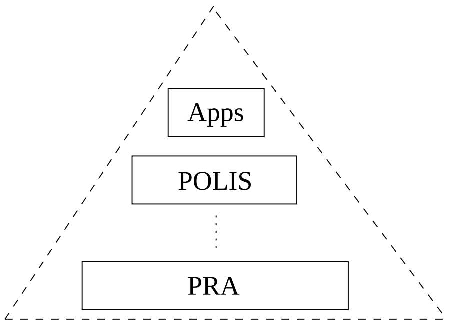


Figure 1.1: POLIS acts as an abstraction layer, providing summarisation facilities to applications (e.g. information systems).

The resulting summarisation logic, POLIS, provides an abstraction layer which allows applications and information systems to integrate summarisation facilities, without detailed knowledge of either the implementation, or the summarisation models. Figure 1.1 shows a hierarchical decomposition of these abstraction layers, where POLIS separates applications using summarisation facilities from the lower implementation layers. We will discuss the implementation of POLIS in more detail in Section 6.4.

1.3 Contributions of this Dissertation

The main contribution of this dissertation is the development of a summarisation logic, called POLIS. POLIS provides an abstraction layer to the task of summarisation, which allows it to be used as a service in a tiered architecture. At the same time, POLIS retains the high expressive power of logical retrieval approaches, allowing the definition of new summarisation methodologies.

The high-abstraction point of view of summarisation taken here also makes it possible to compare automatic summarisation to other retrieval tasks on a conceptual level. We show the similarity of summarisation to other retrieval tasks – such as anchor-text retrieval, structured document retrieval, and page partitioning – when observing them on a conceptual level.

This high-level comparison of POLIS to ad-hoc retrieval tasks additionally motivates the development of probabilistic summarisation models derived from existing retrieval approaches. These models, together with an XPath-based syntax and a possible-worlds semantics, form a main part of the specification of POLIS.

Finally, we provide two concrete example where POLIS can be used beneficially. Firstly, for the task of expert finding, and using the Enterprise TREC Track as a test collection, we show how POLIS can be used to generate expert profiles, and how these profiles can be used to find "helpful" experts for given topics. Furthermore, using sample documents from Wikipedia, we show how POLIS can be used to generate text snippets typically found on search engine result pages.

1.4 Dissertation Outline

The remaining chapters of this dissertation are organised as follows:

Chapter 2 introduces the basic notions and terminology used in information retrieval, pertinent to the research reported here. This includes concepts common to information retrieval, such as document representation, document-query matching, and the evaluation of information retrieval systems.

Chapter 3 provides an overview of past research on automatic summarisation, and the development of abstraction layers to information retrieval tasks. It looks at models for the summarisation of documents, systems that were developed in the course of this research, and the way in which these systems allowed for user control. It also provides an overview of the use of abstraction logics in information retrieval.

Chapter 4 shows how a high abstraction view of summarisation is similar to other retrieval tasks, such as anchor-text retrieval, structured document retrieval, and web retrieval based on page partitioning. All of these share the concept that information from different sources is aggregated, and the aggregated knowledge used relevance assessments.

Chapter 5 looks at probabilistic models for document summarisation. In this chapter, we aim to provide a parallelism between ad-hoc retrieval models, and their application to the task of document summarisation. Taking these modified ad-hoc models as a starting point, we then develop probabilistic summarisation models which do not have a correspondence in ad-hoc retrieval models. Some of the models developed here will be implemented in the POLIS summarisation logic.

Chapter 6 provides a formal specification of POLIS. Specifically, **Section 6.2** details the development of the POLIS syntax, and provides BNF trees of POLIS. This is followed by **Section 6.3**, which discusses the possible worlds semantics employed by POLIS. Finally, **Section 6.4** shows how POLIS is implemented. POLIS expressions are translated into a set of equivalent Probabilistic Relational Algebra (PRA, Fuhr and Rölleke (1997)) expressions, which can be executed on suitable IR frameworks (such as HySpirit (Fuhr and Rölleke (1998)), which was used here).

Chapter 7 provides an evaluation of the summarisation models developed in chapter 5. We use two corpora to test the proposed models: the Document Understanding Conference (DUC) AQUAINT corpus, and the INitiative for the Evaluation of XML Retrieval (INEX) 2005 corpus. These two corpora provide a coverage of both main aspects of POLIS: summarisation and structured documents. While the DUC corpus was specifically developed for testing summarisation systems and methodologies, the INEX corpus consists of a large number of highly complex structured documents. The efficiency of the proposed summarisation models is measured using the ROUGE evaluation framework.

Chapter 8 discusses two real-world usage scenarios for POLIS: expert finding, and text snippet generation. Expert finding is an information retrieval task which has gained attention in recent years. Given a set of expert candidates, and a number of documents as evidence, the aim of expert finding is to locate possible experts for a given topic-query. We argue that a summarisation logic can be successfully used to address expert finding. Additionally, we discuss how POLIS can be used to generate text snippets found in the results pages of search engines, and provide samples derived from the Wikipedia collection.

Chapter 9 concludes the discussion. In this chapter, we provide a summary of the topics discussed in this dissertation, and provide an outlook of future work, and possible future research continuing from the work discussed here.

Chapter 2

Basic Concepts of Information Retrieval

2.1 Introduction

Taking the expression "Information Retrieval" literally, IR is concerned with the retrieval of information of any kind, including asking friends for information, or researching a topic in a library. Consequently, Information Retrieval as a discipline is concerned with all aspects of organising and structuring information. However, information retrieval *systems* have traditionally only been concerned with the retrieval of information relevant to a user. A user of an information retrieval system will have a need for specific information (the *information need*) formalised as a query, and will have access to a large body of documents (the *collection*). The task of an information retrieval system is to help a user locate those documents which potentially satisfy her *information need*. This is usually achieved by ranking the documents in the collection by their relevance to the user query.

These three blocks – a user's information need formulated as a query, a collection of documents, and a system's response to the user query – can be arranged to form the traditional *conceptual* model of information retrieval, shown in Figure 2.1. A collection of documents is indexed, resulting in an internal document representation, which can be further processed by the system. Similarly, a user's information need is formulated by the user as a query, which is stored in the system in some internal representation, which might undergo further processing.

The query and document representations are matched using a retrieval function, resulting in a list of documents ranked by their relevance to the user's information need. Matching documents

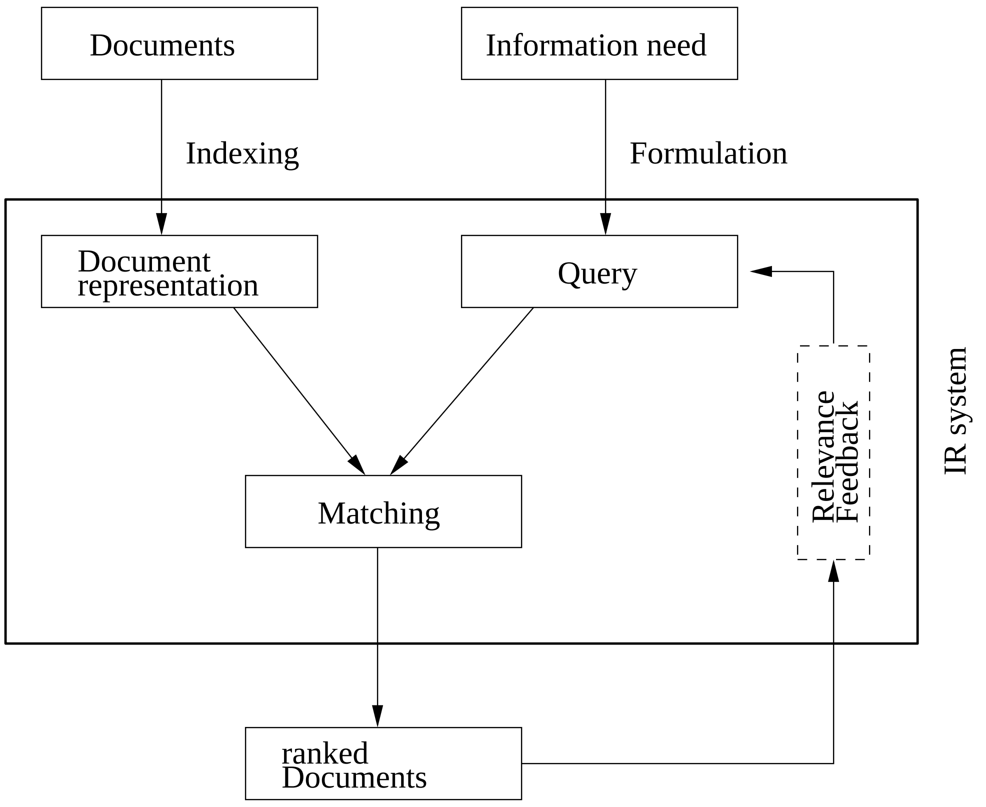


Figure 2.1: Conceptual model of a typical IR system (Croft (1993)).

and queries is one of the main research areas of information retrieval, where the goal is to find an "ideal" model, which will always rank documents relevant to a user's information need above those non-relevant.

The list of ranked documents is presented to the user, and can be used to reformulate the query, by transforming the query according to provided relevance feedback. Relevance feedback is optional, and can be included either as genuine user feedback, or more often as pseudo-relevance feedback, where the top ranking documents are assumed to be relevant.

In this chapter, we look at the main concepts outlined in the conceptual model of information retrieval. We start by discussing ways in which documents and user's information needs can be represented in an IR system. This is followed by an investigation of options to match the document representation and the query. Our discussion then turns to evaluation measures used to judge information retrieval systems. All of these topics are common to different information retrieval tasks. The last section in this chapter will therefore look at how these different topics are relevant to document summarisation, and where document summarisation diverges from the conceptual IR model.

2.2 Document and Query Representation

In a typical information retrieval system, documents are not processed in their original form, but are handled in an internal representation which is the outcome of some *indexing process*. The aim of this internal representation is to model the content of the original document as accurately as possible, while ignoring details irrelevant or out of scope for the information retrieval system. This scope is determined by the information retrieval system, and might therefore vary from system to system; for example, typical IR systems will ignore text formatting and images, whereas a multi-media IR system might include images in its internal representation.

The process of indexing transforms an input document into a document identifier, and a set of features, which are assigned to the document identifier. In the most simple form, features might be a list of words or *terms*, which correspond to the words present in the original document, however, more complicated indexing schemes are conceivable, where features are not individual terms but term-phrases, or semantically augmented concepts (Lewis and Jones (1996)). Although higher-level indexing features have been used, most IR systems will assign simple terms as indexing features.

2.2.1 Term Pre-processing

Before terms are assigned to documents as indexing features, they will usually undergo further processing steps, such as case-normalisation; two other common processing steps, which we will discuss here, are *stemming* and *stop-word removal* .

Stemming is a process to reduce the risk of inadvertently mismatching terms in the document representation and the query, by reducing terms both in the document representation and the query to their *stem*. For example, a document represented by the feature "stemming" would not be retrieved by a query containing the term "stemmed", although both refer to the same concept. A *stemmer* would reduce both "stemming" and "stemmed" to their common root form "stem", thus producing a match between document and query. One common class of stemmers are so-called *suffix strippers*, such as the widely used Porter-Stemmer (Porter (1997)).

Stopword removal has two aims: to reduce the overall size of a document representation, and to reduce "noise" in the documents. In his 1958 work, Luhn pointed out that when ordering words in a collection of documents by their number of occurrences, the content bearing, or *significant* words, would occupy the middle ranks, while the very frequent and very rare words would confer little information regarding the content of documents (Luhn (1958)). While Luhn considered the significance of words in the context of automatic summarisation only, similar arguments also apply with regard to the representation of documents in an information retrieval system (van Rijsbergen (1979)). Low content bearing words such as articles ("the") and prepositions ("in", "at", "of") can be safely removed from the document representation without losing significant content, reducing the overall representation size by up to 50% (van Rijsbergen (1979)).

2.2.2 Term Weighting

Features assigned to documents should be as *discriminating* between documents as possible, such that assigned features are very specific to the documents to which they were assigned. In order to maximise the discriminativeness of terms, and to assign a large number of features to documents, indexing schemes assign weights to terms to quantify the degree to which they discriminate documents. Term weighting is a continuation of the work of Luhn (Luhn (1958)), where terms are ranked according to the frequency with which they occur within a block of text (e.g. a document). According to Luhn, highly significant words are those occurring with a medium frequency: high frequency terms are most likely stop-words, whereas low-frequency words are unlikely to be used in a query.

Accordingly, indexers assign to terms in documents a weight based on their number of occurrences: the *tf*, or *term-frequency* weight. This tf-weight can be estimated by a range of functions, all of which focus on the idea of measuring how often a term occurs in a document with respect to the the total number of terms. Weighting usually only occurs after stop-words have already been filtered out of a document representation, so it is safe to assume that the highest frequency terms at the weighting stage are medium frequency terms in the raw document representation. The tf-measure assigns a higher weight to terms occurring more frequently. If $n_L(t,d)$ denotes the number of occurrences of term t in document d, and $N_L(d)$ denotes the total number of term occurrences in document d, a reasonable estimate of tf is $\frac{n_L(t,d)}{N_L(d)}$. Other tf estimates try to normalise the weight according to document length or maximum tf (e.g. Singhal et al. (1996); Salton (1971)).

In addition to weighting terms by their within-document occurrences, Sparck-Jones proposed to include information about the prevalence of a term in an entire collection of documents (Jones (1972)). This so-called *inverse document frequency* (idf) also provides a measure of the discriminativeness of a term: if a term occurs in few documents, it is highly discriminative, and its *idf* score should be high. The inverse document frequency of a term t is measured as the logarithm of the total collection size divided by the number of documents containing t: $\log\left(\frac{|D|}{|d_t|}\right)$, where $|D|$ is the size of the document collection, and $|d_t|$ is the number of documents containing term t. A combination of *tf* and *idf* has led to the development of the *tf-idf* retrieval models, which we will discuss in section 2.3.

Our outline of term weighting schemes provided here is by no means comprehensive. Instead, it is meant to highlight the ideas behind weighting schemes, such as distinguishing between a document-term frequency and a collection-term frequency, which can also be found in the probability mixtures of Language Modelling approaches, or the different probability estimates in the probabilistic retrieval models. We will discuss this in greater detail in chapter 5.

2.2.3 Data Structures

A third aspect concerning the representation of documents in information retrieval systems are the data structures which contain the internal document representations. While this aspect is usually hidden from users, and does not directly influence the choice of weighting scheme or retrieval model, we believe it still warrants a closer look, if only for sake of completeness. While a large number of data structures suitable for the representation of documents have been proposed, we will only focus on three strategies, covering three major families of data structures.

The first family of data structures are known as *inverted lists* (Harman et al. (1992)). In this data structure, for each index term a list of all documents in which it occurs is stored. At retrieval time, it is possible to immediately retrieve all documents in which the term occurs. Additionally, it is possible to augment the entries in the inverted list such that the head of the list also contains the *idf* weight, while all entries in the list represent both the documents in which the term occurs, as well as the *tf* weight in the document. Inverted lists allow for very efficient implementations of retrieval models.

A second family of data structures are centered around the concept of *document vectors* (Salton et al. (1975)). Each document in the collection is represented by a vector of terms, where the weight of a term is represented by the value of its dimension. The dimensions of the vector can be set to either the number of terms in the document, or the number of terms in the collection, such that all collection terms not present in a document are set to zero. The vector representation is a very “natural” representation for systems implementing vector based retrieval.

The last family of data structures discussed here represents documents using a relational approach (Rölleke and Fuhr (1996)). In the most simple form the content of documents is represented using a binary relation *term*, where individual rows in the relation represent a term together with document in which it occurs. Probabilistic relational operators applied on the "term" relation carry out the calculation of term weights. Relational data representations are commonly used in the context of integration of IR and Database Management Systems (DBIR, or IR-on-DB).

2.2.4 Query Expansion and Reformulation

All the processing steps outlined for the indexing of documents similarly apply to queries. Query terms are stop-word filtered, stemmed, and weighted in the same way as as terms in documents. However, information retrieval systems can optionally include *query expansion* and *query-reformulation* facilities, by which the representation of the query is augmented (Hancock-Beaulieu (1992)).

Query expansion aims to increase recall, by adding to the expanded query all those terms related to the concepts expressed in the original query. For example, if a user enters the query "car manufacturers", the query could be extended with all terms that also refer to the concept "car", such as "automobile", or more general terms, such as "vehicle" (Qiu and Frei (1993)). Expanding the query in this way reduces the risk of a mismatch between terms in the query and terms in documents, and can be implemented by giving the IR system access to a thesaurus.

The unguided bulk expansion via a thesaurus can be turned into a more guided query expansion strategy, by using *relevance feedback* (Rocchio (1971)). Relevance information provides an IR system with feedback on the quality of documents retrieved in response to a query. This information can then be used to reformulate the query, for example by increasing the weight of query terms found in "good" documents, and decreasing the weight of terms found in "bad" documents. The query can also be expanded with the top-weighted terms found in "good" documents. Relevance feedback can be provided interactive, where a user actively labels returned documents as relevant or non-relevant, or can be given as pseudo-relevance feedback. Pseudo-relevance feedback is a two-staged retrieval strategy, where the system retrieves documents in response to the original query, augments the original query by assuming the top-K returned document to be relevant, and presents a user with the list of documents returned for the augmented query (Mitra et al. (1998)).

While query expansion can increase recall, and relevance feedback can improve precision, there is a risk in both strategies, called *topic-drift*. Topic drift occurs if the meaning of the original query is distorted by expansion and reformulation. While this is less of a problem with interactive relevance feedback, where the user actively reinforces the intended meaning of the query, it can become a problem with too aggressive automatic expansion strategies (Clough and Sanderson (2004)).

2.3 Document and Query Matching

The task of an information retrieval system is to find in a collection of documents those which most likely fulfil a user's information need. However, a user's information need is usually not directly accessible, but is encapsulated in the entered query. A retrieval system will therefore attempt to rank documents with respect to the query, according to some ranking model. The assumption here is that that those documents which best match the query will be most relevant for the user.

In this section, we will look at some ranking models to provide a historical timeline, and to highlight the steps which led to the development of modern retrieval models, which will be discussed in chapter 5.

2.3.1 Boolean Matching

Boolean matching was one of the earliest approaches for matching queries and documents. A query is interpreted as a logical expression, where in the most basic case terms are assumed to form a conjunction (Baeza-Yates and Ribeiro-Neto (1999)). For example, the query "car manufacturers" is interpreted as meaning "car AND manufacturers". In more advanced boolean retrieval systems, the inclusion of keywords , such as OR and NOT allows for a larger degree of freedom when forming a query.

Documents are retrieved if they subsume the logical expressions; for the sample query given, all documents containing the terms "car" and "manufacturer" would be retrieved.

Retrieval systems based on boolean matching suffer from a number of problems. The conjunctive interpretation of queries leads to the exclusion of documents which would otherwise be considered relevant for a user if they do no match a single query term; a careful query formulation is necessary in order to achieve acceptable recall levels. Furthermore, boolean matching does not allow for any ranking of documents; users would face a large, unordered collection of documents in response to their queries.

2.3.2 Coordination Level Matching

A second approach for matching queries to documents is the *coordination level matching*. Here, the similarity between a document and a query is measured as the number of terms overlapping between document and query. Assuming that both document and query are represented by a list of terms of length n, where n is the number of terms, such that a position i in the list is set to 1 if term i is present in the document or query, and is set to 0 otherwise, we can formally define the coordination level similarity as:

$$\text{Similarity}(D,Q) \quad := \quad \sum_{i=1}^{n} D_i Q_i \tag{2.1}$$

The simple coordination function 2.1 can additionally be augmented by length normalisation measures. Coordination level matching overcomes the problem of low recall by too specific queries as outlined for boolean queries. Additionally, coordination level matching allows for a ranking of documents. However, only the presence or absence of terms is considered for ranking documents, ignoring term weights which might have been assigned by an indexer.

2.3.3 Cosine Coefficient Matching

A more advanced matching model is *cosine coefficient matching*, a geometric interpretation of documents and queries. In this model, both document and query are interpreted as vectors, and the similarity between query and document is measured as the cosine of the angle between the two vectors, such that it varies between 0 for unrelated documents and queries (where the angle between vectors is 90°), and 1 for identical vectors (where the angle is 0°). The cosine similarity measure can be defined as:

$$\text{Similarity}(D,Q) \quad := \quad \frac{\sum_{i=1}^{n} D_i Q_i}{\sqrt{\sum_{i=1}^{n} (D_i)^2 \cdot \sum_{i=1}^{n} (Q_i)^2}} \tag{2.2}$$

The cosine measure allows for a more refined matching between documents and queries, however, as with coordination level matching, term weights are not reflected in the model.

2.3.4 Vector Matching

Another geometric matching function between documents and queries is the *vector matching* (Salton and McGill (1986)). Both documents and queries are vectors in an n-dimensional vector-space, where n corresponds to the number of distinct terms in the collection. The absence of term t_i in a document is denoted by setting the document vector's dimension i to 0, however, unlike in the cosine similarity measure, the presence of a term is denoted by setting its dimension to the weight assigned by an indexer. The similarity of documents to the query is measured as the angle between the query vector and the document vectors, such that smaller angles denote a higher degree of similarity.

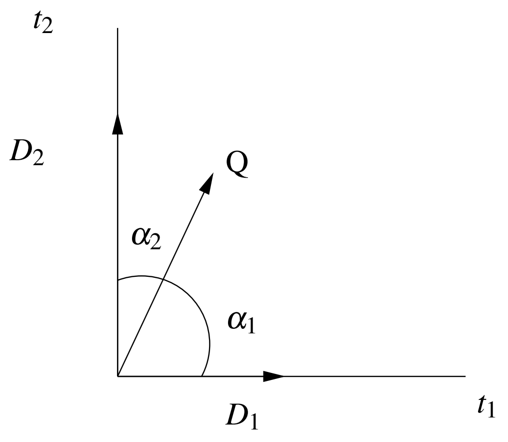


Figure 2.2: Vector Space Model: Document D_1 contains term t_1, document D_2 contains term t_2. Query Q contains both t_1 and t_2.

Figure 2.2 shows an example of matching two documents D_1 and D_2 against a query Q. D_1 contains only term t_1, and document D_2 contains only term t_2; query Q contains both t_1 and t_2.The similarity between D_1,D_2 and Q is measured as the angle between the representation vectors, here α_1 and α_2. In the given example, D_2 is more similar to Q than D_1, as α_2 is smaller than α_1.

2.4 Evaluation in Information Retrieval

The purpose of an information retrieval system is to help a user locate those documents most likely to fulfil his information need. To develop more efficient retrieval models, it is necessary to first find criteria by which an information retrieval system can be judged. Developing sufficient evaluation criteria is a complex task, as the perceived benefit of an IR systems is influenced along a number of dimensions: a good IR system not only has to return relevant documents, but should also do so quickly, and be easy to use.

The ease of use is of concern mostly to research carried out in the context of user interaction, and will not be discussed any further. This leaves two main criteria by which IR systems can be judged: their *efficiency*, which concerns the speed with which a system can index documents and process user queries, and their *effectiveness*, which measures how well a system returns those documents relevant to a user query. While efficiency is important for the perceived usefulness of information retrieval systems, and is a focal point of research in the context of DB-on-IR, we will mainly focus on effectiveness here.

2.4.1 Information Need and Relevance

Any information system user's motivation for using the system is his or her information need: the actual or perceived lack of information necessary to perform a task. The user translates this information need into the actual query presented to a system. The task of any information system is to provide those documents in the collection most likely to contain information pertaining and fulfilling the user's information need. The criterion on which the decision whether to present a document or not to the user is made is called *relevance*. A relevant document is thus a document that both pertains to and fulfils a user's information need. Information systems should ideally also sort those documents shown to the user by their degree of relevance, such that most relevant documents are ranked highest.

2.4.2 Performance Measures

Effectiveness is measured in terms of the quantity of relevant documents returned in response to a user query (Cleverdon et al. (1966)). Several measures of effectiveness have been proposed over the years, the most common of which will be discussed here.

Before we can start to talk about these retrieval methods, we first need to look at the different states, or categories, a document can be in. Given a document collection and a query, a document can be either relevant or non-relevant for the query. This state, *relevance*, is independent of the actual IR system used. When an actual IR system processes a given query, it will retrieve a number of documents. These documents can be classified as *retrieved*, while all other documents remain *non-retrieved*. Every document in the context of an information retrieval system is thus in one of four possible states: *relevant retrieved, relevant non-retrieved, non-relevant retrieved,* and *non-relevant non-retrieved* (see Figure 2.3).

We can now define effectiveness measures in terms of these categories. The most commonly used measures are *precision* and *recall*. Recall is defined as the fraction of relevant documents which have been retrieved, while precision is the fraction of retrieved documents that are relevant.

Documents	retrieved	non-retrieved	
relevant	$A \cap B$	$\neg A \cap B$	B
non-relevant	$A \cap \neg B$	$\neg A \cap \neg B$	$\neg B$
	A	$\neg A$	

Figure 2.3: Categories of documents.

We can thus formally define precision and recall as:

$$\text{Precision} = \frac{|\, A \cap B \,|}{|\, A \,|} \qquad \text{Recall} = \frac{|\, A \cap B \,|}{|\, B \,|} \tag{2.3}$$

where $|\, A \cap B \,|$ is the number of documents relevant and retrieved, $|\, A \,|$ is the number of documents retrieved, and $|\, B \,|$ is the number of documents relevant. Another, less commonly used, measure of effectiveness is *fall-out*, which can be interpreted as the inverse of recall. Fall-Out is defined as the number of non-relevant documents retrieved out of all non-relevant documents, or more formally:

$$\text{Fall-Out} = \frac{|\, A \cap \neg B \,|}{|\, \neg B \,|} \tag{2.4}$$

Precision and Recall can be combined as a weighted harmonic mean, known as the *F-measure*, which can be generally defined as (van Rijsbergen (1980)):

$$F_\beta = \frac{(1+\beta^2) * (precision * recall)}{(\beta^2 * precision + recall)} \tag{2.5}$$

The value of β determines the influence of recall on the weighted mean; recall is "β"-times as important as precision. Commonly used F-measures are F_1, where recall and precision are evenly weighted, $F_{0.5}$, where precision is weighted twice as much as recall, and F_2, where recall is weighted twice as much as precision.

2.4.3 Test Collections

For all these outlined effectiveness measures, it is necessary to have knowledge of both the collection size and the number of relevant documents in the collection. While the size of the collection might be recorded by an indexer, it would be impossible to acquire relevance judgements for all collections and all possible user queries. To overcome these limitations, a number of *test collections* were constructed, which consist of a collection of documents, a set of queries, and a list of documents in the collection judged relevant by domain experts (Sparck-Jones and Rijsbergen

Collection	number of documents	number of queries
Medlars	1033	30
Cranfield 1.400	1398	225
CACM	3204	65
Reuters-21578	21.578	5 categories
RCV-1	806.791	103 categories
DUC-AQUAINT	1250	50
TREC-1	742.611	50
TREC-8	1.634.243	400 (total)
E-TREC Emails	174,311	55
TREC Terabyte	$\sim$ 25.000.000	50

Table 2.1: IR test collections

(1976)).

Judging the relevance of documents has become more problematic over time, as the size of collections has increased: while for the initial, small, test collections, such as the Cranfield collection, *total* relevance judgements were possible, where every document in the collection is judged for relevance, more recent, larger collections, such as the TREC-1 to TREC-8 collections (Harman (1993); Voorhees and Harman (2000)), do not allow for exhaustive relevance judgements, so that documents are only *partially* judged. Table 2.1 lists some commonly used IR test collections.

One way to carry out partial judgement is *pooling*: all IR systems participating in an evaluation conference report their top-K results for a given query; these results are *pooled*, and the pooled set of documents is judged for relevance. However, pooling favours systems that participated in forming the pool, thus giving systems that did not participate in the pooling a disadvantage at evaluation time (Voorhees and Harman (2000)).

While most test collections were constructed for evaluating the performance of ad-hoc retrieval systems, there are additional collections dedicated to evaluating other IR tasks. The Reuters-21578 and the RCV-1 corpora were constructed to aid the evaluation of text classification systems (Hayes and Weinstein (1991); Lewis et al. (2004)). Both corpora consist of Reuters newswire stories, a set of categories, and an assignment of documents to categories as a "ground truth".

The DUC-AQUAINT corpus was built to aid the evaluation of multi-document summarisation systems. It consists of newswire stories from different sources, which are grouped into 50 topics. Each topic has a description, which could be interpreted as a query. The task is to generate a summary for each topic, which is compared to a set of given "reference" summaries.

Evaluation of summarisation systems is a notoriously difficult task, and will be discussed in more depth in Chapter 3. However, there are subtleties and difficulties with the evaluation of IR systems in general, such as the validity of pooling outlined above. Robertson devised a methodology for carrying out experiments in IR, and for the validity of inferring results from the experiments (Robertson (1981)).

2.5 Document Summarisation

The traditional purpose of an information retrieval systems has been to produce a ranked list of documents in response to a user query, without informing the user about the contents of the documents returned. In his 1968 book, Lancaster explicitly stated that

> ”An information retrieval system does not inform [...] the user on the subject of his enquiry. It merely informs on the existence [...] of documents relating to his request.” (Lancaster (1968))

Lancaster’s statement was correct with respect to early information retrieval systems, which only reported on the existence of documents relevant to a given query. However, its validity is questionable with regard to modern IR system, which provide text snippets highlighting the used query terms in-situ, thus informing the user on the content of documents.

Furthermore, Lancaster’s interpretation of what an IR system does and does not do to a user’s state of knowledge directly conflicts with the view of summarisation systems as a subclass of information retrieval systems: a summarisation system *does* inform a user, by providing a condensed representation of one or more documents supplied by a user, or by presenting a condensed representation of documents pertaining to a user’s information need (which, however, have not been provided by the user), as is the case with a number of modern IR ad-hoc retrieval systems.

In addition to using summarisation systems to present a condensed information representation to users, summaries have successfully substituted full documents for the purpose of indexing, resulting in a slimmer document representation (see e.g. Sakai and Sparck-Jones (2001)).

2.5.1 Summarisation Need

Although we defer a detailed discussion of the different aspects pertinent to summarisation systems and summarisation in general to the next chapter, we here turn to the influence of an *information need* on summarisation approaches. The conceptual model of information retrieval

(Figure 2.1) shows that in traditional ad-hoc retrieval the actual, materialised query accessible by a retrieval system is derived from an inaccessible information need. This information need derives from a user's lack of knowledge or understanding; the user hopes to fill this lack of knowledge with information gained from those documents retrieved in response to the materialised information need.

Query-based summarisation systems[1] have been used in conjunction with ad-hoc retrieval systems, to provide a condensed representation of the retrieved documents. In this context, the query used to guide the summarisation process was the original query formulated from the user's information need. The information need that implied the original query thus also affects the summary generation.

However, with abstraction layers to summarisation such as POLIS, or stand-alone summarisation systems, it is conceivable that users would input a specific document to a summariser, which they know contains relevant information, and would provide a query *specifically for the purpose of summarisation*, not for finding documents to fill an information need. Following the conceptual model of information retrieval, what would this query be motivated by? We argue that such a summarisation query originates from a *summarisation need*, which, however, is actually a more specific subclass of the traditional information need. We will illustrate this with an example.

Suppose a user is interested in biographical information on the democratic candidates for the 2008 US presidential election. The user also has a document which contains all the necessary information, but is too long and unspecific to be useful. A query-based summarisation system could thus be asked to provide a summary, guided by the summarisation query "biographical information democratic candidates 2008 US presidential election". While this query is motivated by the specific need for a summary, it actually still stems from an information need. The query could similarly have been used for a question answering system, or for a focused retrieval approach in the context of structured document retrieval (see Chapter 4).

For the remainder of this dissertation, we will therefore not distinguish between a traditional information need, and the queries derived from it, and a summarisation need, and the queries originating from this summarisation need.

[1]We will discuss the differences between generic and query-based summarisation approaches in Chapter 3.

Chapter 3

Literature Review and Background of Document Summarisation

3.1 Introduction

In this chapter, we discuss methods of document summarisation, and other aspects and problems pertinent to the summarisation process.

Before we can look at methods of automatic text summarisation, we need to find a definition of what a summary is. Hovy defines a summary as “a text that is produced from one or more texts, that contains a significant portion of the information in the original text(s)” (Hovy (2005)). Hovy also rather arbitrarily specifies that a summary is “no longer than half of the original text(s)” (ibid). While the ideal length of a summary (and thus its inverse, the compression rate) depends on the context in which the summary will be used, Lin was able to show that most methods used for automatic summarisation perform best for summaries between 15% and 40% the size of the original document (Lin (1999)).

Sparck-Jones (Sparck-Jones (1998)) and Hovy and Lin (Hovy and Lin (1998)) identified a number of different types of summaries, which can be classified along a number of dimensions:

- *indicative* vs *informative* vs *critical* summaries: indicative summaries provide a preview of the content of the full text, while informative summaries should cover all main topics of the full text. Critical summaries should additionally express opinions about the source text(s).

- *extract* vs *abstract* summaries: extracts are generated using material verbatim present in the source text, whereas abstracts regenerate some of the content.
- *single document* vs *multi document* summaries: single document summaries provide a summary for a single text, while multi document summaries provide a single summary for a number of source texts.
- *generic* vs *user oriented (query based)* summaries: generic summaries are generated considering only the material present in the source text(s), while query-based summaries include external information (such as a query) into the generation process.

While research on automatic summarisation began as early as the late 1950s (Luhn (1958)), it suffered a hiatus of some decades. Most of the material discussed in this chapter will therefore only go back to the early 1980s.

3.1.1 Automatic Summarisation: Key aspects

Automatic text summarisation is determined by a number of key aspects, which have been addressed by research in the area. Conceptually, some of these aspects could be regarded as individual modules, or stages, of a summarisation system, while others cover problems surrounding research on summarisation, such as evaluation. While a clear distinction between some of these aspects is not always possible, the summarisation process can be broadly split into three phases (Hovy (2005); Sparck-Jones (1998)):

- *Topic Identification:* Detecting the most important concepts in documents. (Section 3.2)
- *Topic Fusion (Interpretation):* Combining the extracted material into a coherent representation. This includes a discussion of the difference between *extract* and *abstracts*, and composition preprocessing steps such as anaphora resolution. (Section 3.3)
- *Summary Generation:* Production of the summary using techniques such as text- and sentence-planning. (Section 3.4)

Section 3.5 provides an overview of logic-based retrieval frameworks. This is followed by section 3.6, where we will look at evaluation methods for automatically generated summaries. We also show how existing summarisation system allow for user control in section 3.7.

3.2 Topic Identification

The first stage in any summarisation approach is the topic identification phase, in which individual components of text (such as words, sentences, or paragraphs) are given a score. Only high scoring components are used for summary generation.

3.2.1 Feature-based Measures

Luhn in his early work on summarisation proposed to measure the significance of sentences based on the frequency of the sentence words (Luhn (1958)). A sentence is more significant if many frequent terms (i.e. terms occurring often in the entire document) occur in close proximity. Furthermore, Luhn proposes to employ "bracketing" for extraction, extracting only the fraction of a sentence were frequent terms occur in close proximity, ignoring more distant parts of the sentence. A number of other researchers have created summarisation methods which include frequency information for sentence scoring. The systems by Hovy and Lin (Hovy and Lin (1998)), Wei *et al.*(Wei et al. (2008)), Teufel and Moens (Teufel and Moens (1998b)) and Kupiec *et al.* (Kupiec et al. (1995)) all include term frequency information. Furthermore, Witbrock and Mittal describe a *generative* model of summarisation, where term frequency information is used to calculate the probability that a word in a text will also appear in its summary, given a number of features (such as word length, average sentence length, neighbouring words etc.) (Witbrock and Mittal (1999)).

Baxendale developed a summarisation model which considered the location of sentences in the document to be important evidence (Baxendale (1958)). Similarly, Edmundson proposed a method of determining the importance of sentences that included a location feature in the sentence score (Edmundson (1969)). In addition to using term counts, Edmundson proposed to include cue words, title and heading words, and sentence location in the score estimation. Cue words are split into "bonus" and "malus" words, which respectively raise or lower a sentence score. The score of a sentence is also raised if it contains words present in the document title and headlines, or if its location is early in the document. The overall score of a sentence is determined as a weighted linear combination of the individual method scores, $a_1C + a_2F + a_3T + a_4L$, where a_1 to a_4 are the weighting parameters. Edmundson does not make it clear where these weights come from. A main contribution of this research is Edmundson's identification of key features to be used for scoring sentences, which have subsequently been used in a large number of summarisation systems.

In the research reported so far, features were used directly to assign a score to a sentence. Going beyond the direct scoring of sentences, researchers also evaluated the use of these surface level features as input to machine learners.

Kupiec *et al.* developed a machine learning approach based on features similar to the ones used by Edmundson (Kupiec et al. (1995)). In their work, technical documents are compared to given reference abstracts. A Bayesian classifier is used to weight individual features based on their presence in the reference summaries. Kupiec *et al.* found that term frequency information alone resulted in the poorest performance, while positional information alone showed the best individual performance. However, a combination of different features improved the performance of their system compared to any individual features. Teufel and Moens recreated the experiment by Kupiec *et al.*, and confirmed their findings (Teufel and Moens (1998a)).

Teufel and Moens used cue phrases and machine classification as a tool for determining the rhetorical status of a sentence (Teufel and Moens (1998b)). For their experiments, Teufel and Moens trained their classified on scientific articles. Sentences were classified according to whether they conveyed information on the goal of the reported research, background information, methods used etc. Classified sentences were then used to fill a rhetorical template.

Aone *et al.* showed that while a combination of features improves the performance of a summarisation compared to individual features, the ideal combination of features is heavily dependent on genre and the actual documents (Aone et al. (1998)). Experimenting on the TREC 5 corpus, they found that the location feature provided the best individual performance, which could be improved by adding other features. However, although all test documents were newspaper articles, the actual "ideal" combination of features was different for different subcollections (corresponding to different newspapers).

A variant of these machine learning approaches based on Bayesian classifiers are measures based on Hidden Markov Models (HMM). Conroy and O'Leary propose a HMM that given a set of features computes the probability that a specific sentence is a summary sentence (Conroy and O'Leary (2001)). They claim that preferring HMMs over Bayesian classifiers is advantageous, as HMMs make fewer assumptions about the independence of sentences in document and summary. Using the TREC collection as a test corpus, they report a significant improvement over a naive Bayes classifier.

Another approach at sentence extraction based on HMMs is proposed by Jing and McKeown (Jing and McKeown (1999)). They argue that most human abstractors create abstracts by cutting and pasting parts of sentences in the original document. Using a sufficiently large body of training data, an HMM could be trained to identify the most important parts of sentences.

Kan and McKeown propose a hybrid approach to summarisation, that combines information extraction (IE) methods with question answering and summarisation approaches (Kan and McKeown (1999)). Their system initially determines the foci of a document using a term-frequency measure, and assigns a type (such as "person") to each of them. The system then uses a question answering approach to find those sentences that most likely answer questions based on the type of foci.

Carbonell and Goldstein proposed a method of sentence scoring based on the Maximal Marginal Relevance (MMR) (Carbonell and Goldstein (1998)). In their work, retrieved sentence are scored according to the similarity to the set of already retrieved sentences, such that sentences which are very similar to retrieved sentences will be penalised. Carbonell and Goldstein argue that ranking sentences based on MMR reduces redundancy in the set of retrieved sentences.

Lin and Hovy adopted the location feature to determining the most likely positions of important sentences (Lin and Hovy (1997)). Experimenting on the TIPSTER Ziff-Davis corpus, their approach matched sentences against a set of topical keywords, giving a *yield* for that sentence. Sentence positions were then ranked according to their average yield, resulting in a so called *Optimal Position Policy (OPP)*. Comparing sentences extracted from the text with the provided abstracts they were able to measure the coverage for specific positions. While their experiments confirm the usefulness of the location feature, Lin and Hovy also point out that the derived OPP is genre-specific, and thus does not provide "optimal" positions for all kinds of texts.

3.2.1.1 User-oriented (Query-based) summaries

All the above features are tailored to produce generic summaries, i.e. summaries which are not geared towards a user query. Additional summarisation methods were devised, which take a user query into account when calculating a sentence score.

A query based approach by Goldstein *et al.* is based on a scoring model that combines statistical and linguistic features using a weighted linear combination similar to Jelink-Mercer smoothing in language modelling (Goldstein et al. (1999)). A query in their system is represented as a query vector, and is used to reweight the statistical features of the model.

Similar to the inclusion of a query as an additional feature by Goldstein *et al.*, Strzalkowski *et al.* use query overlap as an additional feature for sentence scoring (Strzalkowski et al. (1998)). A sentence receives one point for each term with which it overlaps the user query. This total query score is then used as an additional feature in the weighted combination of features to give a total sentence score. Tombros and Sanderson report a similar query-overlap feature in their study of the usefulness of query-based summaries for retrieval tasks (Tombros and Sanderson (1998)).

3.2.2 Lexical Connectedness Measures

All the methods discussed so far calculate a sentence score only based on the sentence and the document itself. A number of scoring methodologies were devised which additionally include information about other sentences in the document. These methods commonly try to determine the "connectedness" of sentences, assuming that sentences with many connections are more important.

Mitra *et al.* proposed the use of text relationship maps to measure the connectedness of paragraphs (Mitra et al. (1997); Salton et al. (1997)). Using a threshold, the resulting graph only links the most similar paragraphs. Generating a summary is accomplished by traversing a path in the generated graph, starting from the most connected nodes. Mitra *et al.* discuss various heuristics for traversing the graph in order to produce a summary.

Barzilay and Elhadad propose to use lexical chains to model the source text (Barzilay and Elhadad (1997)). Generated chains are given a weight based on their length and their homogeneity, and only the highest scoring chains are used for the summary generation. Barzilay and Elhadad propose several heuristic methods for extracting sentences, based on membership in the selected chains.

Mani and Bloedorn derive a graph representation of the source document based on lexical similarity, adjacency, and co-reference (Mani et al. (1999)). Words in the source document form nodes in the graph, which are connected by one of the previously mentioned relations. Summarisation is achieved by removing constituents from the generated graph.

Zha models documents as weighted undirected graphs and bipartite graphs for the purpose of keyphrase extraction (Zha (2002)). His approach initially creates a bipartite graph of a document, with edges between terms and sentences, where edges are weighted by the number of occurrences of a term in a sentence. The algorithm also creates an undirected weighted graph of the document, where sentences form the vertices of the graph, and have edges between them if they share common terms; these edges are weighted by the similarity of their two vertices (sentences). This undirected graph is used to cluster sentences into topical groups. Zha argues that the generated clusters form a hierarchy, with finer levels of granularity at lower nodes in the hierarchy. He proposes that such a hierarchy of nodes could thus be used to generate summaries at different compression ratios.

A more sophisticated version of the connectedness measures based on the discourse of texts was proposed by Marcu (Marcu (1997b, 1998)). The central argument, on which Marcu's work is based, is that the nuclei of a rhetorical structure tree (RS-Tree) constructed for the source document constitute an adequate summary. Marcu uses rhetorical structure theory to derive a discourse structure of the document to be summarised (Marcu (1997a)). This rhetorical structure is used to derive an importance score for each sentence.

Hu *et al.* proposed a document-summarisation framework in which the scoring of sentences is influenced not only by other sentences in the same document, but also by information found in other documents pertaining to the document to be summarised (Hu et al. (2008)). In their work, annotations to a document, such as comments on a blog posting, are included in the sentence scoring function.

3.2.3 Linguistic Measures

A third methodology for detecting the most salient concepts in text is centered around the use of linguistic analysis. While there is often an overlap with the measures already discussed, the approaches discussed here still warrant closer inspection.

McKeown *et al.* discuss the use of linguistic features as input to a machine learning system for the purpose of multi-document summarisation (McKeown et al. (1999)). Their system breaks all input documents into paragraph-sized chunks, and compares paragraphs to detect whether they are similar. Similarity here is a binary decision, and is decided based on whether two paragraphs report on the same object. This analysis is carried out using WordNet synonymy information (Miller et al. (1990)) and semantic verb classes. This similarity information is subsequently used to cluster paragraphs, resulting in nodes of topics containing the most salient concepts.

Yang *et al.* propose a similar method in the context of event detection, where sentences are clustered according to their similarity (Yang et al. (1998)). Most similar sentences are assumed to be representative of the topics of a document, and are used for summary generation. Wang *et al.* discuss a similar approach, where sentences are clustered according to semantic similarity

(Wang et al. (2008)). Their approach initially calculates sentence similarity scores which are used to generate clusters of similar documents. The final summary is formed by extracting the most representative sentences from each cluster.

3.3 Topic Fusion (Interpretation)

All the methods discussed produce a set of most salient textual elements of documents. An *extract*-type summary system would directly use these textual elements to generate a summary. However, *abstract*-type summarisers will additionally include a topic fusion, or interpretation, stage, where the given textual components are reformulated, or even enhanced with background knowledge. This implies that *abstract* type summaries are domain specific, as background knowledge has to be contextual. Most summarisation systems to date are of extract type, and research into topic fusion has remained limited.

Ono *et al.* propose a method of sentence generation for Japanese based on sentences rhetorical status (Ono et al. (1994)). Initially, the system builds a rhetorical relation tree for the source document. Sentences in the document are weighted by their rhetorical relations, and are categorised into one of three categories based on the relation. According to the relation, an inner node's right or left terminal node are penalised, and the highest penalised nodes are removed from the tree. The remaining tree forms an abstract of the source document.

Reimer and Hahn developed a highly formalised approach of information condensation based on operators of terminological logics (Hahn and Reimer (1998)). In their work, knowledge representation structures are condensed under operators producing concepts of higher abstraction. They use the example of summarising information on notebooks by subsuming individual models under a higher abstract concept of notebooks. However, it remains unclear how the initial knowledge representation structures could be retrieved from plain text. Reimer and Hahn also do not elaborate on the production of plain text from the condensed knowledge representations.

Hovy and Lin included a topic fusion stage in their summarisation system SUMMARIST (Hovy and Lin (1998)). One approach for fusion was based on recognising the concepts behind individual words using WordNet. Instances of concepts are counted, and the most specific generalised concept of a set of terms is used as their *fuser*. A second fusion approach was based on finding words appropriate as a substitute for a number of other words. To find these *signature* words, Hovy and Lin used the 1987 Wall Street Journal corpus, which is split into 32 classes.

Counting the occurrences of terms in the individual classes, they replaced the highest count terms with the class name.

Conceptually, it would also be possible to derive generalisations for sets of terms using script based inference, such that the set of terms *drive in, pay, fill tank, close tank* generalises to *gasoline fill up* (Hovy and Lin (1998); Schank and Abelson (1977)).

One main obstacle for topic fusion and the generation of *abstract* type summaries is the lack of domain specific knowledge, and the inability to capture domain knowledge from the document (or set of documents) to be summarised.

3.4 Summary Generation

The topic identification stage, possibly followed by a topic fusion stage, results in a conceptual representation of a document's most important topics, internal to the computer the summariser is running on. Additional steps are necessary to translate the internal representation into a textual summary output to a user. This translation may utilise techniques of natural language processing, such as text planning, sentence planning, and sentence realisation (Hovy (2005)).

A summary generation stage is usually more important for extract type summariser, as the extracted sentences might exhibit dysfluencies, dangling anaphora, topic shifts, and general grammatical inconsistencies (Johnson et al. (1997)).

Hirst *et al.* developed a summarisation system for medical records (Hirst et al. (1997)). To generate textual summaries from extracted sentences, their system applies smoothing operators to surface level sentence features. For example, two sentences in close proximity starting with an identical noun phrase are assumed to co-reference the same object; to repair the repetitive nature of the sentences, the noun phrase in the second sentence is pronominalised. The system also exploits rhetorical relationships between sentences. Sentences marked as *evidence* and *conclusion* will be connected by a cohesive implication, such as *therefore*.

Mani *et al.* consider the initial set of extracted sentences as a draft summary, which they augment with background information from the source document (Mani et al. (1999)). The augmented draft then undergoes revision based on rules involving aggregation and elimination operations (Dalianis and Hovy (1996)). The system also involves smoothing operations at sentence level, to arrive at more compact sentences.

The concept of smoothing sentences is similar to the sentence compression operation employed by a number of summarisation systems. Barzilay *et al.* carry out summarisation by means of manually generated sentence compression rules (Barzilay et al. (1999)).

Knight and Marcu investigate two models of sentence compression, one based on a noisy channel EM algorithm and a decision-based deterministic model (Knight and Marcu (2000)). Both models try to compress the syntactical parse tree of a sentence into a more compact one. The use of machine learning in the form of the EM algorithm would help overcome the problem of having to manually define compression rules. Experiments showed no statistically significant difference between the two models.

Banko *et al.* change the perspective on summary generation from sentence *compression* to sentence *generation* (Banko et al. (2000)). Their systems builds a model of surface realisation based on a bigram language model. Sentence generation can now be seen as finding the sequence of words that maximises the probability to have been generated by the document to be summarised. However, their system only produces short, headline-like sentences, and does not extend to multi-sentence summaries. Witbrock and Mittal used a similar bigram language modelling approach to order individual terms extracted from a source document into sentences (Witbrock and Mittal (1999)).

While Witbrock and Mittal use a very aggressive cut-and-paste approach, extracting and merging individual words, Jing and McKeown generate summaries by cutting larger sub-sentence fragments from the source document (Jing and McKeown (1999)). Arguing that human-written summaries are often generated by applying a cut and paste process to the original document, they train a HMM to detect the location of summary sentence fragments in the original document. Applying similar cut and paste operations to unseen documents can then be used to create appropriate summaries.

Radev developed an approach for merging information from multiple documents, where for each document a template was filled by an information extraction approach (Radev (1999)). To produce actual summaries, the templates are merged under composition operators, that also detect inconsistent information, and change the output accordingly. However, his approach only works in specific domains, for which templates have been created.

3.5 Logical Retrieval

The previous sections covered aspects pertinent to finding the most salient concepts in text, and to generating actual summaries from the most salient concepts. However, a summarisation logic like POLIS is also concerned with the creation of an abstraction layer to summarisation, here achieved by means of a logical layer. We therefore in this section discuss how abstraction logics have been applied generally in the context of information retrieval, resulting in the general class of *retrieval logics*.

Retrieval logics are rooted in terminological logics (or *description logics*), which originated from AI research, and the need to build high-level representations of domains of knowledge (Baader et al. (2003)). Information in the knowledge domain is represented by *concepts* in the logic. Reasoning in terminological logics is based on subsumption and concept satisfiability. Subsumption is the process of determining if a concept is more general than another, while a concept is satisfiable if it is not subsumed by the empty concept (i.e. a concept description is not contradictory). While the process of reasoning in terminological logics is vastly more complex than what was discussed here, this presentation is sufficient for the remainder of this section, and any further elaboration on reasoning in description logics would be beyond the scope of the work presented here.

Meghini *et al.* discuss the use of terminological logics in the context of document retrieval (Meghini et al. (1993)). In their work, documents are retrieved if they are subsumed by a query, i.e. if the query is more general and covers the concepts in the document. However, it does not become clear how a *ranking* of document would be achieved by their logic, as the subsumption checking results in a binary decision whether to retrieve a document.

Later research on retrieval logics incorporated probabilistic reasoning into the retrieval process, thus allowing a ranking of documents. POLIS is one member of a family of such probabilistic logics (see Figure 3.1), which we will briefly present here.

POOL is an object oriented retrieval logic, which allows the representation of objects in the logical framework, and makes it possible for the logic to conduct inference and reasoning on the objects in its knowledge base (Roelleke (1999)). POOL is a general retrieval logic, in the sense that it was not developed for specific retrieval tasks. The two elements in the top-most layers in the figure, POLIS and POLAR, are both retrieval logics for specific tasks: POLIS is a summarisation logic, while POLAR is a logic for annotation-based retrieval (Frommholz (2007)).

Annotation-based retrieval determines the retrieval status value of a document not only based on a document's own representation, but also on information conveyed by other documents referring to it.

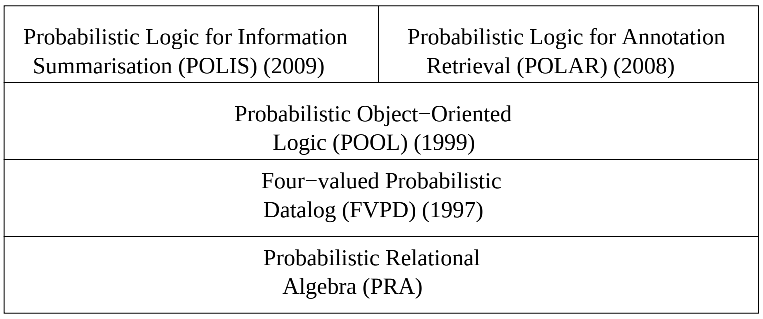

Probabilistic Logic for Information Summarisation (POLIS) (2009)
Probabilistic Logic for Annotation Retrieval (POLAR) (2008)
Probabilistic Object−Oriented Logic (POOL) (1999)
Four−valued Probabilistic Datalog (FVPD) (1997)
Probabilistic Relational Algebra (PRA)

Figure 3.1: Hierarchy of logical retrieval approaches.

Both POLAR and POLIS can be seen as descendants of POOL, as they share semantic concepts and processes of probabilistic reasoning. The two bottom layers of Figure 3.1 display lower level languages, which are used to implement higher-abstraction languages such as POOL, POLAR, or POLIS, rather than be directly used as abstraction layers. FVPD extends traditional Datalog in two ways: propositions can take either of four values (true, false, inconsistent, unknown) as opposed to the normal two valued (true, false) truth assignments, and all truth value assignments hold with a certain probability rather than a deterministic absolute truth. We will not use FVPD in the discussion of POLIS, however, we will occasionally make use of Probabilistic Datalog (PD, not shown in Figure 3.1), which is a probabilistic extension of deterministic Datalog. PD only uses two truth values like traditional Datalog, however, truth value assignments are probabilistic (Fuhr (1995)).

Although Figure 3.1 shows POLIS as a descendant of POOL, performance constraints prohibited an implementation of POLIS in POOL. Instead, the implementation of POLIS relies on Probabilistic Relational Algebra (PRA, found in the bottom layer of Figure 3.1) for the actual implementation. PRA is a probabilistic extension of the traditional Relational Algebra found in database systems, and introduces both tuple probabilities as well as probabilistic relational operators (Fuhr and Rölleke (1997)). We will discuss this in more detail in section 6.4.

3.6 Evaluation Methodologies

The summary generation stage of a summarisation produces the final output. However, there is no indication of whether the summary produced is good. This section will discuss approaches to evaluate the quality of summaries, and will look at different measures of summary quality.

Jones and Galliers classify evaluation methods for summaries into *intrinsic* and *extrinsic* methods (Galliers and Jones (1995)); a third methodology discussed here is the *subjective* evaluation:

- *intrinsic evaluation:* automatically generated summaries are compared with a set of manually written reference summaries.
- *extrinsic evaluation:* also called *task-based* measures, where summaries are judged by how well they support user to carry out a task.
- *subjective evaluation:* evaluators judge the quality of a summary based on coherence, style, and user preference.

We will look at methods belonging to these three classes in the following sections.

3.6.1 Intrinsic Evaluation Methods

Edmundson proposed the use of statistical methods to measure the percentage of extract-worthy sentences (those containing information present in the reference summaries) in the automatically generated summaries (Edmundson (1969)). Based on the comparison to extract-worthy sentences, Edmundson also details to types of errors a summarisation system can make in picking a sentence: *type 1* errors are those where a system does not extract a worthy sentence, and *type 2* errors, where the system picks a sentence not extract-worthy.

Ono *et al.* compared sentences generated by their system with manually selected key sentences in the source documents (Ono et al. (1994)). Human judges picked the most important *key* sentences of articles in the test corpus. The system-generated abstracts were then compared with the selected *key* sentences.

Due to insufficient data to split a collection into distinct training and test sets, Kupiec *et al.* applied a cross-validation evaluation to their system (Kupiec et al. (1995)). For each journal of the test corpus, the journal was repeatedly split into one document used for testing, and the remaining documents used for training. Results were summed over journals.

One obstacle for using intrinsic methods for summary evaluation is the lack of corpora consisting of documents and their corresponding extracts. However, large corpora exist consisting of documents and their abstracts (such as scientific publications). Both Goldstein *et al.* and Marcu proposed methods to create an extract corresponding to a given <*document, abstract*> pair (Goldstein et al. (1999); Marcu (1999)). Goldstein *et al.* expand every sentence in the abstract into the smallest subset of sentences in the full text containing all the key concepts (keywords and noun-phrases) of the abstract sentence. Marcu uses a greedy EM algorithm to generate an extract from an abstract and a full text. His algorithm initially starts with the complete full text, and greedily removes sentences that so that after removal the similarity between the abstract and the residual text is maximised. The process is repeated until removing further sentences from the original text would decrease similarity. The remaining original text is an extract corresponding to the given abstract.

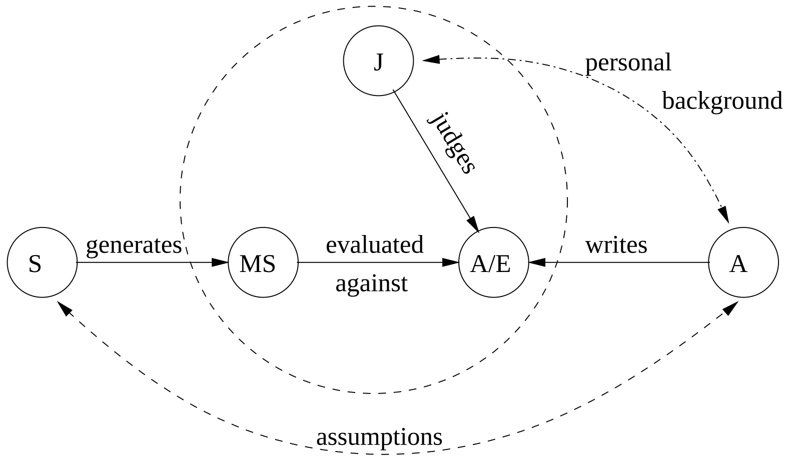


Figure 3.2: A human abstractor *A* writes a summary *A/E*. A summarisation system *S* generates a machine summary *MS*, which is evaluated against A/E. A human judge *J* judges the content of *A/E*.

Another problem for intrinsic evaluations are variations in the reference summaries. A graphical representation of the problem is shown in Figure 3.2. A human abstractor provides a reference summary A/E, which is compared to an automatically generated summary *S*. Evaluation frameworks typically only compare summaries based on their content. However, this means that reasonable but contradicting assumptions made by both the human abstractor and the developers of the summarisation systems which influence the contents of the summaries will lower the overlap of the two summaries, and thus the performance score of the system. This also holds true when summaries are evaluated by a human judge: differences in personal background might lead to different assumptions in what constitutes a "good" summary.

Teufel and Moens show that the content of human written summaries is influenced by the abstractors personal background and knowledge (Teufel and Moens (1998b)). Intrinsic evaluation methods make the semiotic connection that "good" summaries are those which largely overlap with reference summaries. However, since the content of the reference summaries is not bound by objective criteria, judging a generated summary against a single human written reference summary does not provide a significant quality evaluation. Donaway *et al.* show that evaluations of machine summaries may vary to a large degree given different manual reference summaries (each of which would ordinarily be considered ideal) (Donaway et al. (2000)).

Similar to the way a human judges content-evaluation of a manual summary is influenced by his personal background (and the difference in background the abstractor), assumptions made in the construction of a summarisation system will be different to the assumptions taken by individual abstractor when writing a summary. One approach to mitigate the influence of individual author's preferences in the evaluation process, taken by the Document Understanding Conference (DUC), is to provide multiple (in case of DUC four) reference summaries for each topic, against which the generated summaries are matched using cross-validation (Dang (2006)).

3.6.2 Extrinsic Evaluation Methods

One of the largest evaluation studies involving extrinsic evaluation measures with eighteen participant systems was TIPSTER/SUMMAC (Firmin and Chrzanowski (1998)). Summaries of participant systems were evaluated based on their usefulness as indicative summaries in categorisation and ad-hoc retrieval tasks. For the categorisation task, judges independently classified documents and their generated summaries. The quality of a system was measured as the degree of overlap between the two classifications. For the ad-hoc task, judges labelled generated summaries as relevant or not-relevant for the given query.

Similar ad-hoc retrieval evaluations were also carried out by Jing *et al.*, where users had to judge the relevance of document based on summaries, by Wolf *et al.*, where the number of relevant documents correctly identified in a given time span, together with the error rate, was used to compare different summarisation approaches (Jing et al. (1998); Wolf et al. (2004)), and in a substantial number of other publications (Tombros and Sanderson (1998); Brandow et al. (1995); Mani and Bloedorn (1999)).

A third kind of extrinsic evaluation uses reading comprehension as a measure of summary quality. Morris *et al.* based their reading comprehension test on four GMAT (Graduate Management Admission Test) reading comprehension exercises. Human subjects had to pick one multiple choice answer for each exercise, were the exercise text could either be the full text, a machine extract, a manual abstract, or completely blank. The number of correctly answered exercises was used as a summary quality measure.

Maybury used a similar question answering approach, where participants were shown log files of a battle simulation, and business news describing joint ventures (Maybury (1995)). Subjects were shown either the full texts, or summaries thereof, and had to fill in key information presented in the full texts. The amount of correctly filled in data was used as a measure of summary quality.

Hovy and Marcu measured the informativeness of a summary by the extent to which test subject were able to recreate the source document from reading the summary (Hovy and Marcu (1998)). Subjects were given the full text, a summary, or no text. Experimental evidence shows that the test group given summaries performed closer to the full text group than to the no text group.

Although extrinsic evaluation methods have been used for evaluating summaries, they are not without problems. As summaries are evaluated in the context of another retrieval task, the experiments need to be carefully designed to ensure that the evaluation only considers the influence of summaries, disregarding the influence of the actual retrieval system. For example, a relatively poor retrieval system might adversely affect the way users perceive the presented summaries.

3.6.3 Subjective Evaluation Methods

Subjective evaluations can be considered a part of the intrinsic evaluation measures, as summaries are judged in isolation. However, it is possible to split intrinsic methods into two subclasses: those which compare machine summaries to reference summaries, and those which judge the quality of the generated summaries in terms of cohesion, grammaticality, style, and fluency. The following work made use of the latter evaluation method.

Brandow *et al.* used an *acceptability evaluation* as part of the overall evaluation of their system (Brandow et al. (1995)). To judge the quality of their system, experienced news analysts evaluated the summaries according to established industry readability guidelines, additionally using the full text to determine the acceptability of the summary content.

Participant systems in the Document Understanding Conference (DUC) were compared to four given reference summaries, and also judged along subjective quality dimensions, including grammaticality, non-redundancy, referential clarity, focus, and structure and coherence (Dang (2006)).

Although human judgement is valuable for evaluating "non-technical" aspects of summaries such as fluency and grammaticality, the subjective influence of individual judges means that subjective judgements should not be used as the only criterion by which a system is evaluated. Furthermore, a single system should always be evaluated by several judges, to reduce the influence of individual personal opinions in the overall evaluation.

3.7 User Interaction with Summarisation Systems

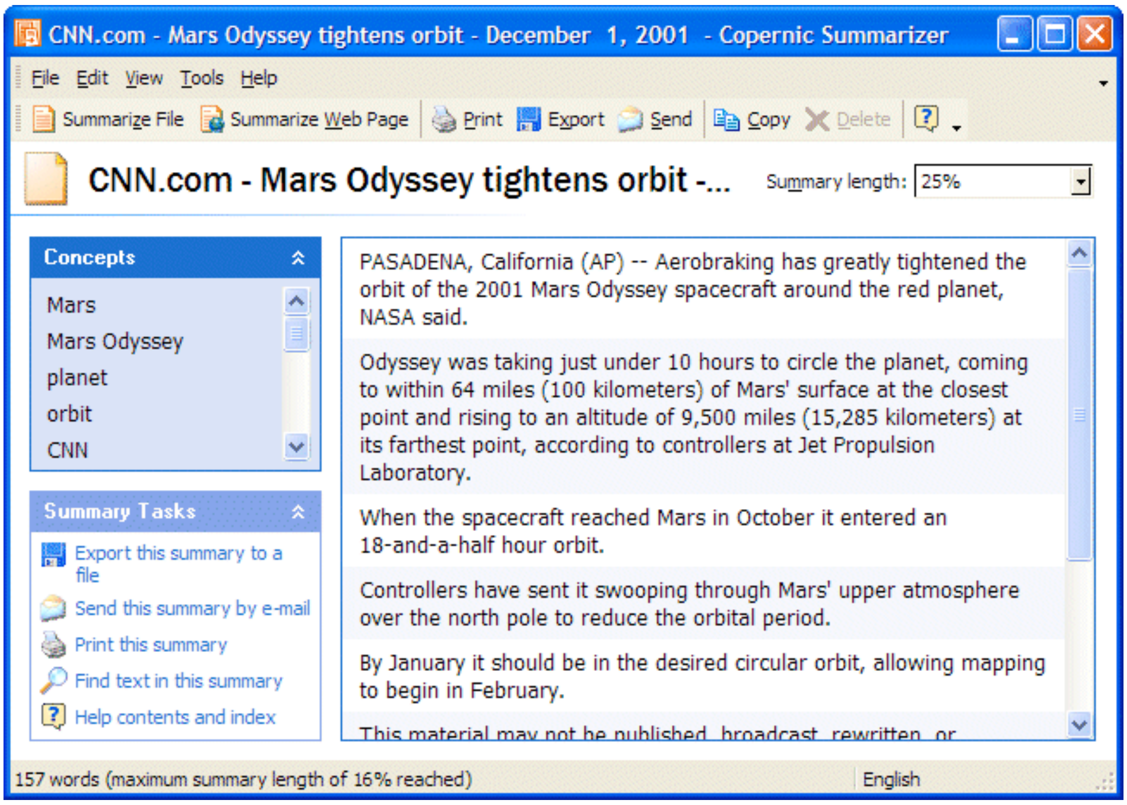
The previous sections discussed how past summarisation approaches generate abstracts or extracts from source documents, and how the quality of the generated summaries can be evaluated by comparison with given reference summaries, or by evaluating the usefulness of generated summaries in real tasks or scenarios.

A third aspect we address now is the user interaction with summarisation systems. There are several parameters which influence the summary generation, such as the choice of summarisation approach, the length of the summaries, or the use of a summarisation query. In this section, we will look at ways in which existing summarisation systems allow users to control these parameters. Controlling summarisation systems is usually performed in one of two ways: by manipulating the summariser through a Graphical User Interface (GUI), or by setting parameters in a configuration file or on the command line. We will present examples of both classes of interaction, starting with GUI-based systems.

3.7.1 Control through Graphical User Interfaces

The first class of summarisation systems allows user to set those parameters controlling the summarisation process through widgets in a Graphical User Interface. The options available depend on the underlying summarisation approach, and the degree to which developers of the system grant users access to relevant controls.

Figure 3.3 shows the GUI of Copernic Summarizer, a summarisation system developed by Copernic Inc. Users can set the summary length (using the drop-down list in the top right corner), and can manipulate the summary's content based on the inclusion or exclusion of specific “concepts” (found on the left side of the screen). Although we will not discuss Copernic Summarizer in more detail here, its GUI can be seen as a template for all those GUIs discussed in the remainder of this subsection.


CNN.com - Mars Odyssey tightens orbit - December 1, 2001 - Copernic Summarizer

CNN.com - Mars Odyssey tightens orbit -...

Summary length: 25%

Concepts: Mars, Mars Odyssey, planet, orbit, CNN

PASADENA, California (AP) -- Aerobraking has greatly tightened the orbit of the 2001 Mars Odyssey spacecraft around the red planet, NASA said.

Odyssey was taking just under 10 hours to circle the planet, coming to within 64 miles (100 kilometers) of Mars' surface at the closest point and rising to an altitude of 9,500 miles (15,285 kilometers) at its farthest point, according to controllers at Jet Propulsion Laboratory.

When the spacecraft reached Mars in October it entered an 18-and-a-half hour orbit.

Controllers have sent it swooping through Mars' upper atmosphere over the north pole to reduce the orbital period.

By January it should be in the desired circular orbit, allowing mapping to begin in February.

This material may not be published, broadcast, rewritten, or

157 words (maximum summary length of 16% reached) English

Figure 3.3: Parameters in the Copernic Summarizer are controlled through a Graphical User Interface. ©Copernic Inc.

Strzalkowski *et al.* present a summarisation system that allows a user to control certain parameters of the summarisation process (Strzalkowski et al. (1998)). A user is able to select whether a summary should be query-based or generic, and if query-based, specify query terms. Additionally, the user can set the minimum compression rate. However, there is no control on the level of granularity at which material is to be extracted.

Aone *et al.* present a system in which sentences are scored based on NLP features. Their system also provides a GUI which allows users to turn on or off individual features of the summarisation model (Aone et al. (1998)).

Leuski *et al.* present the user-interface iNeATS, which controls the summarisation process of the summarisation system NeATS (Leuski et al. (2003); Lin and Hovy (2001)). iNeATS allows users to set the compression rate, sentence cut-off length, and redundancy filters.

3.7.2 Control through Configuration Files

The second class of summarisation systems allows users to control the summarisation process through configuration files, or via command-line options. Parameters such as choice of models, compression ratio, or summarisation query are defined using specific syntactic constructs.

The MEAD summarisation system by Radev *et al.* allows users to fine-tune the summarisation approach using either command-line options or configuration files. Users can select summarisation features, and specify parameters for these features (Radev et al. (2004)).

```
compression_basis sentences
compression_absolute 1
classifier
/clair4/projects/mead307/source/mead/bin/default-classifier.pl
    Centroid 3.0 Position 1.0 Length 15 SimWithFirst 2.0
reranker /clair4/projects/mead307/source/mead/bin/default-reranker.pl
    MEAD-cosine 0.9 enidf
```

Figure 3.4: MEAD configuration file.

A sample configuration file is shown in Figure 3.4. Given this configuration file, MEAD weights sentences according to their length, position, similarity with first sentence, and similarity to centroid of all sentences, and generates a summary which is one sentence long.

While configuration files of this kind provide a more fine-grained access to the parameters of a summarisation system, they also require users to be familiar with details of the implementation of the summariser (here: the path to summariser libraries), and to have detailed knowledge of the features and models implemented by the summariser. There are only limited advantages over manipulating the summarisation systems directly, e.g. by changing the source code.

Furthermore, although MEAD represents documents internally using an XML-like format, users cannot specify the granularity at which material for summaries should be collected.

3.7.3 Discussion

In the two previous subsections, we introduced the two main options by which users can manipulate and control summarisers: Graphical User Interfaces, and configuration files. The main advantage of GUIs is their ease of use, especially for end users who may not be familiar with summarisation models or other concepts such as query-based summarisation. However, GUI-based interaction is always limited to the options provided by the user interface, and as such does not offer the same level of control provided by a command-line or configuration-file based interaction. In comparison to GUIs, configuration-file based control is limited by the number of options which can be parsed by the summarisation system. These latter approaches allow users to tweak more aspects of the summarisation process, however, they are also more complicated to use, and may thus be overwhelming or unusable to end users. The expressive power of a GUI-based summarisation system is limited by the options available through the GUI, while the expressiveness of configuration files is restricted by the total set of options available.

Ideally, a summarisation system would provide a GUI for end users, and would also accept command line options for more advanced users. POLIS tries to integrate both usability and flexibility in a different way. POLIS allows users to select the element granularity at which data for summaries is collected, and also allows users to set the summary length, the model to be used, and – if appropriate – a summarisation query. However, these options are provided through an XPath-based syntax, which is more intuitive than the configuration files used by e.g. MEAD.

By focusing on a specific user group (i.e. knowledge engineers who have some notions about summarisers, or ideas how summaries might be beneficially used in their projects, but no in-depth knowledge of summarisation approaches) POLIS treads a middle ground between the flexibility and complexity of configuration files, and the ease of use of Graphical User Interfaces.

3.8 Conclusion

This chapter presented an overview over the main stages of a summarisation system, and provided details of research into each of these stages. Additionally, we looked at evaluation methods for automatic summarisation, and some of the issued surrounding summary evaluation that make it a difficult topic. These two points will become important in the next chapters, where we will develop summarisation models for POLIS, and will evaluate these models on actual test corpora.

The last section concluded the discussion by presenting an short, indicative overview of user interaction with summarisation systems. While most systems provide interaction via a graphical user interface, POLIS instead provides an XPath-like syntax, which allows users to specify the granularity at which extraction in documents takes place, to set query terms by which user selected elements are to be weighted, to define the models by which element extraction is to take place, and to set the upper summary size limit in terms of either the number of elements, or the number of words.

Chapter 4

Summarisation: Parallels to Other Retrieval Tasks

This chapter will develop a parallelism between summarisation and a number of other retrieval tasks. We believe that highlighting these parallelisms aids the better understanding of some of the complexities surrounding automatic summarisation, and also forms the foundation for some of the discussions in the later chapters.

The other retrieval tasks discussed and compared to summarisation in this chapter are anchor-text retrieval, website partitioning, and structured document retrieval. While these tasks at first seem remote from summarisation (and, additionally, from each other), we believe that they can be related from an abstract, conceptual point of view: all of the retrieval tasks at hand rely in some form on processes of *evidence propagation* or *knowledge augmentation*, which is essential for the summarisation models implemented by POLIS, and the POLIS semantics. While it would have been possible to find parallels to additional retrieval tasks, the tasks compared here share a very similar knowledge augmentation process. Especially for the case of structured document summarisation and structured document retrieval, the process of knowledge augmentation is almost identical.

We start by discussing the idea behind knowledge augmentation and evidence propagation conceptually in section 4.1. Sections 4.2 to 4.5 discuss how knowledge augmentation and evidence propagation relate to the mentioned retrieval tasks, and relates them to each other. Section 4.6 concludes the discussion.

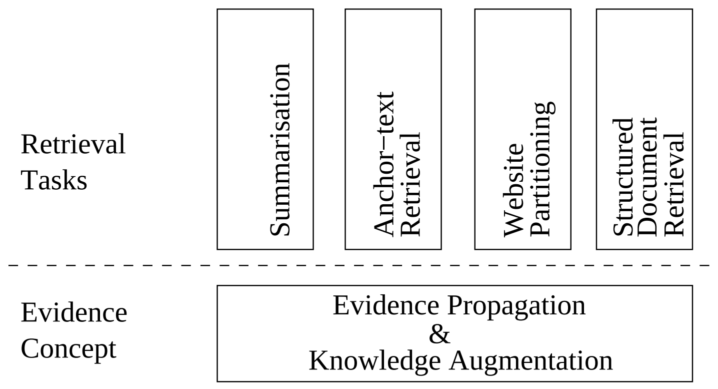


Figure 4.1: Parallelism of retrieval tasks by the common concepts of evidence propagation and knowledge augmentation.

4.1 Evidence Propagation and Knowledge Augmentation

Both evidence propagation and knowledge augmentation attempt to address the same problem: given an item of interest (such as a document, a website, or – for the case of structured document retrieval – subelements of documents), how can the representation of this item be improved such that it incorporates information not intrinsic to the item. This definition might at first seem vague and complex, but formalises a fairly intuitive notion.

Suppose a retrieval system works on a collection of documents, which are either books, or book reviews. Normally, all these documents would be processed in isolation, however, for the given domain, there is a dependency between documents: reviews are about books. Here, reviews add additional information to the representation of a book in isolation, such as comparisons to other books, or a reviewers comments on a book. A searcher looking for a specific book would normally only retrieve that book itself, however, exploiting the information provided by the associated review, he might also retrieve other books which are related to the on he was looking for. The evidence provided by the review is propagated to the representation of the book.

While the given example shows the propagation of evidence between individual documents, it works at any level in the hierarchy of information items. At the one end of the spectrum, a whole indexed collection of documents might be augmented by another indexed collection, while at the other end, a section in a structured document is augmented by knowledge of its sentences. Irrespective of the scale, evidence propagation and knowledge augmentation occur between an *evidence source*, and an *evidence destination*, where the destination is augmented by knowledge of the source.

While the flow of evidence from source to destination is identical for any evidence propagation or knowledge augmentation strategy, there are two types of evidence which can be propagated: term-frequency evidence, and term-probability evidence.

4.1.1 Term-Probability Propagation

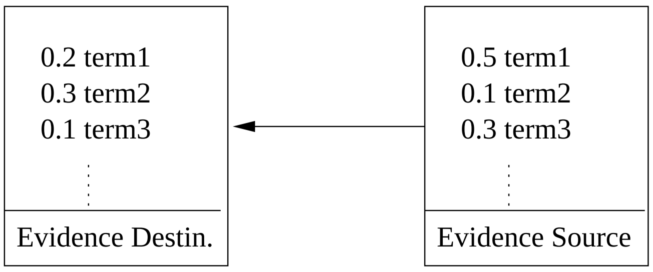


Figure 4.2: Term-probability propagation: probability evidence is pushed from the source to the destination.

The first type of evidence propagation is the term-probability propagation, which augments the source representation by incorporating term *probabilities* provided by the evidence source. The term probabilities (or, more generally, term weights) in both representations would have been generated by common indexers as discussed in Chapter 2. Augmentation of probabilities in the augmented document could be achieved by deriving a simple mean of the term probabilities in both propagation destination and propagation source. For the example in Figure 4.2, the term probabilities in the augmented destination document d_{AD} could thus be given by the estimate

$$P(t|d_{AD}) \quad := \quad \frac{1}{2}P(t|d_D) + \frac{1}{2}P(t|d_S) \tag{4.1}$$

where d_D is the evidence destination document, and d_S is the evidence source document. For a larger number of evidence sources, the augmentation of document term probabilities could be modelled as a weighted sum over the evidence source probabilities. For a document d, which is augmented by n evidence sources, the augmented probabilities could be estimated as:

$$P(t|d_{AD}) \quad := \quad \frac{1}{n+1}P(t|d_D) + \sum_{1\cdots n} \frac{1}{n+1}P(t|e_i) \tag{4.2}$$

Figure 4.3 shows how such evidence propagation schemes could be implemented in Probabilistic Datalog.

While such a naive probabilistic evidence propagation is conceptually simple and easy to implement, it suffers from two problems. Firstly, the reliability of evidence is not accounted for. Every source of evidence is treated equal, such that unreliable sources of evidence contribute

```
        aug_term(T,D) :− term(T,D) | (D).
        aug_term(T,D) :− link (D,D2) & term(T,D2) | (D2).

        w_aug_term INDEPENDENT(T,D) :− aug_term(T,D).
```

Figure 4.3: Probabilistic Datalog expression to perform probabilistic term augmentation.

as much to the augmented probability as very trustworthy ones. However, this problem can be mitigated easily, by not applying a linear weight across all sources of evidence, but instead weighting evidence sources by some reliability criterion.

The second problem concerns the stability of evidence probabilities. A probability could be considered stable if minor changes to the document representation from which the probabilities were derived does not lead to major changes in the probabilities. Suppose the occurrence probability of a term t in a collection c of documents is estimated by the quotient of locations of t, $n_L(t,c)$, and the total number of locations in c, $N_L(c)$: $P(t|c) = \frac{n_L(t,c)}{N_L(c)}$. If t occurs $1,000$ times in a collection of $100,000$ locations, the probability $P(t|c)$ would be 0.01. Suppose we are adding another document to the collection, which consists of 100 locations, eight of which would contain t. In this case, the probability $P(t|c)$ would change to 0.01006993. For all intents and purposes, the probability $P(t|c)$ remains unchanged, and can thus be considered stable.

As a second example, suppose that t occurs 10 times in a collection of $1,000$ locations. The Probability of observing t in c would again be 0.01. Again, we are adding a document to the collection, which consists of 100 locations, and where t again occurs eight times. Here, the probability $P(t|c)$ changes to 0.0163634. This change of $P(t|c)$ could make a difference at retrieval time; we would therefore not consider the probability stable.

While it would be possible to counter the instability of probability estimates by adjusting the weights, this cannot be done as easily as for the reliability of evidence. Where the reliability of evidence is a measure which could be aggregated by a system over time, or could be derived from aggregating the reliability scores of a number of systems, the stability of a probabilities would have to be reported by the sources of evidence themselves. Highly reliable evidence sources would thus receive a discounted weight, whereas less reliable sources might receive a boosted weight for more stable statistics. However, this leads to the question of how reliable a reported stable statistic from an unreliable source is. The complexities arising from these problems would warrant a much closer investigation, which is out of scope for this dissertation.

The problems mentioned suggest that term probability augmentation should only be carried out where a) sources of evidence are reliable, or reliability does not matter, and b) probabilities are stable, or stability does not matter.

4.1.2 Term-Frequency Propagation

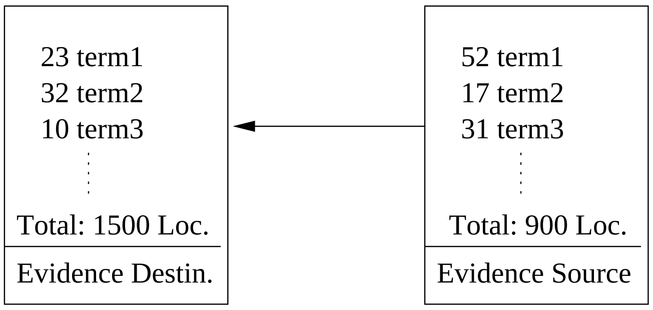


Figure 4.4: Term-frequency propagation: frequency evidence is pushed from the source to the destination.

The second type of evidence propagation, the term *frequency* propagation, overcomes some of the problems mentioned for propagating term probabilities. Here, instead of propagating the probabilities of terms as derived from the frequencies, the actual frequencies serve as evidence to augment probabilities at the evidence propagation destination. An example of this is shown in Figure 4.4.

Suppose that we are again interested in the linear occurrence probability $P(t|d_{AD})$, which is estimated as the number of locations $n_L(t,d)$ of term t divided by the total number of locations $N_L(d)$. For the given example, we can estimate the probability as

$$P(t|d_{AD}) \quad := \quad \frac{n_L(d_D,t)+n_L(d_S,t)}{N_L(d_D)+N_L(d_S)} \tag{4.3}$$

or, following the discussion of probability propagation, can estimate the general probability $P(t|d_{AD})$ for n sources of evidence as

$$P(t|d_{AD}) \quad := \quad \frac{n_L(t,d_D)+\sum_{1\cdots n} n_L(t,e_i)}{N_L(d_D)+\sum_{1\cdots n} N_L(e_i)} \tag{4.4}$$

Figure 4.5 shows the implementation of a frequency-based term augmentation strategy in Probabilistic Datalog.

```
aug_term(T,D) :- freq_term (T,D).
aug_term(T,D) :- link (D,D2) & freq_term(T,D2).
freq_aug_term  DISJOINT(T,D) :- aug_term(T,D).

aug_loc (D) :- locations (D).
aug_loc (D) :- link (D,D2) & locations (D2).
freq_aug_loc  DISJOINT(D) :- aug_log(D).

w_aug_term(T,D) :- freq_aug_term (T,D) & aug_loc(D) | (D).
```

Figure 4.5: Probabilistic Datalog expression to perform frequency-based term augmentation.

Augmenting term probabilities based on frequencies in this way overcomes the problem of assessing the stability of probabilities: since the final, augmented estimate is derived directly from the total observed term occurrences, sparse evidence – for which probabilistic augmentation would have provided an unstable estimate – has only a small impact on the overall term probabilities.

The problem of unreliable evidence sources is still present in the frequency propagation scheme, however, it can still be mitigated by weighting frequencies according to a source's reliability.

In retrieval tasks where evidence propagation is used, both augmentation strategies can usually be used interchangeably. For the remainder of this chapter, we will only refer to the more general concept of knowledge augmentation and evidence propagation, without providing details as to which specific strategy is used.

With the actual evidence propagation operations laid out, we can now turn to information retrieval tasks which use evidence propagation and knowledge augmentation to rank elements.

4.2 Summarisation

We will start the discussion of augmentation in summarisation by taking a high abstraction point of view of summarisation. For the purpose of summarisation, a document is a concatenation of subelements, a subset of which would provide a good summary of the document. For example, Figure 4.6 depicts a document which is a concatenation of sentences (labelled 1 to *n*). The set of all sentences forms the entire document, while a specific subset (or specific subsets, as there might be more than one) form good summaries of the document.

Here, evidence and knowledge is pushed from the sentences to the entire document. More

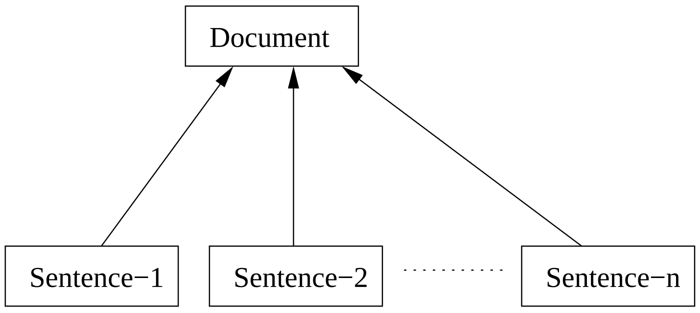


Figure 4.6: Summarisation: evidence provided by the sentences is propagated to the entire document. Summarisation restricts the document tree to the most highly contributing sentences.

generally, we might say that evidence and knowledge is pushed from the potentially summary-forming elements to the context to be summarised (a point we will discuss in greater detail in later chapters). The task of automatic summarisation is then to identify the most important concepts (i.e. items of evidence / knowledge), and to select those sentences which most highly contribute to them.

To illustrate, we will discuss one way to achieve these two goals. To identify the most important concepts discussed in a document, one could count the number of evidence sources which contribute to the item of knowledge in the evidence destination. For the example shown in Figure 4.6, this would reduce to counting the number of sentences which contribute to knowledge items in the document. Those items which are contributed to by the most sentences would regarded as the most salient concepts discussed in the document.

As a second step, the set of sentences to form the summary needs to be found. Again, we will provide one exemplary way of achieving this. Suppose that we have identified the most salient concepts in a document, in the way outlined above. To select the set of sentences to form the summary, we could prune the evidence propagation tree, by removing all nodes which do not contribute evidence or knowledge to the most salient concepts, or only provide very sparse evidence. The summary would then consist of those sentences which are the most significant sources of evidence for the most salient concepts.

We argue that document summarisation is similar to other retrieval tasks from a conceptual point of view, in that they all share evidence propagation and knowledge augmentation schemes (as shown in Figure 4.1). In all the retrieval task discussed in this chapter, the retrieval models incorporate outside evidence (i.e. evidence not found in the document or other retrieval element to be processed itself) into processes such as probability estimation or modelling retrieval models. To show this similarity, we now compare the abstract view of summarisation developed previously to high abstraction views of three other retrieval tasks: anchor-text retrieval, website partitioning, and structured document retrieval.

4.3 Anchor-text Retrieval

Anchor-text retrieval is a link-based ranking method, where evidence from hyperlinks is propagated to documents (Craswell et al. (2001a)). A hyperlink is a relationship between two documents, such that the link is placed at the evidence source, and provides a description of the evidence destination. This destination description is called *anchor text*. For example, a web page linking to the main Queen Mary, University of London web page would contain in its HTML code a hyperlink of the following form:

```
<A HREF="http://www.qmul.ac.uk">Queen Mary, University of London</A>
```

Here, `http://www.qmul.ac.uk` is the location of the link destination, and `Queen Mary, University of London` is the anchor text. Only the anchor text is displayed in the source document. When a user clicks on the anchor, the destination page is displayed. Anchor-text retrieval attempts to exploit the information provided by the anchor-text, by incorporating this evidence into link destinations when ranking documents.

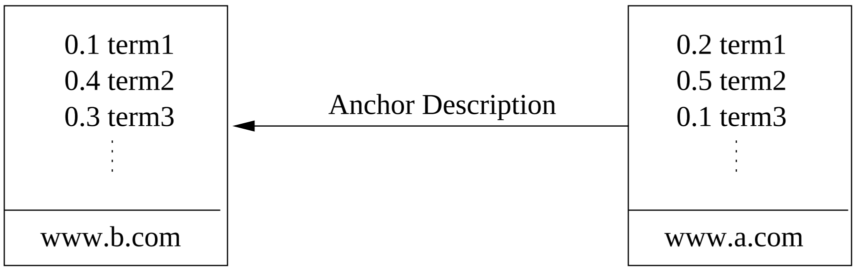


Figure 4.7: Anchor-text retrieval: evidence provided by the anchor-text is propagated to the evidence destination “www.b.com”.

For example, Figure 4.7 depicts the situation were a page `www.a.com` links to a page `www.b.com`, using the anchor-text “Anchor Description”. The link destination “www.b.com” does not contain either of the two terms, so normally it would not be retrieved in response to a query with those terms. However, via the mechanism of evidence propagation, both terms are added to the representation of “www.b.com”, such that it would be retrieved for a corresponding query.

Anchor-text retrieval is a more specific form of general link-based retrieval methods, and as such only makes use of the *anchor description* assumption (ignoring both the *recommendation* *assumption* (Brin and Page (1998); Kleinberg (1999)) and the *topic locality assumption* (Davison (2000)) used by general link-based retrieval methods). The anchor description assumption states that the anchor-text of a link describes its target. Using the earlier example, the text "Queen Mary, University of London" is a description of the destination `www.qmul.ac.uk`. Assuming correctness of the anchor description assumption, it is reasonable to treat terms in the anchor-text as belonging to the link destination, rather than (or in addition to) the link source (Brin and Page (1998); McBryan (1994)).

Although link-based retrieval methods and the propagation of anchor-text evidence seem reasonable to improve the quality of retrieval, results of experimental evaluations have been mixed. Experiments on the TREC web-track (Hawking et al. (1999)) have found that link-based retrieval methods performed not better than non link-based methods for search tasks. However, Craswell *et al.* report significant effectiveness improvements for the task of website finding when comparing an anchor-text retrieval method to a non link-based method (Craswell et al. (2001a)).

Recently, research on anchor-text retrieval has expanded into non web-IR retrieval scenarios. Wu *et al.* report on the application of link-based retrieval in the context of book-search, where the task is to find best entry points in books for specific queries (Craswell et al. (2001a); Wu et al. (2008)). In their work, Wu *et al.* attempt to improve book search by propagating back-of-book-index (BOBI) information to the pages linked by the index. However, they argue that a "naive" propagation of term information would not result in significant changes in term probabilities, as the BOBI would provide very sparse information. Instead, they develop a more advanced "voter" model. Unfortunately, no actual performance figures are reported.

4.4 Website Partitioning

The field of website partitioning is concerned with detecting the constituent blocks of a webpage (such as text spans, or a group of input boxes), and to determine which blocks best describe the website, or are most relevant to a given user query. Website partitioning could thus also be considered to be a subclass of structured document retrieval, where the structure is not explicitly available via element tags, but is instead hidden and only conveyed visually.

While the actual process of determining the best decomposition of a web page is beyond the scope of this dissertation, Fersini *et al.* propose a term-weighting scheme for terms in visual blocks where the weight of a term in one block is determined by the occurrence of the term in

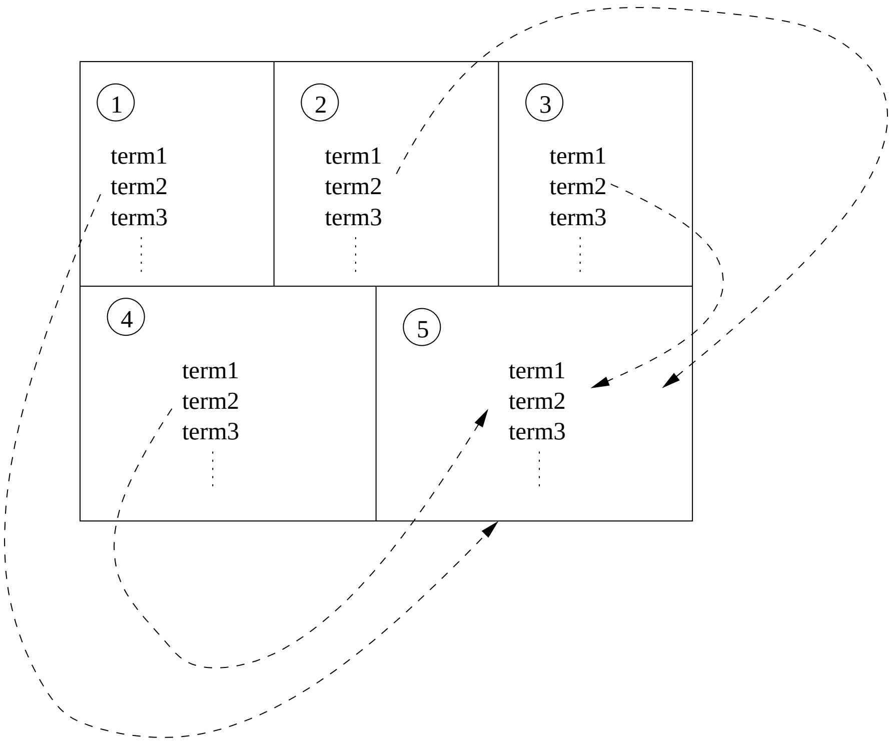


Figure 4.8: Web partitioning: Term weights in block 5 are influenced by term frequencies in other blocks.

other blocks (Fersini et al. (2008); Cai et al. (2003)). This, again, can be considered an instance of evidence propagation. In their work, Fersini *et al.* propose to retrieve visual blocks based on a modified TF-IDF retrieval approach, where the TF component takes into account not only the occurrence of terms in the block at hand, but reweights the terms according to their presence in other, important, visual blocks. This so called *Image Weighted Term Frequency* IWTF is substituted for the traditional TF in the retrieval approach. The IWTF is a combination of the traditional TF weight, and a term importance score TI, which is computed as the inverse of occurrences of a term in other blocks, ITI: if a term occurs frequently in other blocks, it is considered important, whereas a term which only occurs in one block is considered less important. The ITI of a term t in visual block k of document j is defined as

$$ITI_{ikj} := \frac{\bar{\sigma}_{ikj}}{\bar{\gamma}_{ij} - \bar{\sigma}_{ikj}} \tag{4.5}$$

where $\bar{\sigma}_{ikj}$ is the number of occurrences of t in k, and $\bar{\gamma}_{ij}$ is the number of occurrences of t in the entire document j[1].

[1] A somewhat problematic formulation: a term which only occurs in one block would have an undefined ITI, as a division by zero would occur.

The TI of a term t_i in a document d_j is then given as

$$TI(t_i, d_j) \quad := \quad \max_k \{ITI_{ikj}^{-1}\} \tag{4.6}$$

Here, evidence in the form of term frequencies is propagated between visual blocks. Figure 4.8 depicts the propagation of evidence for the case of five blocks, where blocks one to four influence the term weights in block five.

4.5 Structured Document Retrieval

Structured document retrieval (SDR) is a comparatively recent addition to the family of information retrieval tasks, and was largely motivated by the specification of the XML markup language (Malik et al. (2006)). SDR differs from traditional IR ad-hoc retrieval in that retrieval does not always retrieve entire documents, but also smaller document elements. This retrieval task – also known as *focused retrieval*, is not only concerned with retrieving relevant elements, but also elements which are most specific for a given query, i.e. elements which contain as little information not pertaining to the user query as possible.

Knowledge augmentation and evidence propagation are of central concern in SDR. We will illustrate this with a sample document structure, depicted in Figure 4.9.

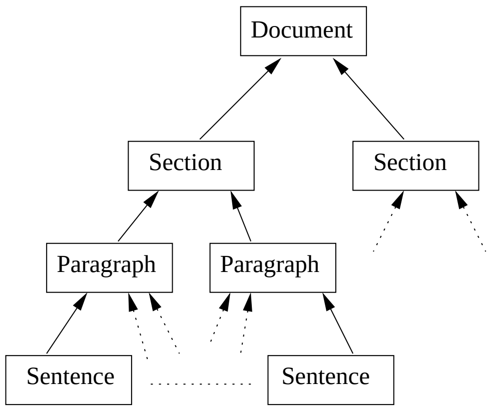


Figure 4.9: Structured Document Retrieval: Knowledge of subelements is pushed up the document tree, and augments the knowledge of parent nodes.

In a well formed and well balanced structured document, the actual textual content will only be present at the leaf nodes. In Figure 4.9, text would thus only be present in the sentences. However, sentences in isolation might no be judged relevant for a given user query, whereas a composition of sentences, such as a paragraph, might be both relevant and specific. However, paragraphs do not have knowledge of their own (i.e. do not contain textual content), as it would violate the well-formedness of the document. Knowledge of paragraphs can thus only be acquired by a process of knowledge augmentation, by pushing evidence from child-nodes up the document hierarchy.

The knowledge of any non-leaf node is thus a composition of the knowledge of its childnodes. SDR is very similar to anchor-text retrieval in this respect, where childnodes correspond to link-sources, parentnodes correspond to link destinations, and the textual content of nodes (their knowledge) corresponds to the anchor text.

The evidence propagation carried out for structured document retrieval similarly applies to the summarisation of structured documents, and is therefore carried out for POLIS as well. We will discuss these aspects in greater detail in Chapters 5 and 6.3.

4.6 Conclusion

This chapter highlights the parallelisms between summarisation and other information retrieval tasks, based on the two related concepts of knowledge augmentation and evidence propagation. From the perspective of knowledge augmentation and evidence propagation, summarisation sits alongside structured document retrieval (which pertains to the use of structure by POLIS), Anchor-text retrieval, and website-partitioning.

All four retrieval tasks incorporate information from other items in the domain of interest into the retrieval process, beyond the derivation of simple collection statistics as found in ad-hoc retrieval. However, the tasks differently restrict the sources of additional knowledge to be included in the retrieval process. Both summarisation and structured document retrieval consider a tree-like hierarchy of items, in which knowledge is only propagated amongst nodes on the same branch. There is thus a strict subpart-superpart relationship between the knowledge source and the knowledge destination, and knowledge will usually propagate from subpart to superpart (however, this is not a strict requirement). Websites, as processed by website-partitioning approaches, could be considered to be a subclass of structured documents, however, since content of a website could be linked-in from another website, a set of websites is more similar to a graph than a set of trees. This also holds certainly true for the case of anchor-text retrieval, where hyperlinks between documents form a directed graph.

The purpose of comparing summarisation and the other retrieval tasks discussed in this chapter is twofold. The summarisation models implemented by POLIS are derived from traditional retrieval tasks used in ad-hoc retrieval. By showing the similarity between summarisation and tasks such as structured document retrieval, we provide a rationale for developing these summarisation models.

Secondly, the parallels developed in this chapter all rely on the process of knowledge augmentation and evidence propagation. The same process is also found in both the POLIS summarisation models and the semantics we develop for POLIS in Chapter 6.3. This chapter thus motivates the summarisation models developed in Chapter 5, and lays the foundations for the POLIS semantics.

Chapter 5

Summarisation Models

5.1 Introduction

The overall aim of the work discussed in this dissertation is the development of a summarisation logic, which provides an abstraction layer to the task of document summarisation. We will start the discussion on how to develop such a logic in this chapter, by looking at probabilistic models for summarisation.

Document summarisation, both singe- and multi-document summarisation, is concerned with identifying the most salient concepts (e.g. sentences) in the set of concepts to be summarised. To postulate the existence of most salient concepts implies that there are also less salient concepts, such that it becomes possible to order concepts by salience. In such an ordering, the highest ranking concepts would be most salient (or most important), and should be included in a summary, while lower ranking concepts are not as important, and can therefore be disregarded when generating a summary.

To arrive at this ordering, we first need to define a function who's domain is a set of concepts (namely those to be summarised), and who's range is a set of concepts together with a numerical value assigned to each individual concept. This numerical value is an indication of the importance of a concept, and can be used to arrange concepts by importance. We will call this function the *summarisation model*.

The purpose of the summarisation model is to assign numerical scores to individual concepts for the task of arranging them by importance only; it is not supposed to "judge" concepts, assigning numerical "grades" based on how informative or important they are in absolute terms. To give a more concrete example, suppose a concept c_1 is assigned a score of 1, and a second concept c_2 is given a score $1,000$. This does not mean that c_2 is $1,000$ times as informative as c_1, but merely that it is *more* informative (or important) than c_1, and would be given a higher rank in a linear ordering of concepts.

In this chapter, we will develop several probabilistic summarisation models. Probabilistic models have a long history in information retrieval to model numerous retrieval tasks. For the most common information retrieval task, the ad-hoc document retrieval, probabilistic models have been used to retrieve the most relevant documents, by measuring the probability $P(d \rightarrow q)$ of a document *implying* a query.

In addition to ad-hoc retrieval, probabilistic models have been used as part of summarisation systems at various stages of the summarisation process, for example as a probabilistic Bayesian classifier (see Kupiec et al. (1995)) trained on selecting the most salient concepts , or to model anaphora resolution (see Ge et al. (1998)).

Here, we will use the aforementioned ad-hoc retrieval models as a basis for developing our own summarisation models. We will start the discussion by investigating the meaning of the probability $P(d \rightarrow q)$, and how it has been implemented by a number of ad-hoc retrieval models. We then investigate how these retrieval models can be modified to be applicable to document summarisation.

Section 5.2 opens the discussion by introducing traditional ad-hoc retrieval models, and shows how these retrieval models can be classified into disjunctive and conjunctive models. Section 5.3 discusses how sentences (or elements at any other granularity) can be ranked for summarisation according to the retrieval implication $d \rightarrow q$. This is followed by section 5.4, where we discuss both query-based and non query-based summarisation models implemented by POLIS. Section 5.5 concludes the discussion.

5.2 Retrieval Models

The purpose of any ad-hoc retrieval model is to rank a set of documents according to their relevance for a given query. In his seminal 1986 paper, Van Rijsbergen detailed that an estimation of the relevance of a document d for a given query q should aim to measure "the extent to which q might be inferred from d", expressed as $d \rightarrow q$ (van Rijsbergen (1986)). To reflect the uncertainty

		Probabilistic Term Combination	
		Disjunctive	Conjunctive
Task	Retrieval	Disjunctive retrieval models	Conjunctive summarisation models
	Summarisation	Disjunctive summarisation models	Conjunctive summarisation models

Figure 5.1: Disjunctive and conjunctive probabilistic models.

of the retrieval process, the implication from d to q does not provide a simple *true* or *false* as it would in a propositional logic, but instead yields a *probability* of implication, $\mathrm{P}(d \rightarrow q)$, which measures the extent to which d is relevant to q.

Following Van Rijsbergen, the probability $\mathrm{P}(d \rightarrow q)$ can be rewritten as the conditional probability $\mathrm{P}(q|d)$. This conditional probability is the foundation for almost all probabilistic ad-hoc retrieval models, and is estimated based on the occurrence of terms in documents and queries. While the number of retrieval models based on this conditional probability is fairly large, all of them are based on one of two assumptions for estimating the probability of observing a term in a document: a disjunctive interpretation, or a conjunctive interpretation. We will use the retrieval models developed here as the basis of our summarisation models (see Figure 5.1), and will successively fill the four cells in the figure in this chapter.

5.2.1 Disjunctive Combination of Terms

The first family of retrieval models treats terms in the term-space as disjunctive events. Members of this family of models are characterised by retrieval functions which combine evidence with respect to the query terms in a disjunctive fashion.

The most prominent members of this family are the retrieval models belonging to the family of TF-IDF ad-hoc retrieval models (Salton and Buckley (1988)). Although TF-IDF retrieval models in the traditional sense do not probabilistically model the term-space and the retrieval function, it is possible to interpret TF-IDF as a probabilistic retrieval function (Hiemstra (2000)); we will follow this interpretation here.

TF-IDF is a combination of two weights: the *term-frequency* TF, and the *inverse document* *frequency* IDF, which measures the *informativeness* of a term. The informativeness of a term is inversely proportional to the number of documents it occurs in: the fewer documents it occurs in, the more informative it is (Jones (1972)). The IDF value of a term t with respect to a collection c can therefore be estimated as (Robertson (2004)):

$$\text{idf}(t,c) \quad := \quad -\log\frac{n_D(t,c)}{N_D(c)} = -\log P_D(t|c) \tag{5.1}$$

where $n_D(t,c)$ is the number of documents in c which contain t, and $N_D(c)$ is the total number of documents in c. Estimating IDF as a fraction of $n_D(t,c)$ and $N_D(c)$ is equivalent to the inverse log of $P_D(t|c)$. In addition to measuring the informativeness of a term, IDF can also be seen as a measure of term discriminativeness, i.e. a measure of how well a term can discriminate between documents.

In this formulation, both TF and IDF are heuristically defined, and can take arbitrary values. However, it is also possible to derive a probabilistic formulation of TF-IDF from this heuristic one. In this probabilistic formulation, the term-frequency TF of a term t in document d is modelled by the probability of observing the term t in document d; TF is thus modelled as the conditional probability $P(t|d)$. There are several ways in which this conditional probability can be estimated, which we will discuss later in this chapter.

A probabilistic formulation if IDF is not as immediate as the formulation of TF. The formulation of $\text{idf}(t,c)$ shows that IDF can be modelled via the inverse logarithm $-\log P_D(t|c)$. The retrieval probability $P(d \rightarrow q)$, however, does not refer to the document collection c (this also holds for the rewritten formulation $P(q|d)$). However, rewriting this conditional probability can lead to a probabilistic IDF estimate using the collection c. It is possible to rewrite the conditional probability $P(q|d)$, assuming that terms are disjoint, as the two partial probabilities $P(t|d)$ and $P(t|q)$:

$$P(q|d) \quad = \quad \sum_t P(t|d)P(q|t) \tag{5.2}$$

Here, $P(t|d)$ follows the estimation shown for TF. The second probability $P(q|t)$ measures the probability of observing the query q (the *event* q) given the term t. If t only occurs in this query q, the probability $P(q|t)$ will be 1.0. However, with increasing occurrences of t in members of the set of all possible queries, the probability $P(q|t)$ decreases; $P(q|t)$ thus exhibits a measure of discriminativeness of terms also found in IDF. The same line of reasoning would also hold for a (hypothetical) probability $P(d|t)$, which would estimate the discriminativeness of a term t with respect to the set of all documents. More general, it would thus be possible to say that any probability $P(c|t)$ estimates the discriminativeness of a term t with respect to the collection c. In the TF-IDF model, $P(t|d)$ and $P(c|t)$ are combined *conjunctively*, over a *disjunctive* set of terms. The canonical version of TF-IDF is given as

$$\begin{aligned} \mathrm{RSV}(d,q) &:= \sum_{t \in q} \mathrm{TF}(t,d) \cdot \mathrm{IDF}(t,c) && (5.3) \\ &\Downarrow \\ P(q|d) &:= \sum_{t \in q} P(t|d) \cdot P(c|t) && (5.4) \end{aligned}$$

The disjunction of independent terms is modelled as the sum over all query terms, whereas the per-term evidence is the conjunction of $P(t|d)$ and $P(c|t)$. Variations of the TF-IDF model differ in their maximum likelihood estimate of the evidence probabilities. While the IDF component remains largely fixed, there are several ways in which the document based probability $P(t|d)$ can be estimated.

5.2.1.1 Linear Occurrence-based Estimates

The linear occurrence-based estimate of $P(t|d)$ is one of the simplest, and directly models the notion that $P(t|d)$ is based on the frequency with which a term occurs in a document. The linear estimate can be modelled by several strategies, common ones being the maximum likelihood estimate of observing a term, and an estimate based on the maximum term frequency.

The maximum likelihood probability of observing t in d can be estimated as the fraction of occurrences of t given the total number of locations in d. Formally,

$$P_{ML}(t|d) := \frac{n_L(t,d)}{N_L(d)} \qquad (5.5)$$

where $n_L(t,d)$ is the number of occurrences of t in d, and $N_L(d)$ is the total number of locations in d. However, estimating the probability $P(t|d)$ using this maximum likelihood estimate leads to a bias in favour of short documents: irrespective of the actual query, the probability for a short document to be ranked high is larger than the probability of a long document to be ranked high. One way to compensate for this bias is to include a document length normalisation into the estimation of the TF. This can either be done by including information about the actual length of the document (as is the case with the BM25 TF estimate, discussed below), or by normalising the per-term TF by the highest scoring term.

This latter TF estimate works by modelling the probability $P(t|d)$ as the frequency of t with respect to the most frequent term t_{max}. Whereas in the maximum likelihood estimate the probability of a rare term (such as a term occurring only once) would decrease for longer documents, its probability remains fixed with respect to the maximum frequency term in the maximum frequency estimate, assuming that longer document introduce a larger set of distinct terms, not just more occurrences of the same terms (e.g. Robertson and Walker (1994)). The actual estimate of $P_{max}(t|d)$ can be formalised as

$$P_{max}(t|d) \quad := \quad \frac{n_L(t,d)}{n_L(t_{max},d)} \tag{5.6}$$

where $n_L(t,d)$ is the number of occurrences of term t in document d, and $n_L(t_{max},d)$ is the number of occurrences of the most frequent term in d. Performance evaluations indicate that estimating $P(t|d)$ using the maximum frequency works better than the maximum likelihood estimate (Singhal et al. (1996)).

5.2.1.2 Non-linear BM25-like Estimates

All estimates for the probability of observing a term in a document outlined in the previous section were based on linear estimates. In this section, we will look at *non-linear* estimates. Non-linear estimates are associated with the concept of "boosting" the probability of observing rare terms in documents, thus narrowing the range of observed probabilities.

The use of non-linear TF estimates was popularised by the Okapi BM25 weighting function, which included a boosting function based on the fraction of a document's length with respect to the average document length (Robertson et al. (1992)). Robertson *et al.* motivate the use of such a non-linear term-probability estimates with the amount of information gained from observing a term: observing a term for the first time yields greater information than any subsequent observations (Robertson et al. (2004)). In BM25, the probability $P_{BM25}(t|d)$ is estimated as

$$P_{BM25}(t|d) \quad := \quad \frac{n_L(t,d)}{n_L(t,d)+K} \tag{5.7}$$

where $n_L(t,d)$ is the number of occurrences of t in d, and K is the boosting function. The value of K is set such that it is proportional to the fraction of the length of d and the average document length

$$K \quad \propto \quad \frac{N_L(d)}{\text{avgdl}(c)}, \tag{5.8}$$

and can be formally given as

$$K \quad := \quad k_1 \times \left((1-b) + b \times \frac{N_L(d)}{\text{avgdl}(c)} \right). \tag{5.9}$$

Here, $N_L(d)$ is the length of document d, and avgdl(c) is the average length of documents in the collection c; k_1 and b are free parameters, and are commonly reported to be set to 2.0 and 0.75, respectively. One advantage of modelling $P(t|d)$ using this approach is the inclusion of a document-length normalisation directly into the probability estimation process. A disadvantage of modelling $P(t|d)$ in this way is that it is necessary to obtain collection statistics before the probability can be estimated. To overcome this problem, it is possible to combine the non-linear estimate of BM25 with a within-document term statistic only, not using collection statistics. In this approach, the K of BM25 is replaced with the *average* frequency with which terms occur in a document. Using this estimate, it is possible to relate the TF-IDF retrieval model to the Poisson retrieval model (Roelleke and Wang (2006)); we therefore refer to the resulting estimate as $P_{Poisson}(t|d)$. The estimate can formally be given as:

$$P_{Poisson}(t|d) \quad := \quad \frac{n_L(t,d)}{n_L(t,d) + n_L(t_{avg},d)} \tag{5.10}$$

where $n_L(t,d)$ is the number of occurrences of t in d, and $n_L(t_{avg},d)$ is the average number of occurrences of a term in d.

5.2.2 Conjunctive Combination of Terms

The second family of retrieval models is characterised by treating terms in the term-space as conjunctive events. We will focus on the most prominent of these models, the Language Modelling ad-hoc retrieval model (Ponte and Croft (1998); Croft and Lafferty (2003)). The Language Modelling retrieval model is inspired by research in the field of natural language processing, such as speech recognition. Where retrieval models such as TF-IDF or BIR attempt to predict the probability that a document is relevant for a given query, LM tries to predict how likely a document *generated* the given query. For a query q and a document d, the canonical model of Language Modelling can be formulated as

$$P(q|d) \quad := \quad \prod_{t \in q} P(t|d) \tag{5.11}$$

where $P(q|d)$ is the probability that d generates q, and $P(t|d)$ is the likelihood of observing t in d. The model thus combines individual term probabilities in a conjunctive fashion.

Language Modelling assumes that for every document, a *document model* exists, which produces terms with a certain probability. The actual document itself is generated by repeatedly sampling the document model. The probability $P(q|d)$ is approximated by estimating the likelihood that the document model generated the query q. However, the document model itself is inaccessible, and the likelihood that a model will generate a specific term has to be estimated from the given document (which is an incomplete sample of the document model). To counter zero probabilities for terms not observed in the document (which *might* have been produced by the document model, if the sample – the document – would have been longer), Language Modelling incorporates a *background model* into the estimation of $P(q|d)$, which is the likelihood of observing the term when sampling any random document model in the collection.

The ranking probability $P(q|d)$ is thus estimated using two sources of evidence: the term-document probability $P(t|d)$, and the term-collection probability $P(t|c)$. Both $P(t|d)$ and $P(t|c)$ are ordinarily estimated as linear, occurrence-based measures. To combine the two sources of evidence, LM disjunctively combines both probabilities using a *smoothing method*. Several such smoothing methods exists (see (Zhai and Lafferty (2001)) for an overview and a performance evaluation); here, we will only focus on the Jelinek-Mercer smoother, which estimates $P(q|d)$ as a weighted linear combination of $P(t|d)$ and $P(t|c)$, using a weighting parameter λ. The Language Modelling retrieval function using the Jelinek-Mercer smoother can be formally stated as:

$$P(q|d) \quad := \quad \prod_{t \in q} (1-\lambda) \cdot P(t|d) + \lambda \cdot P(t|c) \tag{5.12}$$

While Language Modelling estimates the probability $P(q|d)$ via a conjunction of independent terms, the evidence is combined disjunctively. This is evident for the Jelinek-Mercer smoother, but also holds for other smoothing operations, such as Dirichlet, or Absolute Discounting.

5.3 Summary-Element Weighting by $P(d \rightarrow q)$

In the previous section we looked at ways in which traditional ad-hoc retrieval models estimate the probability $P(q|d)$, and thus ultimately the probability $P(d \rightarrow q)$. Before we can start do derive summarisation models from these ad-hoc retrieval models, we need to look at the way in which the implication probability $P(d \rightarrow q)$ can be interpreted in the context of document summarisation.

While it is possible to reformulate $P(d \rightarrow q)$ as $P(q|d)$ to ease the implementation of models estimating it, it is more "intuitive" to reason about the retrieval process in terms of an *implication* from a document d to a query q, than to consider the conditional probability of a query q *given* a document d. A document should be ranked high if it strongly implies the query; it should be ranked high if it is "about" the query. We can extend this intuitive notion of the relevance based ranking of documents to a relevance based ranking of summary elements, by saying that such elements should be as much "about" the source as possible. For example, when summarising a single document by extracting sentences from the document, the extracted sentences should be as much "about" the document as possible. However, this definition of "aboutness" should not be limited to single documents. The example would be equally valid for multi document summarisation, where sentences should be as much "about" the *set* of documents as possible.

We therefore need to find more generalised expressions for such concepts as "sentence" or "document". The purpose of a summarisation logic such as POLIS is to provide an abstraction layer to the task of summarisation, and to allow for more flexible strategies for the summary generation. Summaries will therefore not necessarily be build from sentences, but from textual material at any possible level of granularity. To accommodate this flexibility in our discussion of summarisation models, we will refer to these text-extracts as *elements*. A sentence is an element, however, an element could also be a paragraph, or – suitable document markup permitting – sub-sentence fragments.

Similarly, a summarisation logic should be able to perform both single- and multi-document summarisation. For our discussion, we thus need a suitable term to address both a single document in single-document summarisation, and a set of documents in multi-document summarisation. We will use the term *context* in our discussion of summarisation models to refer to the "data" to be summarised; in single-document summarisation, the context would be the single document, while in multi-document summarisation, it would be the set of documents to be summarised.

With this terminology, we can now give a conceptual definition as to how a summariser should rank elements. While ad-hoc retrieval models should rank documents according to their "aboutness" for a given query, summarisation models should rank elements according to their "aboutness" with respect to their context. Following the idea that documents should be ranked according to the degree to which they imply a query, elements should be ranked according to the degree to which they imply their context: $P(e \rightarrow c)$. Under this interpretation, elements take the role of documents for the purpose of ranking, while the context takes the role of the query. Following Van Rijsbergen, the probability $P(e \rightarrow c)$ can thus be rewritten as the conditional probability $P(c|e)$.

However, in addition to the distinction between single- and multi-document summarisation, automatic summariser can also be classified as either generic or query-based, where in the latter case the weighting of elements should additionally based on a given query. Again using a conditional probability as an importance measure, we could incorporate a query q into the formulation either as a source of evidence, resulting in the conditional probability $P(c|e,q)$, or as an additional event conditioned on an element, resulting in the conditional probability $P(c,q|e)$. In the next section, we will look at ways in which the conceptual summarisation model $P(c|e)$ and its query-based variant can be implemented by reformulating the outlined ad-hoc retrieval models.

5.4 Summarisation Models

To develop our summarisation models, we will again distinguish between a disjunctive and a conjunctive collection of evidence in the form of term probabilities. We will also discuss the shortcomings of the traditional retrieval models for application to summarisation, and present possible solutions, leading to new, ad-hoc retrieval inspired summarisation models.

5.4.1 Non Query-based Models

The first class of summarisation models we look at are the so called non query-based or generic models. These models do not consider any external evidence (such as a query) for estimating the salience of an element, and instead rely entirely on the context and element evidence.

5.4.1.1 Disjunctive Models

We mentioned in the background section on disjunctive TF-IDF models that the canonical form of such models is

$$P(q|d) \quad := \quad \sum_{t \in q} P(t|d) \cdot P(q|t) \tag{5.13}$$

where $P(q|t)$ is an informativeness probability rather than an occurrence based probability. We discussed on page 76 that the informativeness probability $P(q|t)$ can also be captured by the more general notation $P(c|t)$, by which we can express the TF-IDF models as

$$P(q|d) \quad := \quad \sum_{t \in q} P(t|d) \cdot P(c|t) \tag{5.14}$$

By substituting e for d, and c for q, we would arrive at the first summarisation model:

$$P(c|e) \quad := \quad \sum_{t \in c} P(t|e) \cdot P(X|t) \tag{5.15}$$

However, this formulation is problematic with respect to the second probabilistic estimate, the informativeness based probability $P(X|t)$. For classical ad-hoc retrieval, the X in the second conditional probability would be the collection c, so that the probability provides a collection statistic of the form $P(c|t)$.

The probability $P(c|t)$ is a discriminativeness measure to give higher weights to documents (here: elements) containing informative terms, and relies on a well defined term-space to be meaningful. A typical document collection for ad-hoc retrieval will fulfil these requirements by providing a heterogeneous document collection with varying topics, and a sample space large enough to acquire meaningful probabilities. These two properties typically do not hold in the context of summarisation.

Firstly, the context to be summarised will either consist of a single document (which by definition is not heterogeneous), or will be made up of documents concerned with a common topic, so that divergence between documents is smaller than for a collection of documents covering multiple topics. Secondly, the size of the context to be summarised will be smaller than a typical document collection by several orders of magnitude, so that informativeness or discriminativeness statistics will be sparse and not as reliable as they are for full collections.

Another problem with using the traditional IDF in the context summarisation lies in the independence of elements and collections. In a collection for ad-hoc retrieval, documents are truly independent from each other, and *collection* is a construct that only comes into existence by the definition that a set of documents *is* a collection. However, for summarisation, all elements are *actual* subparts of the context to be summarised, not merely information carriers contained in a random grouping. This dependence between elements and contexts alone puts the strong discriminativeness of IDF in question.

To overcome these shortcomings associated with strong discriminativeness of the IDF, we propose to rank elements by a ranking model that uses a "soft" discriminativeness, which is less dependent on the assumptions outlined for the IDF. We model this "soft" discriminativeness with a sublinear mixture of the element term frequency $P(t|e)$, and the context term frequency $P(t|c)$. This sublinear mixture is modelled on the TF component of the BM25 retrieval model (Robertson et al. (1992)), where the probability of a term in a document is estimated using a document length normalisation based on a sublinear mixture of the form

$$P(t|d) \quad := \quad \frac{n_L(t,d)}{n_L(t,d)+K} \tag{5.16}$$

where K is a normalisation factor proportional to the fraction of document length and average document length (i.e. $K \propto \frac{N_L(D)}{\text{avgdl}(c)}$).

To estimate our "soft" discriminativeness, we combine $P(t|e)$ and $P(t|c)$ in a similar mixture, where the context probability $P(t|c)$ is inserted in position of the normalisation parameter K. We start by observing that the probability $P(c|e)$ can be estimated as a combination over all terms in both e and c:

$$P(c|e) = \sum_t P(t|e,c) \tag{5.17}$$

However, we now need to estimate the probability $P(t|e,c)$. To do so, we propose to combine $P(t|e)$ and $P(t|c)$ in the nonlinear mixture found in the BM25 model. We thus arrive at the final estimate for $P(c|e)$:

$$P(c|e) \quad := \quad \sum_{t \in e} \frac{P(t|e)}{P(t|e)+P(t|c)} \tag{5.18}$$

Here, the collection probability $P(t|c)$ acts as a boost for term probabilities, such that terms with a low element probability and a low context probability receive a strong boost, whereas the boost levels off for terms which are prominent among many elements. The collection probability $P(t|c)$ thus acts like an informativeness measure, by giving a higher probability to terms which are rare among all elements of a context.

This ranking model will be our *base ranking model*, which we will also use as a basis for one of the query-based models in the next section.

5.4.1.2 Conjunctive Models

Similarly to the disjunctive models, we can translate the conjunctive Language Modelling approach to be applicable to summarisation. Again starting from the canonical model, we can derive a conjunctive summarisation model:

$$P(c|e) \quad := \quad \prod_{t \in c} P(t|e) \tag{5.19}$$

Including a background model to account for the case of zero probabilities, and using a Jelinek-Mercer smoother, a summarisation model based on the Language Modelling retrieval model can be given as:

$$P(c|e) \quad := \quad \prod_{t \in c} (1-\lambda) \cdot P(t|e) + \lambda \cdot P(t|X) \tag{5.20}$$

However, here the lack of a proper collection statistic is even more pronounced than for the disjunctive models. The ad-hoc retrieval model distinguished between three sources of evidence: the query, the documents, and the collection. If we assume that contexts take the role of a query for summarisation, it would be highly problematic to also use context statistics as a background model. Completely disregarding the background model, however, would lead to a zero probability problem in cases where an element does not contain a term present in the context. Since it is highly unlikely that every element contains all terms present in the larger context, this would be the case for almost any element.

Additionally, the problems with respect to the heterogeneity and size of the sample space mentioned for the disjunctive models also hold here. We therefore did not try to derive a non query-based summarisation model from the conjunctive ad-hoc retrieval models.

5.4.2 Query-based Models

As a second class of summarisation models, we will now investigate the inclusion of queries into the summarisation process. The terminology for these models varies (commonly used are "query-based", "query-biased", "user-centered"), we will call them *query-based* summarisation models here. The generic, non query-based models discussed in the previous section estimate the importance of an element e by modelling the conditional probability $P(c|e)$. The query-based models developed here additionally include a query q into the estimation process, such that an element is not only "about" its context, but also "about" the given query.

5.4.2.1 $P(c|q,e)$ or $P(c,q|e)$?

The non query-based models estimate the importance of an element e by modelling the conditional probability of observing the context c given e: $P(c|e)$. To include a query bias, the query q needs to be included in the estimation process. However, this can be done in (at least) two ways: q could be either included as evidence, such that elements are ranked by the probability $P(c|q,e)$, or q could be modelled as an event conditioned on e, such that the ranking probability is $P(c,q|e)$.

Following our approach of modelling summarisation on ad-hoc retrieval, it seems more reasonable to assume that q should be conditioned on observing e: ad-hoc retrieval ranks documents by the conditional probability $P(q|d)$, so summarisation should rank elements by the probability $P(q|e)$. Query-biased summaries are thus the non query-based "aboutness" probability $P(c|e)$ in conjunction with the query probability $P(q|e)$. On the other hand, an estimate of the form $P(c|q,e)$ would indicate a ranking of the query with respect to the context.

The query-based summarisation models discussed in the remainder of this section therefore estimate the importance of an element by modelling the probability $P(c,q|e)$. We will develop two query-based models: a macro-averaging model, which estimates $P(c|e)$ and $P(q|e)$ independently and combines both probabilities to estimate $P(c,q|e)$, and a micro-averaging model, which estimates $P(c,q|e)$ in an integrated fashion.

5.4.2.2 Macroaveraging Model

The macroaveraging model is an additive model, where the ranking probability $P(c,q|e)$ is modelled by estimating the two probabilities $P(c|e)$ and $P(q|e)$ independently, and combining them in a suitable way. The partial probability $P(c|e)$ is identical to the *base model* we presented in the previous section, and can therefore be estimated using the base model.

The second partial probability, $P(q|e)$, is an instance of ad-hoc retrieval, where elements take the role of documents. Again, the small size of the sample space limits the accuracy of informativeness probabilities, however, here there is no overlap in the definition of query and context, as was the case for the conjunctive non query-based model. The probability $P(q|e)$ can therefore be estimated by any ad-hoc retrieval algorithm seen fit.

Both probabilities are combined to give an overall estimate of the ranking probability $P(c,q|e)$.

The canonical form of the macroaveraging model can thus be given as

$$P(c,q|e) \quad := \quad P(c|e) \oplus P(q|e) \tag{5.21}$$

where $P(c|e)$ is estimated using the summarisation base model, $P(q|e)$ is estimated using any suitable ad-hoc retrieval model, such as TF-IDF or Language Modelling, and $\oplus$ is a suitable function to combine $P(c|e)$ and $P(q|e)$. The combination function $\oplus$ could be modelled using a weighted linear addition, or, interpreting both $P(c|e)$ and $P(q|e)$ as independent events, could be based on a probabilistic OR, using the inclusion-exclusion principle.

We call this model the *macroaveraging model*, as both $P(c|e)$ and $P(q|e)$ are estimated independently, and are combined only on the level of individual elements. Information about individual term probabilities is lost in the process. We will therefore now investigate a second model, where individual term probabilities are retained.

5.4.2.3 Microaveraging Model

Our second query-based summarisation model, the microaveraging model, does not rely on the non query-based base model for estimating $P(c|e)$. Instead, the query-based probability $P(c,q|e)$ is estimated using an integrated model, based on individual term probabilities. This estimate relies on the theorem of total probability. Using the Total Probability Theorem, we interpret a context c as an event, the probability of which can be estimated by a series of partial conditional probabilities, conditioned on the individual terms in the term space.

An initial decomposition on the term space, using the theorem of total probability, thus allows us to rewrite $P(c,q|e)$ as

$$P(c,q|e) = \sum_{t \in e} P(c,q|t) \cdot P(t|e) \tag{5.22}$$

This decomposition requires an estimation of the (unknown) probability $P(c,q|t)$, which could be based on a weighted linear combination of the two probabilities $P(c|t)$ and $P(q|t)$, such that the overall formulation for $P(c,q|e)$ could be defined as

$$P(c,q|e) = \sum_{t \in e} \lambda \cdot (P(c|t) + (1-\lambda) \cdot P(q|t)) \cdot P(t|e) \tag{5.23}$$

This decomposition, however, is problematic for the aim of ranking elements with respect to a query and the context. Here, both $P(c|t)$ and $P(q|t)$ – which could be interpreted as a linear occurrence probability, or an informativeness probability – are static for all elements with respect to the context and a given query, such that a ranking of elements would only be based on the within-element probability $P(t|e)$. Instead, it would be more suitable to have a static informativeness probability derived from the context, and a ranking of elements according to similarity with the query.

We therefore propose to estimate the ranking probability $P(c,q|e)$ using two partial, conditional probabilities, $P(c|t)$ and $P(t|e,q)$, which range over all terms in an element e. The probability $P(c,q|e)$ is thus estimated as

$$P(c,q|e) \quad := \quad \sum_{t \in e} P(c|t) \cdot P(t|e,q). \tag{5.24}$$

However, there are two problems with this formulation. Firstly, the probability $P(c|t)$ could be interpreted as either an occurrence probability, or as an informativeness probability. If $P(c|t)$ is estimated as a linear occurrence probability, its value will always be 1.0, as summarisation in POLIS is based on a single context to be summarised. While this seems like a major defect in the formulation, it has a surprisingly limited effect on the performance of the model (see Chapter 7). It would also be possible to interpret $P(c|t)$ as an informativeness probability, with the characteristics of the traditional IDF formulation. The resulting model would be a microaveraging model with an added IDF weight (i.e. a "microaveraging-IDF"), where the IDF weight of a term with respect to the context is determined by the number of elements the term occurs in: a term occurring in many elements receives a low score, while a rare term is given a higher weight.

Secondly, information about the probability $P(t|e,q)$ is not directly observable, and needs to be estimate in an appropriate way. The problem of estimating the joint conditional probability of two events given the individual conditional events is fairly common in the context of Bayesian Inference Networks. One possibility to estimate the joint probability is to combine the partial probabilities in a suitable way. The joint conditional probability $P(t|e,q)$ can thus be estimated as

$$P(t|e,q) \quad := \quad P(t|e) \oplus P(t|q) \tag{5.25}$$

where $P(t|e)$ and $P(t|q)$ are the two partial probabilities of observing a term t in an element e or the query q, respectively, and $\oplus$ is a suitable combination function, such as a weighted linear combination, or, viewing terms as events, a combination using the inclusion-exclusion principle.

		Probabilistic Term Combination	
		Disjunctive	Conjunctive
Task	Retrieval	$P(q\|d) = \sum_{t\in q} P(t\|d)\cdot P(q\|t)$ generalised here to: $P(q\|d) := \sum_{t\in q} P(t\|d)\cdot P(c\|t)$	$P(q\|d) := \prod_{t\in q} P(t\|d)$ implemented as Jelinek-Mercer: $P(q\|d) := \lambda\cdot P(t\|d) + (1-\lambda)\cdot P(t\|c)$
	Summarisation	POLIS-Base model: $P(c\|e) := \sum_{t\in e} \frac{P(t\|e)}{P(t\|e)+P(t\|c)}$ POLIS-macroaveraging model: $P(c,q\|e) := P(c\|e) \oplus P(q\|e)$ POLIS-macroaveraging model: $P(c,q\|e) := \sum_{t\in c} P(c\|t)\cdot P(t\|e,q)$	N/A

Figure 5.2: Disjunctive and conjunctive models for retrieval and summarisation.

We call this model the *microaveraging model*, as here the element- and query-evidence is combined on the term-probability level, as opposed to the macroaveraging model, where a combination of evidence takes place at the element level only.

5.5 Conclusion

In this chapter, we investigated the ranking of elements with respect to the context to be summarised. We looked at ways in which suitable summarisation models could be derived from traditional ad-hoc information retrieval models, such as TF-IDF or Language Modelling, and how such models can be classified depending on the way in which they combine term evidence. In addition to the generic summarisation models, we developed three query-based summarisation models: a macroaveraging model, where element- and query-probabilities are mixed at element level, and two microaveraging models, which mix element- and query-probabilities at the term level, and which differ in the interpretation of probabilities. Figure 5.2 follows the matrix spanned by Figure 5.1, and presents the summarisation models developed in this chapter.

In the next chapter we will develop the summarisation logic POLIS. Part of the definition of POLIS includes its implementation based on Probabilistic Relational Algebra (PRA), where we will also discuss how the summarisation models developed in this chapter can be implemented in PRA. Finally, in Chapter 7 we evaluate the performance of the summarisation models developed in this chapter.

Chapter 6

POLIS

6.1 Introduction

The previous chapters introduced some main building blocks of a summarisation logic: a comparison with other abstraction logics, and the benefits of such logics in building information systems, provide the motivation for using abstraction logics with summarisation as a retrieval task. The parallels between summarisation and –amongst other tasks– structured document retrieval provided the rationale for developing probabilistic summarisation models derived from ad-hoc retrieval approaches. Additionally, the parallels between said retrieval tasks under the concept of knowledge augmentation lays the foundation of the semantics of POLIS. However, we so far have not made clear why a *specialised* logic for summarisation is necessary: as the summarisation models discussed in the previous chapter are derived from retrieval models, it would be conceivable to *implement* summarisation in a retrieval logic, such as POOL. We will now show that such an approach suffers from serious drawbacks, which are sufficient to motivate the development of a dedicated summarisation logic.

6.1.1 Summarisation in POOL

POOL was developed as an abstraction for *hypermedia* retrieval, i.e. the retrieval of structured documents, with added support for concepts such as object classification, and relationships between objects. Retrieval strategies in POOL are expressed in a syntax which bears resemblance to Probabilistic Datalog, such that those objects to be retrieved are specified as a target, for which the system tries to find objects in its knowledge-base that satisfy the specification. An expression to retrieve a document about “sailing” could thus look as follows:

```
?− document(D) & D[sailing]
```

Similarly, it would be possible to translate summarisation needs into POOL statements. The need for a summary about “doping in sport”, built from sentences within a document, could be expressed as

```
?− document(D) & D[ sentence(S) & S[doping in sport ]]
```

Modelling summarisation in POOL in this way would weight sentences according to the retrieval models built into POOL, not the probabilistic models developed in chapter 5. The summarisation models we developed weight sentences according to the degree of implication between summary-forming elements and the context to be summarised. POOL supports a form of implication, denoted by →, so it would be conceivable to implement a refined summarisation strategy in POOL as:

```
?− document(D) & D[
                        sentence(S) &
                        S[doping in sport] &
                        S → D
                      ]
```

However, there are still several limitations to this approach. Firstly, it would not be possible to quantify the retrieval process, to allow summary-size limitations. The restriction to a specific summary size would therefore need to be externalised. Furthermore, while it might be possible to express the summarisation models developed in chapter 5 in POOL, they would need to be specified explicitly. A knowledge engineer thus would not only need to posses knowledge of summarisation models in general, but would additionally need to be able to express them in POOL, rendering the abstraction which motivated the use of abstraction logics obsolete.

A dedicated summarisation logic overcomes both problems: POLIS expressions allow for size quantifiers, and implicitly use probabilistic summarisation models.

6.1.2 Chapter Outline

The remainder of this chapter provides a detailed specification of POLIS. Section 6.2 will introduce the POLIS syntax, which is derived from the established XPath notation. Section 6.3

introduces the semantics of POLIS, which is a variant of possible-worlds semantics. This is followed by section 6.4, which discusses the implementation of POLIS using Probabilistic Relational Algebra (PRA).

6.2 POLIS Syntax

In this section, we will introduce the syntax of POLIS.

POLIS is a summarisation logic for *structured* documents. Structured documents not only contain the actual textual content, but additionally provide markup ("tags") to annotate the content. Such markup will usually provide information regarding the structure of documents (structural markup), outlining chapters, sections, paragraphs, and sentences. At a level below this structural markup (e.g. within sentences), tags might also provide layout information, such as enclosing italicised words in appropriate tags (e.g. <it>italicise< /it>), or providing additional information regarding references used in the text.

POLIS expressions exploit the availability of this this structural markup, by including the path to the level at which users want material to be extracted into the summarisation logic expression. As mentioned in previous chapters, POLIS makes use of a syntax which is modelled on XPath to achieve this (Clark and DeRose (1999)).

XPath is a language which was developed for selecting nodes from an XML document. Based on a tree representation of XML documents, an XPath expression provides a path for traversing the tree. An XPath expression consists of an *axis*, a *node test*, and a *predicate*. An axis in an XPath expression denotes a number of individual elements in the XML document, which can be found by traversing the document tree according to the path.

```
<document>
 <section>
  <paragraph>
   <sentence>
     A single sentence in a paragraph,
     in a section, in a document.
   </sentence>
  </paragraph>
 </section>
</document>
```

Figure 6.1: A sample XML document, showing a single sentence within a paragraph, embedded in a section.

For example, considering the sample XML document shown in Figure 6.1, the path expression necessary to find all sentence-elements in the document would look like

/document/section/paragraph/sentence

(note that the sample document only has one such element).

For POLIS, we only make use of the definition of an axis. Node test and predicates do not form part of the POLIS syntax. Instead, POLIS contains additional constructs for specifying query terms in an element, choosing models for the summarisation process, and limiting the maximum size of summaries. Furthermore, POLIS does not fully exploit the complexity offered by structured documents in general; for example, attributes and attribute values, as well as references between nodes, are not interpreted by POLIS. Generally, the focus of POLIS is on document-centric structured documents, rather than data-centric documents.

6.2.1 Syntax Definition

We will now properly specify the syntax of POLIS. We will follow the convention of using BNF production trees to show how POLIS expressions are constructed. The syntax tree can be split into three separate groups: one group providing information on the path of elements to be extracted, a group showing the specification of query terms, and a group detailing auxiliary definitions.

The first part of any POLIS expression starts with a definition of the elements at which the extraction of material for the summary generation will take place. We say that a POLIS-expression starts with a root-definition, followed by a specification of the axis of the elements to form the summary:

The BNF rules in Table 6.1 exhibit the influence of XPath on POLIS; an axis is used to define the components to be extracted. The actual path in the document tree is given by an axis-definition. An axis-definition can be either a path-element (called *axis-element* here), or a summary-definition, restricting the number of elements (or words) to appear in the summary. Axis-elements can be followed by either a node-definition followed by a summary definition, or a node-definition followed by an axis definition. This path expression is followed by a summary definition.

A summary definition is either a summary restriction, or a query-definition followed by a summary restriction. A query definition consist of a specification of the summarisation model to be used, and the query terms. There are currently three models implemented in POLIS, two microaveraging query-based models (signified by "MICRO" and "MICRO-IDF", respectively), and a macroaveraging model (denoted by "MACRO"). The specification and development of these models were shown in chapter 5. If no query terms are specified (i.e. a summary definition is resolved as a summary restriction) the standard, non query-based summarisation model will

expression	::=	root-element axis-definition
root-element	::=	'/'
axis-definition	::=	axis-element
		| summary-definition
axis-element	::=	node-definition summary-definition
		| node-definition '/' axis-definition
node-definition	::=	'*'
		| '*' '{' node-restriction '}'
		| NAME
		| NAME '{' node-restriction '}'
node-restriction	::=	context-identifier
context-identifier	::=	NUMBER
summary-definition	::=	':' '{' summary-restriction '}'
		| query-definition ':' '{' summary-restriction '}'
query-definition	::=	{ model '"'query-term'"' }
summary-restriction	::=	range-restriction
		| element-restriction
range-restriction	::=	'at_most' NUMBER 'terms'
		| 'at_most' NUMBER
element-restriction	::=	NUMBER
		| NUMBER ',' element-restriction
model	::=	'MACRO'
		| 'MICRO'
		| 'MICRO-IDF'
query-term	::=	NAME
		| NAME query-term

Table 6.1: POLIS syntax: production of a path to summary-forming elements; production of query terms and query biased summaries, and summary size limitations.

be used.

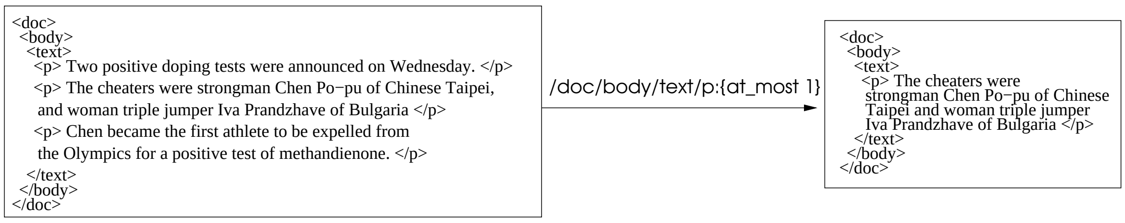
A summary restriction is used to indicate the maximum size of the summary, and can be either a range restriction, or an element restriction. An element restriction simply specifies that a particular element, or a series of elements, should make up the summary. A range restriction limits the actual size of the summary, either in terms of the number of elements in the summary (e.g. a maximum of five sentences in the summary), or by limiting the size of the summary in terms of the number of words in it.

NAME	::=	$[a-z][a-z\_A-Z0-9]^*$
NUMBER	::=	$[0-9]^+$

Table 6.2: POLIS syntax: lexical syntax for names and numbers.

The last of the three BNF trees making up the POLIS syntax provides auxiliary definitions used in the other two trees, specifying the definition of a name, and a number (Table 6.2).

6.2.2 Example



```
<doc>
 <body>
  <text>
   <p> Two positive doping tests were announced on Wednesday. </p>
   <p> The cheaters were strongman Chen Po−pu of Chinese Taipei,
    and woman triple jumper Iva Prandzhave of Bulgaria </p>
   <p> Chen became the first athlete to be expelled from
    the Olympics for a positive test of methandienone. </p>
  </text>
 </body>
</doc>
```

```
<doc>
 <body>
  <text>
   <p> The cheaters were
    strongman Chen Po−pu of Chinese
    Taipei and woman triple jumper
    Iva Prandzhave of Bulgaria </p>
  </text>
 </body>
</doc>
```

Figure 6.2: A sample document, its one-paragraph summary, and the POLIS expression to generate it.

Let us now take a look at how the above POLIS syntax could be used to express summary definitions in actual information systems. We will use a web information retrieval system as a sample application.

Consider a system where the set of retrieved web pages should be presented in such a way as to allow users to immediately judge a document's relevance to their information needs. This is a realistic scenario, as users will normally not look at all web pages retrieved by a search engine, but will instead try to judge a page's relevance by whatever information is presented (such as a pages title, its URL, a text snippet taken from the web page etc.). Summaries of retrieved web pages would have to be indicative of the content, but would have to be short enough to allow a relatively large number of pages to be displayed on a single page. One could argue that a good compromise between the summary size and its content is reached by letting the summaries be up to one paragraph long.

A very simple expression to fulfil this request could look like:

/paragraph:<=1.

This, however, is not supported by POLIS at the moment, as the path expression has to match the actual structure of documents, which would normally be more complex than just paragraph markup. The actual POLIS expression to carry out this operation for a document that has paragraphs, which are children of ’section’ elements inside ’body’ elements would be:

/*/body/sec/p:{at_most 1},

A graphical representation of this process is depicted in Figure 6.2, exhibiting the summarisation of a (truncated) DUC[1] document.

For a second example, we will modify the above POLIS expression to include a query, and will limit the maximum size to 250 words. We will pick up the topic of the shown sample DUC document, doping at the olympic games. A POLIS expression for a 250 word, query based summary on “doping olympic games” might thus look like:

/*/body/sec/p{MICRO “doping olympic games”}:{at_most 250 terms},

assuming that the microaveraging query-based model was used for summarisation.

6.2.3 Summary

In this section, we developed the syntax for the summarisation logic POLIS. We have shown how POLIS statements are related to XPath expressions, and how the need for specific summaries could be expressed and realised in POLIS. More specifically, we showed how a POLIS expression consists of two main components: a path expression component pointing to the user selected elements which should be considered for inclusion in the summary, and a numbered quantification component which imposes limits on the size of the summary. This concludes our discussion of the POLIS syntax.

This section did not include a discussion of the semantics of POLIS expressions. While it would have been possible to provide static semantics for the XPath-like POLIS expressions (e.g.

[1]Document Understanding Conference, see chapter 7

using ordered-attribute grammars), we do not believe that it is necessary to provide such semantics for POLIS. Firstly, the POLIS syntax is constrained such that it is not possible to formulate expressions which would be syntactically correct, but could not be interpreted by POLIS. All possible expression (i.e. allowed by the syntax definitions) constitute correct expressions, which can be interpreted by POLIS. Furthermore, POLIS is not defined by its syntax. It would be possible to conceive a completely different syntax for POLIS, without changing the logic itself. Defining a proper semantics for the syntax would imply an importance for the syntax which is not justified.

Instead, we will provide a "meaning" of POLIS via possible-worlds semantics in the next section.

6.3 POLIS Semantics

In the last section we introduced the XPath-like syntax of POLIS. Here, we will continue to provide a formal specification of POLIS, by introducing the POLIS semantics. While formally specifying the syntax allows us to validate the correctness of a POLIS expression, a formal semantics allows us to reason about the *meaning* of POLIS expressions. To have a rigid definition of this *meaning* is necessary for two aspects of reasoning about POLIS: to check the equivalence of two POLIS expressions, and, more importantly, to validate that a specific implementation of POLIS is correct. The semantics outlined in this chapter is based on the notion of a *possible worlds interpretation*, similar to the *denotational semantics* commonly employed by Description Logics.

We will start with a discussion of possible worlds, which we will subsequently use to define the semantics of POLIS expressions. The semantics of POLIS is directly related to the interpretation of the knowledge in structured documents, which will be outlined subsequently. We will close the discussion with some examples for the interpretation of POLIS expressions and structured documents.

6.3.1 Possible Worlds

6.3.1.1 Introduction

The semantics of POLIS is based on the notion of *possible worlds*, as discussed by Fagin *et al.* in (Fagin and Halpern (1994); Fagin et al. (1995)). The notion of a set of possible worlds is built around an *agent*, which reasons about the world it is in, and by this process of reasoning eliminates all but those worlds it considers possible.

Each world contains a set of propositions, each of which has a certain truth value. For classical two valued logics, this truth value is either *true* or *false*. However, larger sets of attainable values are possible (e.g. four-valued logics (Roelleke (1999))). We will restrict the discussion to the classical case of a two-valued logic for now. A proposition in the context of possible worlds semantics can be considered a *ground atom*; in the context of information retrieval, these ground atoms are usually called *terms*, however, for this chapter we will mainly refer to them by the more general *proposition*.

To show how sets of possible worlds emerge from propositions and their possible truth values, consider two propositions, φ_1 and φ_2, which can have the truth values *true* and *false*. This yields a total of 2^2 possible worlds (two propositions, two truth values): $w_1 = \{\varphi_1, \varphi_2\}$, $w_2 = \{\varphi_1\}$, $w_3 = \{\varphi_2\}$, and $w_4 = \{\}$. In w_1 both φ_1 and φ_2 are true, in w_2 and w_3 only one of the two propositions is true, and in w_4 both propositions are false. The truth values of propositions in worlds also show that by default a Closed-World Assumption (CWA) is taken – any proposition for which no truth value is given is assumed to be false.

An agent reasoning about the set of possible worlds can form a *belief* about the truth value of propositions. The belief of an agent is derived from the truth valuations of propositions in all worlds the agent considers possible; an agent A believes a proposition φ is true if φ is true in all worlds A considers possible from its current world. For example, given the set of worlds $\{w_1 \cdots w_4\}$ outlined above, consider an arbitrary world w_0. If an agent A in w_0 considers w_1 and w_2 possible worlds, it knows φ_1 is true, and has conflicting information about the truth value of φ_2 (reasoning with conflicting information will be discussed in a later section).

The knowledge of an agent is determined by the worlds it considers possible. Instead of reasoning about possibilities, we can rephrase this statement by saying that an agent can *access* other worlds from the world it is in. The belief of an agent in the truth value of a proposition is then determined by the truth values of the proposition in those worlds the agent can access. Information about which worlds are accessible from any one world by an agent are stored in the worlds *accessibility relation*.

Following the example of an agent in w_0, which can *access* w_1 and w_2, the accessibility relation of w_0 would be as follows:

$$R_{w_0} \quad = \quad \{ \, (w_1, w_2) \, \}$$

6.3.1.2 Relation to Information Retrieval

The previous discussion of possible worlds and agents covered all relevant aspects of denotational semantics in general terms, to provide an overview over the concepts used for defining semantics in terms of possible worlds. We now take those definitions, and relate them to concepts commonly found in Information Retrieval. This allows us to formally represent the information contained in documents, as well as give meaning to POLIS expressions.

With a view of IR in general, and structured document retrieval in particular, it would be possible to view elements in structured documents (the document *contexts*) as possible worlds. However, this intuitive approach to relating possible worlds semantics to IR leads to several problems. First, it is not clear which concept in the context of IR would fill the role of an agent. An agent has to access the possible worlds, however, if the contexts of documents are those worlds, what are they accessed by? A second, more severe, problem comes with the probabilistic indexing process commonly used in IR. When indexing a document, terms in documents are usually given a *weight* to reflect their importance for the document, measured by some weighting function seen fit. However, in the context of possible worlds semantics, terms in documents would become propositions in those worlds; it is unclear how the weights (or probabilities) assigned by the indexing process could be reflected in this interpretation.

To overcome these problems, we adopt a slightly different association of denotational semantics and IR. We assume that elements in documents (the *contexts*) are the agents. These agents then access sets of possible worlds, which contain varying subsets of the terms (*propositions*) found in the respective document elements. This interpretation of structured document elements as agents also allows us to incorporate the probabilities assigned by an indexer into the semantics in a meaningful way: the probability of a term in a document-context is the fraction of all possible worlds the context-agent can access, in which the term (the proposition) is true. We will discuss these points in more detail in the following sections.

This interpretation of possible worlds also affects the terminology we adopt for the remainder of this chapter. We will interchangeably refer to "agent" and "context", where "context" will take the precedent for the remainder of the discussion. To visually highlight the correspondence between agents in denotational semantics and contexts in structured documents, we will use a notation related to XML to express the knowledge of a context. We will use statements of the form $< c_i \ \varphi >$ to state that context c_i has knowledge of the proposition φ. A precise meaning of this will be given over the next sections.

An additional complication arises from the Closed World Assumption (CWA) commonly adopted in logics and denotational semantics. The CWA states that any proposition for which a truth value is not explicitly given is assumed *false*. This is problematic in IR, where the indexing process which assigns terms and term-weights to documents is assumed incomplete (either historically, where an indexer used to only pick a handful of terms per document due to memory storage limitations, or intrinsically, as in Language Modelling, where the document itself is assumed to be only a sparse random sample). IR therefore adopted the Open World Assumption (OWA): any proposition for which no explicit truth value is given is assumed to have the truth value *unknown*. To accommodate the OWA into our semantics, we extend the traditional two-valued logic to a three-valued one, where the possible truth values are *true, false* and *unknown*.

However, it is important to point out that these truth values are only needed for truthfully representing the knowledge of documents, and reasoning about it. It is not possible in either POLIS expressions, or traditional structured document, to say that a document is *not* about a specific term. An indexer will therefore never derive a proposition with a truth value *false*. Similarly, it would be impossible for an indexer to arrive at the conclusion that a proposition is *unknown*: neither POLIS nor traditional structured documents allow the explicit inclusion of this information, and the set of potentially unknown proposition with respect to a document would be infinite.

6.3.2 Interpretation Structure and Interpretation Function

To make the informal discussion of possible worlds in the previous section useful, we have to specify them formally. For specifying the semantics formally, we will make use of a Kripke structure (Fagin et al. (1995)). This formal specification consists of an Interpretation Structure M, which contains a finite set of possible worlds, a truth value function that assigns truth values to propositions, the accessibility relation for its set of possible worlds, and the Interpretation Function $\models$, which assigns truth values to propositions with respect to contexts.

Both Interpretation Structure and Interpretation Function will be developed in stages, starting first with a non-probabilistic interpretation structure, which we will extended to a probabilistic variant. This will be followed by a definition of the interpretation function, again in a non-probabilistic and probabilistic variant.

6.3.2.1 Non-probabilistic Interpretation Structure

We start the discussion with the non-probabilistic version of the interpretation structure M. For a set of propositions Φ, and a set of contexts of size n, M is a tuple

$$\mathrm{M} \quad = \quad (\mathcal{W}, \pi, \mathcal{R}), \text{ such that}$$

- $\mathcal{W}$ is a finite set of possible worlds
- π is a function that assigns truth values to propositions in a world w, such that π assigns a truth value to each proposition

$$\pi(w){:}\Phi \quad \rightarrow \quad \{\textit{true, false, unknown}\}$$

- $\mathcal{R}$ is the set $\{R_1 \cdots R_n\}$ of binary accessibility relations on $\mathcal{W}$, such that all tuples $\{w,w'\}$ $\in R_c$ denote those worlds *w'* which context *c* accesses from world *w*

In addition to the basic interpretation structure, we define a function which allows us to reason about all those worlds a context can access from the world it is currently in. We thus define

$$R_c(w) \quad := \quad \{\, w' \mid \{w, w'\} \in R_c\}$$

as the function whose range are all worlds accessible for context *c* from world *w*.

This concludes the definition of the non-probabilistic interpretation structure, which we can now extend to a probabilistic variant more suitable for IR purposes.

6.3.2.2 Probabilistic Interpretation Structure

Section 6.3.1.2 outlined the need to include probabilities into the interpretation of documents, and into the representation of document knowledge. In this section, we extend the non-probabilistic interpretation structure by a probabilistic framework that allows us to reason about probabilities. In order to include probabilities into the interpretation structure, we need to define what probabilities are, and introduce the notion of a probability space. This definition will follow that outlined in (Fagin and Halpern (1994)). The definition of a probability function outlined by Fagin and Halpern coincides with the probabilistic extension to propositional logics proposed by Nielsson (Nilsson (1986)); it thus provides a connection between probabilistic logics, and a semantics about probabilities.

The basic construct for defining a probability space is an event. In our semantics, truth valuations of propositions form the events; events are statements of the form φ *is true*. These events are *atomic*, i.e. not composed by union or complement of other events. The probability of a truth value (or valuation) of a proposition is defined as the probability of the set of worlds where the proposition has the corresponding truth value. We will call the set of all sets of possible worlds $\mathcal{H}$. With these preliminaries in place, we can give a formal definition of a probability space.

A probability space is a tuple $(\Omega, \mathcal{H}, \mu)$, where Ω is the set of atomic events, $\mathcal{H}$ is a σ-algebra of Ω, and μ:$\Omega \rightarrow$ [0,1] is a probability mass function, such that $\forall e \in \Omega$: $\mu(e) \geq 0$ and $\sum_{e \in \Omega} \mu(e) = 1$. Here, μ can also be called the basic probability function.

Given this definition of a probability space, we can define a more generalised probability function over the σ-algebra of Ω, $\mathcal{H}$. Given a probability space *P*, the probability function $P : \mathcal{H} \rightarrow [0,1]$ is defined as

$$\forall w \in \mathcal{H} : P(w) \quad := \quad \sum_{e \in w} \mu(e)$$

The elements in Ω are the basic events, the elements in $\mathcal{H}$ are the possible worlds. Each possible world is a combination of basic events.

With this definition of a generalised probability function P, it is possible to extend the interpretation structure M to allow reasoning about the probability of events. The probabilistic interpretation structure M is defined as a tuple

$$M \quad = \quad \mathcal{W}, \pi, \mathcal{R}, P, \text{ such that}$$

- $\mathcal{W}, \pi$, and $\mathcal{R}$ are identical to those defined for non-probabilistic interpretation structures, and
- P is a function which assigns a probability space $(\mathcal{W}_{c,w}, \mathcal{H}_{c,w}, \mu_{c,w})$ to each context c and world w.

This concludes the definition of both the non-probabilistic and the probabilistic interpretation structures. We will now develop a non-probabilistic and a probabilistic interpretation function, to allow reasoning on the interpretation structures.

6.3.2.3 Non-probabilistic Interpretation Function

Reasoning about the knowledge of documents, and the meaning of POLIS expressions, requires a way to represent the knowledge of document, and some means to determine the truth value of propositions with respect to possible worlds. While the interpretation structure M represents information about the set of possible worlds, and thus about the knowledge of a document, we need an additional component to determine the truth value of propositions. To allow this, we now develop the interpretation function $\models$ ("models"). An expression of the form $(M, w) \models \varphi$ states that the proposition φ is *true* with respect to the interpretation structure M and the world w. This notation is already a shorthand for $(M, w) \models_t \varphi$, saying that the interpretation structure M with respect to world w holds that proposition φ has a truth value of *true*. Similarly, the interpretation function can assign the truth values *false* and *unknown* to a proposition, modelled by $\models_f$ and $\models_u$, respectively. Using this informal notion of the interpretation function, we can now provide a formal definition for the interpretation of propositions. Using the truth value function provided as part of the interpretation structure, the interpretation of propositions is formally defined as:

$$
\begin{aligned}
(M,w) \models_t \varphi &\iff \pi(w)(\varphi) = \mathit{true} \\
(M,w) \models_f \varphi &\iff \pi(w)(\varphi) = \mathit{false} \\
(M,w) \models_u \varphi &\iff \pi(w)(\varphi) = \mathit{unknown}
\end{aligned}
$$

A proposition φ is true with respect to an interpretation M and a world w if and only if the truth value function assigns the truth value *true*. This definition similarly extends to the truth value assignments of *false* and *unknown*.

6.3.2.4 Probabilistic Interpretation Function

Given the definition of the non-probabilistic interpretation function, we can now provide a definition for the probabilistic extension to the interpretation function. The probabilistic interpretation function not only assigns a truth value to proposition, but also a probability of this truth value. Informally, these probabilities can be interpreted as the probability of a possible world in the set of worlds reachable by a context: a context c can reach a set of worlds $R_c(w)$ from world w. For any world $w' \in R_c(w)$, a proposition in w' is assigned a truth value (i.e. *true, false* or *unknown*). For example, $(M,w') \models_t \varphi$ means proposition φ has a truth value *true* in w'. With the probabilistic interpretation, each possible world w' is assigned a probability $\mu_{c,w}(w')$. The probability of any truth value of φ in w is then the sum over the probabilities of the possible worlds in which the truth value holds.

For example, a statement of the form $(M,w) \models_t 0.4\varphi$ means that φ has a truth value of *true* with a probability of 0.4, by summing over the probabilities of all possible worlds in which φ is *true*. Similarly, probabilities are assigned to the truth values *false* and *unknown*. We can generalise the assignment of probabilities by saying that every truth valuation holds with some probability ρ. Based on this generalisation, we can formally define the probability of a truth valuation as

$$
\begin{aligned}
(M,w) \models \rho^t\varphi &\iff \rho^t = \mu_{c,w}(\{w' \mid w' \in \mathcal{R}_c(w) \wedge (M,w') \models_t \varphi\}) \\
(M,w) \models \rho^f\varphi &\iff \rho^f = \mu_{c,w}(\{w' \mid w' \in \mathcal{R}_c(w) \wedge (M,w') \models_f \varphi\}) \\
(M,w) \models \rho^u\varphi &\iff \rho^u = \mu_{c,w}(\{w' \mid w' \in \mathcal{R}_c(w) \wedge (M,w') \models_u \varphi\})
\end{aligned}
$$

Each probability (ρ^t, ρ^f, ρ^u) is defined as the sum over the probabilities of accessible worlds where the corresponding truth value holds. Before we continue with the definition of the POLIS semantics, we will now show the application of these interpretation structures and interpretation functions to a sample document.

6.3.2.5 Examples

To show how the possible worlds interpretation is applied to a structured document, we will refer to the sample document in Figure 6.3. The document is a text snippet taken from a document in the Document Understanding Conference 2005 corpus. The document consists of three paragraphs, which are part of one common “text” tag, embedded in the wider document context.

```
<doc>
 <body>
  <text>
   <p> Two positive doping tests were announced on Wednesday. </p>
   <p> The cheaters were strongman Chen Po−pu of Chinese Taipei,
    and woman triple jumper Iva Prandzhave of Bulgaria </p>
   <p> Chen became the first athlete to be expelled from
    the Olympics for a positive test of methandienone. </p>
  </text>
 </body>
</doc>
```

Figure 6.3: Sample document to detail the valuation of proposition “Bulgaria”.

We will start with a non-probabilistic interpretation of the document, without any probabilities which might have been provided by an indexing process. The sample document in Figure 6.3 contains a number of contexts / agents: there are three “paragraph” contexts (marked $<p>$ in the document) which we will refer to as *p1, p2*, and *p3*, a “text” context *text*, a “body” context *body*, and a “document” context *doc*. Additionally, there is a set of worlds pertaining to an unnamed global context $\circledcirc$. This global context is necessary to provide a world from which context *doc* can access its set of worlds.

For context *p2*, the context in which the term *Bulgaria* appears in the sample document, “Bulgaria” has a truth value *true*. For all other context, there was no indication in the document for “Bulgaria” to have a truth value of either *true* or *false*, and no augmentation of knowledge between contexts has taken place (knowledge augmentation will be discussed in 6.3.3). In those contexts “Bulgaria” therefore has a truth value of *unknown*. The truth values of the proposition “Bulgaria” with respect to the different contexts are shown in Table 6.3.

The accessibility relation for the sample document contains tuples for every context in the document. For example, for contexts *doc, p1* and *p2*, the accessibility relations could be:

$$
\begin{aligned}
R_{doc} &= \{(w_{\circledcirc}, w_{doc})\} \\
R_{p1} &= \{(w_{text}, w_{p1})\} \\
R_{p2} &= \{(w_{text}, w_{p2})\}
\end{aligned}
$$

world w'	proposition φ	truth value $\pi(w')(\varphi)$
$w_{\circledcirc}$	Bulgaria	*unknown*
w_{doc}	Bulgaria	*unknown*
w_{body}	Bulgaria	*unknown*
w_{text}	Bulgaria	*unknown*
w_{p1}	Bulgaria	*unknown*
w_{p2}	Bulgaria	*true*
w_{p3}	Bulgaria	*unknown*

Table 6.3: Truth values for proposition “Bulgaria” from an interpretation of the sample document with respect to the different contexts.

The truth valuation of “Bulgaria” can thus also be explained via the accessibility relations: “Bulgaria” is true in those worlds context *p2* can reach from w_{text}, but is unknown in those worlds *doc* can reach from $w_{\circledcirc}$, or *p1* can reach from w_{text}.

We now further this example to show the probabilistic extension of the possible worlds semantics. Suppose the document in Figure 6.3 has been processed by some indexing function, which yields an assignment of term weights, or term probabilities. Suppose that the term *Bulgaria* by way of an indexing process has been assigned a probability of 0.4, i.e. the proposition “Bulgaria” has a truth value assignment of *true* in its respective context with a probability of 0.4. Figure 6.4 displays how the probabilities assigned by an indexer might conceptually be embedded together with the term they apply to in a context. The figure shows a simplified version of the sample document, with *p2* as the only “paragraph” context, and “Bulgaria” as the only proposition.

```
<doc>
<body>
 <text>
   <p>
     [...] 0.4 Bulgaria [...]
   </p>
 </text>
</body>
</doc >
```

Figure 6.4: The sample document in Figure 6.3 with a probability of 0.4 for proposition “Bulgaria”.

The knowledge that “Bulgaria” is true with a probability of 0.4 also affects the set of possible worlds. To account for this probability, there now need to be at least two possible worlds with respect to context *p2*: one in which “Bulgaria ” is *true*, and one in which it is *unknown*. These two worlds are accessed by *p2*, to derive at the conclusion that “Bulgaria” is *true* with a probability of 0.4, and is *unknown* with a probability of 0.6. Based on this simplest set of possible worlds, we arrive at the following set of truth value assignments:

world w'	$\mu_{p2,w_{text}}(w')$	proposition φ	$\pi(w')(\varphi)$
$w_{p2,1}$	0.4	Bulgaria	true
$w_{p2,2}$	0.6	Bulgaria	unknown

Table 6.4: Probabilities of possible worlds for probabilistic truth value assignment of proposition “Bulgaria”.

In world $w_{p2,1}$ “Bulgaria” is *true*, and in world $w_{p2,2}$ “Bulgaria” is *unknown*. It is also *unknown* in all other possible worlds, reachable by context agents other than *p2*. The accessibility relation for *p2* for the probabilistic sample document is thus:

$$R_{p2} \quad = \quad \{(w_{text}, w_{p2,1}), (w_{text}, w_{p2,2})\}$$

Combining the truth value assignments and accessibility relations with the probability assignments, the interpretation structure contains for proposition “Bulgaria” with respect to those worlds accessible by *p2* the following statements:

$$(M, w_{p2,1}) \models_t \text{Bulgaria} \quad \Longleftrightarrow \quad \pi(w)(\text{Bulgaria}) = \textit{true}$$
$$(M, w_{p2,2}) \models_u \text{Bulgaria} \quad \Longleftrightarrow \quad \pi(w)(\text{Bulgaria}) = \textit{unknown}$$

Following the formal definition of the probability of a truth valuation, we can show that the probabilistic interpretation of “Bulgaria” is:

$$(M, w_{p2}) \models 0.4\ \textit{Bulgaria} \Longleftrightarrow$$
$$0.4 = \mu_{p2,w_{text}}(\{w' \mid w' \in R_{p2}(w_{text}) \wedge (M, w') \models_t \text{Bulgaria}\}) \text{ and}$$
$$0.6 = \mu_{p2,w_{text}}(\{w' \mid w' \in R_{p2}(w_{text}) \wedge (M, w') \models_u \text{Bulgaria}\})$$

This concludes the discussion of the outlined examples. An important concept we briefly mentioned is that of *knowledge augmentation*. Knowledge of one agent is propagated to other agents. For the examples given above, a proposition such as “Bulgaria” would not only be *true* in *p2*, but would also be true to a certain degree in other context. We will formally discuss the process of knowledge augmentation in the next subsection.

6.3.3 Augmentation of Knowledge

The process of knowledge augmentation is concerned with making the knowledge restricted to single contexts available to other contexts. In the case of a hierarchical decomposition of contexts, as in structured documents discussed here, it would be reasonable to augment the knowledge of higher levels in the hierarchy with knowledge of its sublevels. For example, given the sample document in Figure 6.3, it would be reasonable to say that if context *p2* knows that the proposition "Bulgaria" is true, then so should context *text*, since in the hierarchical decomposition of the sample document, *p2* is a subcomponent of *text*. We will now formally define how the knowledge of contexts can be augmented, by incorporating the knowledge of other contexts. The discussion of knowledge augmentation presented here is based on the outline given in (Roelleke (1999)). We will again start with a non-probabilistic version of knowledge augmentation, and extend it to a probabilistic version later.

6.3.3.1 Non-probabilistic Knowledge Augmentation

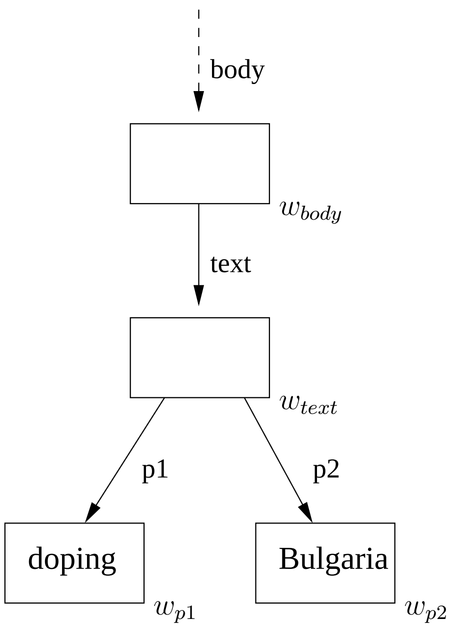


Figure 6.5: Possible World Tree generated from sample document in Figure 6.3.

Consider the simplified tree-representation of contexts in the sample document (Figure 6.5). The tree represents a possible set of accessibility relations amongst the contexts in Figure 6.3. We can use such a tree representation of contexts to define the spreading of knowledge up the hierarchy. To draw a formal definition of knowledge augmentation, we will make use of the concepts of *common* and *distributed* knowledge, discussed by Fagin *et al.* in (Fagin et al. (1995)). Both common and distributed knowledge reason about the knowledge of groups of agents / contexts, rather than individual contexts in isolation. Both concepts will be defined as modal operators for combining knowledge.

Suppose that G is a group of contexts, and every context in G knows that φ is *true*. We can thus say that *everyone* in G knows φ. We will express this *group knowledge* as $< E_G \ \varphi >$. Common knowledge can be interpreted as a transitive closure of this group knowledge: every context in G knows φ, every context in G knows that every context in G knows φ, etc. We will use the shorthand $< C_G \ \varphi >$ to denote that φ is *common* knowledge. To formally define the meaning of $< C_G \ \varphi >$, we will refer to the set of accessibility relations for a group G of contexts. We say that a world w_n is *group-reachable* from another world w_0, if there exists a path in the set of accessibility relations that links w_0 to w_n. This path can be visually derived from tree representations of the kind shown in Figure 6.5, or can be defined explicitly formally as follows. Suppose G is a group of contexts. We can say a world w_n is group-reachable from w_0, if for all worlds $w_0, \cdots, w_n$, such that $1 \leq j \leq n$, there exists a context $c \in G$ for which $(w_{j-1}, w_j) \in R_c$.

We can thus say that $< C_G \ \varphi >$ is true with respect to a world w if and only if φ is true in all worlds group-reachable from w:

$$(M,w) \models < C_G \ \varphi > \quad \Longleftrightarrow \quad \forall w' \text{group-reachable from w:} (M,w') \models \varphi$$

Where common knowledge makes explicit that every context has knowledge of a proposition, it would also be reasonable to augment the knowledge of a context with propositions only present in some subcontexts.

As a second modal operator for combining knowledge of contexts, we will introduce the *distributed* knowledge operator. The *common* knowledge operator explicitly states what is known by every context. On the other hand, the *distributed* knowledge operator starts from the premise that different contexts have differing knowledge, however, this knowledge can be shared and combined if the different contexts can reach the same possible worlds. Suppose a context agent c_1 can access the possible worlds w_i and w_j from a world w, and knows that in w proposition φ_1 is true. Another context, c_2 can access the worlds w_i and w_k from w, and knows that φ_2 is true. Since the accessibility relations of c_1 and c_2 overlap w_i, both φ_1 and φ_2 must be true in w_i. It would thus be reasonable to assume that c_1, too, should know that φ_2 is *true* in w. We will use the shorthand $< D_G \ \varphi >$ to say that φ is distributed knowledge among a group G of contexts. A formal definition of distributed knowledge would be as follows:

$$(M,w) \models < D_G \ \varphi > \quad \Longleftrightarrow \quad \forall \text{ w': (w,w')} \in \bigcap_{c \in G} R_c \rightarrow (M,w') \models \varphi$$

However, this interpretation of distributed knowledge is too strict to be useful for the combination of context knowledge, since it requires a non-empty intersection of accessibility relations of contexts to arrive at distributed knowledge. Furthermore, it does not consider the case of conflicting information, where a context c_1 knows that φ is true, and another context c_2 knows φ is false, since all contexts have to model the same truth value. Roelleke proposes the introduction of a *united knowledge* operator, which can handle incomplete or inconsistent knowledge between contexts. We follow this definition here, amending the operator to handle three-valued truth assignments. The united knowledge operator U_G defines the truth value of a proposition with respect to a context by uniting the knowledge of contexts. We will use the shorthand $< U_G\ \varphi >$ to denote that φ is united knowledge of a group G of contexts. A formal definition of the united knowledge operator can be given as follows:

$$\begin{aligned}
(M,w) \models_t < U_G\ \varphi > \quad &\Longleftrightarrow \quad \exists c \in G : (M,w) \models_t < c\ \varphi > \wedge \\
&\qquad \forall c \in G : ((M,w) \models_t) < c\ \varphi >) \vee (M,w) \models_u < c\ \varphi >) \\
(M,w) \models_f < U_G\ \varphi > \quad &\Longleftrightarrow \quad \exists c \in G : (M,w) \models_f < c\ \varphi > \text{ and} \\
&\qquad \forall c \in G : ((M,w) \models_f) < c\ \varphi >) \vee (M,w) \models_u < c\ \varphi >) \\
(M,w) \models_u < U_G\ \varphi > \quad &\Longleftrightarrow \quad \forall c \in G : (M,w) \models_u < c\ \varphi > \vee \\
&\qquad \exists c_1 \in G : (M,w) \models_t < c_1\ \varphi > \wedge \\
&\qquad\ \exists c_2 \in G : (M,w) \models_f < c_2\ \varphi >
\end{aligned}$$

A group of context has united knowledge to the truth of a proposition if one of the contexts knows it is true, and all other contexts either have knowledge of its truth, or the proposition has a truth value *unknown*. A proposition is known to have a truth value *false* if a context knows it is false, and all other contexts either know it is false, or have a truth value *unknown*. For these two truth value assignments, *true* and *false*, it is relatively easy to derive the united knowledge of a group of contexts. Slightly more complex is the problem of uniting the knowledge of agents where one or more agents have evidence that a proposition is true, while at the some time one or more other agents have evidence that it is false. Roelleke (Roelleke (1999)) derives the truth value *inconsistent* for these cases, however, the semantics discussed here only allows for three truth values. Instead, we therefore say that if contexts provide conflicting information about the truth value of a proposition, its “proper” truth value is not known, and it should be considered *unknown* with respect to the united knowledge of a group of contexts. We will now define the augmentation of a context with the united knowledge of a group of contexts. In this definition, we will use $< (c_1 \cdots c_n)\ \varphi >$ instead of "$< U_G\ \varphi >$, where $c_1 \cdots c_n \in G$".

We can thus say that a context c_a is augmented with the united knowledge of $(c_1 \cdots c_n)$, or $c_a(c_1 \cdots c_n)$. Before we give a formal definition of the augmentation operator, we will discuss the process of augmentation with respect to the sample document shown in Figure 6.3. Say we want to augment context *text* with knowledge of the three subcontexts *p1,p2* and *p3*. A proposition like "Bulgaria" should be true in *text*, if either *text* knows the truth-value is *true*, and the united context *(p1, p2, p3)* gives evidence for true or unknown, or "Bulgaria" is unknown in *text*, and *(p1,p2,p3)* gives evidence for *true*. We will use the notation $A_{c_a(c_1 \cdots c_n)}$ to denote the augmentation operator applied to augment c_a with the united knowledge of $c_1 \cdots c_n$. The augmentation operator can be formally defined as:

$$
\begin{aligned}
(M,w) \models_t < A_{c_a(c_1 \cdots c_n)}\ \varphi > \quad \Longleftrightarrow \quad & ((M,w) \models_t < c_a\ \varphi > \wedge \\
& \forall w' \in R_{c_a}(w) : ((M,w') \models_t < (c_1 \cdots c_n)\ \varphi > \vee \\
& \qquad (M,w') \models_u < (c_1 \cdots c_n)\ \varphi >)) \vee \\
& (((M,w) \models_t < c_a\ \varphi > \vee (M,w) \models_u < c_a\ \varphi >) \wedge \\
& \forall w' \in R_{c_a}(w) : (M,w') \models_t < (c_1 \cdots c_n)\ \varphi >)
\end{aligned}
$$

The interpretations for *false* and *unknown* follow analogously.

For the sample document used throughout this discussion, we can therefore say that the augmented context $\text{text}_a(p_1, p_2, p_3)$ knows proposition "Bulgaria" is *true*, although it is unknown in those worlds *text* can reach in Figure 6.5, or formally:

$$(M, w_{body}) \models_t < \text{text}_a(p1, p2, p3)\ \text{Bulgaria} >$$

6.3.3.2 United and Augmented Truth Value Functions

Following the definition of the *united knowledge operator*, and the *augmented knowledge operator*, we can define the *united truth value* π_U, and the *augmented truth value* π_A. The definitions of the truth values depends on the concepts of *world-paths* and *world-trees*, which in turn are defined in terms of the *group-reachability* discussed on page 109 (Roelleke (1999)).

Using the concept of group-reachability, we can define a *group-world-path* as that sequence of worlds, which make a world w_i *group-reachable* from w_0. The *group-world-path* $(w_0 \cdots w_i)$ thus consists of those worlds $w_0 \cdots w_i$ such that there exist contexts $c \in G$, where $1 \leq c \leq i \wedge (w_{c-1}, w_c) \in R_c$.

A *group-world-tree* is the set of all such *group-world-paths* for an augmented context. For an augmented context $c_a(c_1 \cdots c_n)$ there is a group-world-tree (w,S) if $w \in R_{c_a}(w_0)$ and (w,S)

is a group-world-tree of w and the group of contexts $(c_1 \cdots c_n)$, where S is the set of subtrees of w. Similarly (w,S) is a group-world-tree for world w and group $(c_1 \cdots c_{n+1})$ if there is a set S_n such that (w,S_n) is a group-world-tree of $(c_1 \cdots c_n)$ and there exists a tree t such that t is a group-world-tree of w and c_{n+1}, and $S = S_n \cup \{t\}$.

Based on this definition of a group-world-tree, we can define a function *trees*, which produces group-world-trees as sets of trees for a context c_a and a group $(c_1 \cdots c_n)$, as:

$$\begin{aligned}
\text{trees}(w_0, c_a(c_1 \cdots c_n)) &:= \{\, (w,S) \mid w \in R_{c_a}(w_0) \wedge (w_S) \in \text{trees}(w,(c_1 \cdots c_n))\} \\
\text{trees}(w,(c_1 \cdots c_{n+1})) &:= \{\, (w,S) \mid \exists S_n, t : (w,S_n) \in \text{trees}(w,(c_1 \cdots c_n)) \\
&\qquad \wedge t \in \text{trees}(w, c_{n+1}) \wedge S = S_n \cup \{t\}\}
\end{aligned}$$

We can now define the *united* truth value in terms of a group-world-tree, and the *augmented* truth value in terms of united truth values. Suppose (w,S) is a group-world-tree of a world w and a group $(c_1 \cdots c_n)$. The united truth value π_U can be defined as the union of the truth values $\pi(w')$ where all w' are the roots of subtrees $(w',\{\})$ in S:

$$\pi_U((w,S)) \quad := \bigcup_{(w',\{\})\in S} \pi(w')$$

The truth value π_U is unknown for an empty set S.

We can now define the augmented truth value π_A by referring to the united truth value π_U. Suppose (w,S) is a group-world-tree of a world w_0 and an augmented context $c_a(c_1 \cdots c_n)$. The augmented truth value $\pi_A((w,S))$ of a group-world-tree (w,S) is defined as the union of the truth value of world w, $\pi(w)$, and the united truth value π_U of the group-world-tree (w,S), $\pi_U((w,S))$:

$$\pi_A((w,S)) \quad := \quad \pi(w) \cup \pi_U((w,S))$$

This concludes our discussion of the non-probabilistic knowledge augmentation. We will now extend this discussion to include probabilities.

6.3.3.3 Probabilistic Knowledge Augmentation

Where the non-probabilistic augmentation of knowledge concerns only the truth of a proposition in an augmented context, the probabilistic augmentation operation needs to consider the probability with which propositional truth values hold in those contexts uniting their knowledge. For example, in the previous examples we showed that the augmented context $text_a$ has knowledge of the proposition “Bulgaria”. However, in the case of probabilistic propositions, what should the probability of “Bulgaria” being *true* be, given its probabilities in the contexts uniting their knowledge?

Roelleke (Roelleke (1999)) outlines three possible ways in which augmented probabilities might arise in POOL, his proposed logic for hypermedia retrieval:

- non-probabilistic access and probabilistic content: the knowledge of which worlds a context accesses is certain; only the content of possible worlds is uncertain. This is the case considered here for structured documents and POLIS expressions
- probabilistic access and non-probabilistic content: the knowledge of propositions in contexts is certain, and only the access is uncertain
- probabilistic access and probabilistic content: both the knowledge of contexts and the worlds accessible are uncertain

We will only consider the first case here, since neither structured documents (specifically XML documents) nor POLIS expressions offer the option of uncertain access to contexts. Before we will provide formal definitions of the probabilistic knowledge augmentation operator, we need to provide a sample document which we can use to highlight the augmentation process.

```
<doc>
<body>
 <text>
   <p>
     [...] 0.4 Bulgaria [...]
   </p>
   <p>
     [...] 0.7 Bulgaria [...]
   </p>
 </text>
</body>
</doc >
```

Figure 6.6: The sample document in Figure 6.4 amended for discussing probabilistic knowledge augmentation.

This sample document is shown in Figure 6.6. It is an extension of the document displayed in Figure 6.4; here, a second paragraph has been added. This second paragraph also has knowledge of the proposition "Bulgaria", with a probability of 0.7. Table 6.5 shows the probabilities we obtain with respect to the possible worlds.

We furthermore know that in the unaugmented context *text* the proposition "Bulgaria" is unknown with a probability of 1.0. Additionally, for the contexts *text, p1* and *p2*, we can derive the following accessibility relations:

world w'	$\mu_{p2,w_{text}}(w')$	proposition φ	$\pi(w')(\varphi)$
$w_{p1,1}$	0.4	Bulgaria	true
$w_{p1,2}$	0.6	Bulgaria	unknown
$w_{p2,1}$	0.7	Bulgaria	true
$w_{p2,2}$	0.3	Bulgaria	unknown

Table 6.5: Probabilities of possible worlds for probabilistic truth value assignment of proposition "Bulgaria" in two paragraphs.

$$\begin{aligned} R_{text} &= \{(w_{body}, w_{text})\} \\ R_{p1} &= \{(w_{text}, w_{p1,1}), (w_{text}, w_{p1,2})\} \\ R_{p2} &= \{(w_{text}, w_{p2,1}), (w_{text}, w_{p2,2})\} \end{aligned}$$

To define the probabilistic united and augmented truth values, we first extend the notion of a group-world-tree to that of a *group-context-world-tree*. The inclusion of a context into the tree is necessary, as probability assignments are only defined for a world in conjunction with a context. A group-context-world-tree is a group-world-tree where the single world w is replaced by the tuple (c,w), where c is a context accessing w. A group-context-world-tree is thus of the form $((c,w),S)$. We can now define the united probability assignment with respect to a group-context-world-tree, following the definition of the united truth value provided on page 112.

Suppose $((c,w),S)$ is a group-context-world-tree of world w and the group $(c_1 \cdots c_n)$. We define the united probability $\mu_U(((c,w),S))$ of the tree ((c,w),S) as the product of all probabilities $\mu_{c,w}(w')$ where (c,w') are the roots of subtrees ((c,w'),{}) in S:

$$\mu_U(((c,w),S)) \quad := \prod_{((c.w'),\{\})\in S} \mu_{c,w}(w')$$

The united probability for an empty set S is defined as 1.0.

Following the definition of the united probability assignment, we can define the augmented probability assignment with respect to group-context-world trees, and the definition of the united probability assignment. The augment probability assignment follows the definition of the augmented truth value assignment.

Suppose $((c,w),S)$ is a group-context-world-tree of a world w_0 an an augmented context $c_a(c_1 \cdots c_n)$. The augmented probability μ_A, w_0(((c,w),S)) for a tree ((c,w),S) is defined as the product of the supercontext probability $\mu_{c,w_0}(w)$ and the united probability $\mu_U(((c,w),S))$ of the group-context-world-tree ((c,w),S):

$$\mu_{A,w_0}(((c,w),S)) \quad := \quad \mu_{c,w_0}(w) \times \mu_U(((c,w),S))$$

6.3.3.4 Example

To show show the probabilistic augmentation process working when applied to a document, we will now discuss the augmented reasoning when applied to the sample document in Figure 6.6. We will discuss the augmentation of knowledge with respect to the context *text*. We will start the discussion by outlining all the possible world trees which can be derived for sample document. For the augmentation of *text*, we can derive four possible world trees: one in which *p1* enters a world where "Bulgaria" is known to be true, one where it is unknown, one where *p2* enters a world where "Bulgaria" is known to be true, and one where it is unknown. Figure 6.7 displays all possible trees.

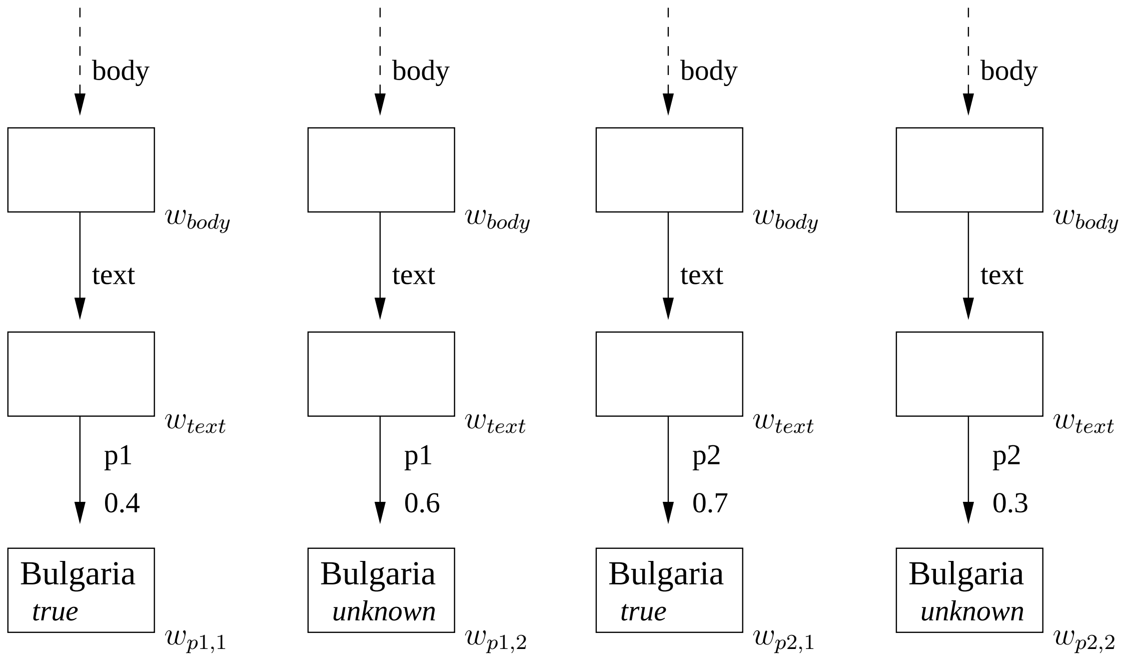


Figure 6.7: Probabilistic World Trees generated from sample document in Figure 6.6.

Now consider four worlds w_1, w_2, w_3, w_4, which correspond to the possible world trees. For these worlds, we can derive the following probabilities for the proposition "Bulgaria":

w'	$\mu_{\text{text}(p_1,p_2),w_{\text{body}}}(w')$	φ	$\pi(w')(\varphi)$
w_1	0.4×0.7	Bulgaria	*true* ∪ *true* = *true*
w_2	0.4×0.3	Bulgaria	*true* ∪ *unknown* = *true*
w_3	0.6×0.7	Bulgaria	*unknown* ∪ *true* = *true*
w_4	0.6×0.3	Bulgaria	*unknown* ∪ *unknown* = *unknown*

For the augmented context $\text{text}_a(p_1, p_2)$ and the proposition "Bulgaria", we thus yield the following interpretation:

$$(M, w_{body}) \models_t < \text{text}_a(p_1, p_2)\ 0.82 \text{ Bulgaria} > \iff 0.82 = \mu_{\text{text}_a(p_1,p_2),w_{body}}(\{w' \mid w' \in R_{\text{text}_a(p_1,p_2)}(w_{body}) \wedge (M, w') \models_t \text{Bulgaria}\})$$

Similarly, "Bulgaria" is *unknown* in the augmented context *text* with a probability of 0.18.

6.3.3.5 *Derivation of augmented probabilities via event spaces*

Instead of obtaining the probabilities we saw by traversing the possible world trees, it would also be possible to consider probabilistic terms in contexts as events. To assign a probability to an event in an augmented context, it would be necessary to combine events in those subcontexts forming the group which unites its knowledge. The presence or absence of terms in contexts, and also the probability of observing such an event, would be *independent*. The presence of a term in a context, and the probability of observing the term in the context, does not influence the probability of observing the term in another context. To derive the probability of observing an event in the augmented context, we need to unite the associated event probabilities under an *independence* assumption. We can achieve this by applying the *inclusion-exclusion principle* (or *sieve principle*) to the set of events.

The general definition of the inclusion-exclusion principle for two events A and B states that the probability of observing A or B is defined as the sum of the event probabilities, minus the product of the event probabilities:

$$P(A \vee B) = P(A) + P(B) - P(A) \times P(B)$$

For the sample document in Figure 6.6, we have two events: observing "Bulgaria" in p_1 with a probability of 0.4, and observing "Bulgaria" in p_2 with a probability of 0.7. Applying the inclusion-exclusion principle, we can derive the probability of observing "Bulgaria" in *text* as:

$$\begin{aligned} P(\text{Bulgaria}) &= P(\text{Bulgaria}, p_1) + P(\text{Bulgaria}, p_2) - P(\text{Bulgaria}, p_1) \times P((\text{Bulgaria}, p_2) \\ 0.82 &= 0.4 + 0.7 - (0.4 \times 0.7) \end{aligned}$$

However, deriving the augmented probabilities of propositions using events is not sound with respect to the definition of *false* or *unknown*: the augmented probabilities in *text* do not form a probability mass function. While we derive the correct probability for the assignment of *true* to the proposition "Bulgaria" in *text*, we would also derive a probability of 0.72 for *unknown*:

$$0.72 = 0.6 + 0.3 - (0.6 \times 0.3)$$

We will not make use of this derivation of augmented probabilities in terms of event spaces for the remainder of this chapter, however, it will become more important in chapter 6.4, where we discuss the implementation of POLIS. There, it is acceptable to derive augmented probabilities via the inclusion-exclusion principle, as neither *false* nor *unknown* can be explicitly assigned to terms. Instead, we will define the probability of *unknown* as the counterprobability of *true*, which yields a probability assignment equivalent to the possible worlds derivation outlined above.

This concludes our discussion of the interpretation propositions in structured documents, and the augmentation of context knowledge in document hierarchies. We will now turn to the interpretation of POLIS expressions.

6.3.4 Semantics of POLIS expressions

The previous sections discussed how knowledge and probabilities in structured documents arise, and how it is possible to reason about this knowledge, to carry out processes such as knowledge augmentation. This discussion directly relates to the interpretation of POLIS expressions, as propositions in POLIS expressions (in the form of query terms) can be interpreted in the same way as in structured documents.

The main difference between interpreting a POLIS expression and interpreting a structured document is the extent to which structure is considered. In structured documents, reasoning is carried out largely context local, where the validity of a proposition is only considered with respect to the context to which it is local. The only exception to this is the process of knowledge augmentation, where the knowledge of some context influence the knowledge of other contexts, such that when considering the *hierarchical* nature of structured document, knowledge is pushed *up* the document tree. Structural requirements are not enforced otherwise. However, POLIS expressions are modelled on XPath in such a way that the individual contexts which form an axis in the POLIS expression are nested. So for any two contexts separated by a forward-slash ("/") in the POLIS expression, it is assumed that the latter context is a *subcontext* of the former one. We need to formalise this notion of structure within the possible worlds framework outlined previously.

Additionally, we need to look at the interpretation of knowledge of the last context given in a POLIS expression. This last context denotes those structural elements which are supposed to form the summary, however, in the case that these contexts have subcontexts of their own, and do no contain any propositions, all propositions would be unknown; it would thus not be possible to apply the retrieval models discussed in a later chapter. Instead, we define that the knowledge of the last context denoted in a POLIS expression is always augmented by the knowledge of all its subcontexts. We will formalise this notion of "all subcontexts".

6.3.4.1 Context Nesting

The syntax of POLIS is modelled on the XML retrieval language XPath, in which the path expression denotes all those elements which should be retrieved. Polis follows this structural path interpretation as a denotation of all those elements which should form the summary.

The path expression can be derived from two simple definitions: a path is a node, or a node followed by a path. A node here is any structural element, such as the "text" elements seen in the previous examples. Syntactically, a path can be defined by:

$$\begin{array}{lll} \text{path} & := & \text{node} \mid \text{path} \\ \text{path} & := & \text{node} \end{array}$$

Informally, the meaning of a path expression can be interpreted such that every node n_1, which is preceded by a node n_0, is a subcontext of n_0. We will now formally define this informal interpretation, using the possible worlds framework. We will use the accessible relations of context to define the *seriality* of contexts, which corresponds to the nesting of contexts in XPath expressions.

We define the seriality of contexts as the property that for any world w', which is accessible to a context c from a world w, every subcontext sc of c has a world w'' in its accessibility relation, which it can reach from w'. Such a subcontext sc is context-serial with respect to c. We will use the notation $\mathrm{CS}(sc,c)$ to denote that sc is context-serial with respect to c. This can be defined formally as:

$$\mathrm{CS}(sc,c) \quad \Longleftrightarrow \quad \forall w \in W : \forall w' in R_c(w) : \exists w'' : w'' \in R_{sc}(w')$$

A POLIS expression is valid, if every node n in the path-expression is context serial with respect to its predecessor node in the path. The path expression also serves to identify elements in the context to be summarised. An element in the context to be summarised is considered for forming the summary if there is a path from the root the element in the context's world tree, which corresponds to the path given in the POLIS expression. We can define this more formally, saying an element e is summary forming if its name occurs in the POLIS expression and in the world tree, and if it is context serial with respect to an element e' in the POLIS expression, it is context-serial with respect to e' in the world tree, and there are no elements e_s which are context serial with respect to e in the POLIS expression. We can formalise this notion as:

$$\begin{aligned} e \text{ is summary-forming} \quad \Longleftrightarrow \quad & e \in \text{POLIS exp.} \rightarrow e \in \text{world-tree} \wedge \\ & \exists e' : \text{CS}(e,e') \text{ in POLIS exp.} \rightarrow \\ & \quad \text{CS}(e,e') \text{ in world-tree} \wedge \\ & \neg\exists e_s : \text{CS}(e_s,e) \text{ in POLIS exp.} \end{aligned}$$

6.3.4.2 Summary Element Augmentation

The interpretation of nested contexts in the POLIS path expression points to those elements in a structured context which will form the summary. However, it is possible that these elements do not have any textual content, but a mere structural elements with nested subcomponents. For example, a generic document might be broken down by subelements such as *section, paragraph*, or *sentence*. Assuming this schema, it would be unusual the observe text content at any but the sentence level, however, it would be perfectly acceptable for users to provide a POLIS expression, such that the summary should be generated from paragraphs. To allow this, we make the implicit assumption that the summary forming elements specified by a POLIS expression are augmented with the knowledge of all their subelements.

The augmentation process thus transitively closes the world-tree for the augmentation process; a summary-forming element is augmented with the knowledge of its subelements, as well as its subelements' subelements, etc.

6.3.5 Summary

In this section, we have developed a possible worlds semantics for POLIS. Having a well defined semantics is important for a logic, in order to verify translations of logical expressions into other logical languages, or to show the correctness of systems implementing the logic. The semantics developed here allows us to reason about the knowledge of structured documents, in order to carry out processes such as the augmentation of a context's knowledge. We have extended this possible worlds semantics to include probabilities, to allow for uncertain reasoning, which is an integral part of information retrieval.

Furthermore, we looked at how the nested structure of contexts represented in a POLIS expression can be reflected in the accessibility relations the possible worlds framework, and how a path in a POLIS expression corresponds to a path in the world-tree, pointing to all those elements which should be considered for inclusion in the summary. Additionally, we looked at the knowledge augmentation of these summary elements with knowledge of their subparts. This concludes our discussion of the POLIS semantics.

6.4 Implementation of POLIS

In the previous two sections we discussed the syntax of POLIS, and presented a formal semantics for the summarisation logic. In this section, we will continue the specification of POLIS, by showing how POLIS, and the summarisation models we developed in chapter 5, can be implemented.

The implementation of POLIS further emphasises the flexibility and adaptability of a summarisation logic; POLIS expressions are not evaluated (read: processed) by a closed-source, single-platform application, but are instead translated into a further intermediate language, which can then be processed on any suitable target retrieval platforms. Implementing POLIS by translating it into another target language is a lightweight, elegant implementation approach, which allows quick and easy modification of the implementation if changes to POLIS are necessary. This implementation approach also more closely follows the methodology of POLIS as a lightweight intermediate abstraction layer, as opposed to a closed-box implementation of POLIS in Java or C. Furthermore, any benefits of the lower layers (such as performance improvements of the PRA execution layer) will be directly available to POLIS

The intermediate language we chose for POLIS is Probabilistic Relational Algebra (PRA) (Fuhr and Rölleke (1997)). PRA is a probabilistic extension to the traditional relational algebra (RA) employed by Database Management Systems (DBMS). Like traditional RA, PRA uses *relations* as a data representation to store knowledge. However, unlike in traditional RA, tuples in PRA are given a probability, and relational operators manipulate not only the contents of relations, but also derive probabilities for tuples in the resulting relations. In PRA, the probability of a tuple denotes the likelihood that the tuple is part of its relation. In the traditional deterministic RA, a tuple is always either part of its relation (in which case its probability is 1.0), or it is not (in which case its probability is 0.0). The probabilities introduced in PRA can be used to express a degree of uncertainty about the represented knowledge. Fuhr, Roelleke *et al.* have shown how it is possible to exploit this uncertainty-modelling to implement IR retrieval models in PRA, and thus to achieve an integration of information retrieval and database systems (Fuhr and Rölleke (1997); Roelleke et al. (2008)).

We will use this integration of DB and IR for implementing POLIS, by translating POLIS expressions into equivalent PRA statements. This ensures that POLIS is based on solid probabilistic and information retrieval foundations, and additionally allows a close integration with existing DBIR systems. The close integration with DBMS additionally allows POLIS to make use of the efficiency of such systems. However, before we can investigate how our summarisation models can be implemented in PRA, we need to provide additional details of the data representation used. As we pointed out, probabilistic relational algebra uses relations as a means of representing data. We therefore need a relational schema that allows us to capture both the structure as well as the content of structured documents in a set of relations. We will therefore start by providing some background on relational data representations, and the process of data retrieval from relations. We then discuss how structured documents can be fully represented by a specific set of relations.

6.4.1 Foundations of DB Systems and Relational Data

Knowledge in PRA is represented by relations. A relation can be thought of as a table, where individual rows in the tables, the *tuples*, are the primary information carriers. Each tuple consists of a set of *attributes*, which are the most atomic units of information. In higher-level data representation languages, such as SQL, attributes can be assigned names, and the information contained in an attribute can be accessed via the attribute's name. However, for both RA and PRA, attributes are simply identified by a numerical index, denoting their position within a tuple. All tuples in one specific relation consist of the same number of attributes, known as the *schema*. A relation can thus be defined as an $n \times m$ matrix of n tuples and m attributes.

A DBMS implementing RA will carry out retrieval on the relational data using a basic set of atomic relational operations: SELECT, PROJECT, JOIN, UNITE, and SUBTRACT (Elmasri and Navathe (1999)). While these operations are deterministic for RA, PRA-variants of these operations need to include probabilities as well. The actual probability estimations carried out by the PRA operators depend on probabilistic *assumptions*, which we will discuss in more detail later on. Additionally, PRA introduces an entirely new operator, the relational BAYES, to derive probabilities from a given (deterministic) relation (Roelleke et al. (2008)).

We mentioned in the previous section that one advantage of modelling POLIS in PRA is the benefit of using the efficiency of a typical DBMS. Efficiency gains associated with database systems can usually be attributed to the use of an index. An index is created for specific attributes (i.e. columns) of a relation, and provides information about the positions of specific attribute-values in the entire relation. The actual performance improvements gained from using an index depend on the type of atomic operation carried out; while a projection on a relation can be implemented efficiently without an index (assuming that no DISTINCT criterion is specified), operations such as a SELECT or JOIN will greatly benefit from locational information about tuples meeting specific attribute values (Connolly and Begg (2001)). For the implementation of POLIS discussed here, and for the efficiency and effectiveness figures reported in the later chapter on evaluation, we assume that indices are present for all attributes used in JOIN, PROJECT, or SELECT operations.

6.4.2 Relational Representation of Structured Documents

In order to be able to make use of the benefits of a relational data representation, we first need to define a relational schema which completely captures all the properties of structured documents: content, structure, and information about attributes and element-types. This relational schema follows the probabilistic object-oriented content representation (POOCR) proposed in (Roelleke (1999)).

Although POLIS is a summarisation logic for *structured documents*, we do not make any assumptions about the structural complexity or the contents of the documents to be summarised. A document could be a flat, unstructured file (i.e. the document consists of a single context only) with plain text, it could feature structured context with plain text only, make use of structure and attributes (such as XML documents), or feature a full set of structured contexts, attributes, classifications and relations between objects (such as POOL documents).

The relational schema used to represent documents should be flexible enough to accommodate all these different complexity classes equivalently. However, this requirement can be easily satisfied, as lower document complexity classes are entirely subsumed by the higher classes: unstructured documents can be thought of as structured documents with a singular context, structured documents are POOL documents without classifications and relations etc. A relational schema sufficient to represent highly complex structured documents is thus also able to capture less complex documents. The most complex structured documents we will consider here are POOL documents, which can represent structure and content like standard XML documents, and additionally support concepts found in object oriented domains, such as classifications, and relationships between objects (Roelleke (1999)).

A relational schema for storing structured documents would thus need to provide facilities to accommodate all of these concepts. Representing all the information contained in structured documents can be achieved by a set of four relations: *term*, *instance_of*, *part_of*, and *attribute*.

The most important – and also most simple – of these relations is *term*, which is used to represent the textual content of contexts. *term* consists of two attributes: *term*, and *context*, where the former attribute holds individual words (*terms*), while the latter holds their respective context. A tuple in this relation represents a term occurring in a context, e.g. a word occurring in a document section, such that a set of tuples holds the entire textual content of specific contexts. Terms in the *term* relation correspond to propositions in the semantics.

The second relation, *instance_of*, represents classifications of objects. With respect to structured documents, it assigns meaning to subcontexts that appear in structured documents. For example, the instance_of relation could provide information that the subcontext "introduction" is of type *introduction* in the structured document "doc1". To do so, the relation consists of three attributes: *object*, which is the object to be classified, *class*, providing the class an object belongs to, and *context*, which is the context in which the classification information is valid. Classifications are propositions, similar to terms. Therefore, the truth value of a classification might differ in different contexts. This is reflected by the *context* attribute.

The *part_of* relation is used to represent the structure of documents, i.e. the hierarchical layout of documents. It uses three attributes to represent this information: *subpart*, *superpart* and *context*. A tuple in the *part_of* relation is again a proposition, although it will not be interpreted as such by the semantics. Any proposition in the relation states that the object denoted in the attribute *subpart* is a part of the object denoted in the attribute *superpart*, and is valid in the context provided in attribute *context*.

Finally, the last feature of structured documents to be represented in the relational schema are *attributes*. An attribute can be an actual attribute inside a tag, as found in XML or XHTML documents, or can be a more complex object, such as a named relation between objects. For representing the value of attributes, the *attribute* relation makes use of four columns: *type*, *object*, *value* and *context*. The *type* attribute specifies the type of an attribute, which for XML documents corresponds to the attribute name. The *object* attribute defines the object to which the attribute is attached, which in the domain of structured documents is a context. The value of a specific attribute attached to an object is given by *value*. Again, the assignment of values to attributes is valid only in a specific context, given by the *context* attribute.

This relational schema of four relations is sufficient to represent all complex, structured documents in a coherent relational framework. Without any further changes to the relational schema, documents of varying complexity and structure can uniformly be mapped onto the set of relations. To help make this brief and abstract description of mapping structured documents into a relational schema more concrete, we will now give an example of how a structured document is translated into the given set of relations.

6.4.2.1 Example

To show how an actual document can be represented using the set of relations outlined above, we will now detail the translation of a sample structured document into its corresponding relations. The sample structured document we use is a simple XML document consisting of three subcontexts, shown in Figure 6.8.

```
<doc id=1>
  <sec id=1.1> some text </sec>
  <sec id=1.2> boats greece </sec>
  <sec id=1.3> sailing east coast </sec>
</doc>
```

Figure 6.8: A sample XML document.

The sample document has a root-context of type “doc”, which consists of three subcontexts of type “sec”. Each of the contexts has an attribute called “id”, which has a numerical value specifying a unique identifier for all contexts of the same type. Additionally, the document holds some textual content: each of the three “sec” contexts contains some text. The first “sec” context contains the terms “some text”, the second “sec” context consists of the terms “boats greece”, and the last “sec” context contains the terms “sailing east coast”.

A relational representation of this sample document needs to reflect the structural layout of the document, the textual content of the contexts, and the types and values of its attributes. Figure 6.9 depicts the relational representation of the sample document shown in Figure 6.8.

Figure 6.9 shows that the relational representation reflects both the structure and content of the sample document in the way outlined in the previous subsection. Additionally, the *attribute*-relation contains an additional attribute, ”serialID”. This attribute is not present in the original document, and is an artifact introduced by the indexing process. The attribute assigns a unique identifier to contexts belonging to the same type – irrespective of whether such an ID is present in the original structured document or not – and is necessary for some of the querying facilities provided by POLIS. This will be discussed in more detail in the next section, where we show the translation of POLIS expressions into equivalent sets of PRA expressions. Additionally, we also attribute

type	object	value	context
id	/doc[1]	1	/
id	/doc[1]/sec[1]	1.1	/doc[1]
id	/doc[1]/sec[2]	1.2	/doc[1]
id	/doc[1]/sec[3]	1.3	/doc[1]
serialID	/doc[1]	1	/
serialID	/doc[1]/sec[1]	1	/doc[1]

part_of

subcontext	supercontext
/doc[1]	/
/doc[1]/sec[1]	/doc[1]
/doc[1]/sec[2]	/doc[1]
/doc[1]/sec[3]	/doc[1]
⋮	⋮

instance_of

object	class	context
/doc[1]	doc	/
/doc[1]/sec[1]	sec	/doc[1]
/doc[1]/sec[2]	sec	/doc[1]
/doc[1]/sec[3]	sec	/doc[1]
⋮	⋮	⋮

term

term	context
some	/doc[1]/sec[1]
text	/doc[1]/sec[1]
boats	/doc[1]/sec[2]
greece	/doc[1]/sec[2]
sailing	/doc[1]/sec[3]
⋮	⋮

Figure 6.9: Relational representation of the document in Figure 6.8.

provide corresponding Probabilistic Datalog (PD) expressions, to aid understanding of the PRA statements.

6.4.3 Translation of POLIS into equivalent PRA expressions

Using the detailed description of how structured documents can be represented by a specific set of four relations, we can now discuss the translation of a POLIS expression into a series of equivalent PRA statements. To show the equivalence of the generated set of PRA expressions with the original POLIS expression, we compare the semantics of POLIS, and the probabilistic summarisation models discussed in Chapter 5, with the semantics of the PRA operators. The semantics of PRA expressions are discussed in detail in (Roelleke (2009)). We will only repeat the most important definitions here, to aid readability of the following section.

All PRA expressions produce probabilistic relations. A probabilistic relation is a pair (T,P), where T is a set of tuples, and P is a function that assigns a probability to tuples: $T \rightarrow [0;1]$. An individual tuple in T is denoted by the letter τ, and attributes in a tuple are addressed by their respective position. The first attribute in τ is thus addressed as $\tau(1)$.

The meaning of PRA operators can be expressed as the changes the operators perform on sets of tuples; we will therefore use a set-theoretic notion to present the semantics of PRA operators. The semantics of an entire PRA expression is given by the chain of PRA operators from which it is built up, working from the innermost PRA operator to the outermost. The output of inner PRA operators is fed as input to outer operators. Using these definitions, we can now detail the semantics of POLIS-generated PRA expressions.

6.4.3.1 Path to Summary-forming Elements

We will start the translation by first extracting the user specified summary forming elements. POLIS steps sequentially through a given logical expression to define an intensional relation that contains all those elements at the user specified granularity. If possible, intermediate relations created to perform this operation will have names based on the element types they are referring to. Additionally, a sequence number is used to give relations unique names. The translation process will also introduce variables, which are selected in alphabetical order. These variables follow the convention of capitalising variable names.

The first element in a POLIS expression is always a "/", followed by either a "*" (where the type of element is not specified, and is thus matched by any element), or by a specific element type. There are thus two cases to be considered for the first translation step.

The sequence "/*", for any element types, can be expressed in PRA by the statement

```
p0 = Project INDEPENDENT[$1](Select[$3="/"](instance_of));
```

The relation *p*0 thus holds all those elements which are subparts of "/". Note that the selection is performed via the *instance_of* relation here, although it would have been possible to carry out the selection equally well on the *part_of* relation.

We start with the semantics of the inner SELECT statement, the result of which is input to the outer PROJECT expression. The semantics of the SELECT are:

$$\begin{aligned} (T,P) &= \text{'SELECT' '[' condition ']' '(' relation ')'} \\ T &:= \{\tau | \tau \in T_{\text{relation}} \wedge \tau(3) = \text{"/"}\} \\ P(\tau) &:= P_{\text{relation}}(\tau) \end{aligned}$$

where "condition" is "$3="/"", and "relation" is set to *instance_of*. The SELECT returns a relation of three attribute tuples, where the third attribute contains the value "/". The tuple probabilities in the resulting relation are equal to the probabilities in the input relation. This relation resulting from applying the SELECT to the *instance_of* relation is used as input to the Project INDEPENDENT. The "independent" keyword indicates that the project operator makes an independence assumption with respect to the tuples in its input relation, and that the output is going to be a distinct relation. All tuples in the input relation which are non-distinct for the given attribute target-list are combined into a single tuple in the output relation. The probabilistic assumption affects the way in which probabilities in the input relation are aggregated for tuples in the resulting relation.

The semantics of the Project INDEPENDENT can be given as

$$\begin{array}{rcl} (T,P) & = & \text{'PROJECT INDEPENDENT' '[' targetlist ']' '(' relation ')'} \\ T & := & \{\tau | \tau' \in T_{\text{relation}} \wedge \tau = \tau'[\text{targetlist}]\} \\ P(\tau) & := & 1 - \prod_{\tau_a \in T_a \wedge \tau_a[\text{targetlist}] = \tau[\text{targetlist}]} (1 - P_a(\tau_a)) \end{array}$$

where "relation" is the output of the inner SELECT, and "targetlist" consists of a single attribute, here 1. Both the *instance_of* and the *part_of* relations are deterministic, so that the given assumption INDEPENDENT for the projection reduces to a traditional DISTINCT projection, where all tuple probabilities are 1. The result of the entire PRA expression is thus a distinct relation consisting of all those tuples in *instance_of* where the third attribute has the value "/".

A corresponding expression in PD would be:

```
p0(A) :− instance_of (A,X,"/");
```

where the second variable "X" of *instance_of* is only necessary due to the limited expressiveness of PD, and simply signifies that the second attribute of *instance_of* can take any value.

Similarly, the translation for a POLIS expression starting with "/*element-type*" can be translated into the PRA expression

*element-type*0 = **Project INDEPENDENT**[$1]
    (**Select** [$2="*element-type*", $3="/"](instance_of)) ;

The only difference between the two PRA expressions is the additional condition in the inner SELECT operator, such that the overall output of the expression is a distinct relation of all tuples in *instance_of* where the third attribute has a value of "/", and the second attribute has the value *element-type*.

The corresponding PD expression to the above PRA statement is

*element-type*0(A) :− instance_of (A,*element-type*,"/")

which is similar to the Datalog expression for the unspecified element type, where here the "don't care" variable "X" of the previous expression has been replaced by the given element-type.

The remaining path expression in a POLIS statement is translated into PRA by iteratively creating new relations for each element in the path, which refer to previously created element-relations via the *part_of* relation. Again, an element in the path can be either a "*", or a specific element-type. For the "*"-expression, the according PRA statement is

```
p1 = Project INDEPENDENT[$2](Join[$1=$2](p0,part_of));
```

where we assume the above expression was generated for the second element in a path (indicated by the element sequence number 1), and where $p0$ is given exemplary and is generally the path-element relation created for the previous path element. The semantics for the JOIN operation are defined as

$$\begin{array}{rcl}(T,P) & = & \text{JOIN '[' condition ']' '(' rel}_a\text{, rel}_b\text{ ')'} \\ T & := & \{(\tau_a[i_1\cdots i_n],\tau_b[j_1\cdots j_n])|\tau_a\in \text{rel}_a \wedge \tau_b \in \text{rel}_b \wedge \forall [i_1\cdots i_n],[j_1\cdots j_n] : i_* = j_*\} \\ \forall\tau\in T : P(\tau) & := & P_a(\tau[i_1\cdots i_n])\cdot P_b(\tau[j_1\cdots j_n])\end{array}$$

The semantics for the PROJECT operation are identical to the above example. The PRA expression thus produces a distinct relation such that it consists of all those elements which are subelements of those path-elements in the previously generated relation (here $p0$). The corresponding PD expression follows the PRA statement, and joins the previously generated relation with the *part_of* relation:

```
p1(A) :- p0(B) & part_of (A,B).
```

The expression for a named element type is very similar, however, additionally makes use of the *instance_of* relation, such that only elements of the given element-type are included in the resulting relation:

```
element-type1 = Project INDEPENDENT[$2](Join[$2=$1]
        (Join[$1=$2](p0, part_of ), Select [$2="element-type"]( instance_of )));
```

Here we assume this is the second step in the processing (visible by the relation sequence counter "1"), and also assume that the first step in the path was "*", so that the JOIN via the "part_of" relation is performed with "p0".

Accordingly, the corresponding PD expression makes use of the previously generated named relation

```
element-type1(A) :− p0(B) & part_of (A,B) & instance_of (A, element-type ,X).
```

Iteratively applying this process to the POLIS expression until a summary definition is encountered will result in a relation which contains all the elements at the user-specified level of granularity, which could potentially form the summary and need to be processed by the weighting models. We will call this final relation *elements* in the sequel. The nesting of contexts, seen here by the inclusion of previously generated relations into the definition of later relations, corresponds to the context-nesting and context-seriality requirements discussed in Section 6.3.4.1.

The main processing step in the generation of summaries is the ranking of elements according to the weighting models. Before this can take place, it is necessary to include terms in subparts of those elements into the set of terms for elements themselves. For example, if summarisation were to take place at sentence level, the sentences should contain all its own terms, but additionally also knowledge about terms in sub-elements, such as layout tags (e.g. italicised text). This process is called knowledge augmentation, and will be covered in the next section.

6.4.3.2 Knowledge Augmentation

In chapter 5, we noted that the weighting of terms in contexts is augmented by the presence of terms in those contexts' subparts. More generally, the *knowledge* of contexts is *augmented* with the knowledge of subcontexts. The knowledge augmentation of elements was discussed in Chapter 4, and is traditionally modelled via a "link" relation, such that a certain context is "about" a term if either that term occurs in the context, or it occurs in a context linked to the current one (shown here with an example in Probabilistic Datalog):

```
about(T,D) :− term(T,D);
about(T,D) :− link (D,D'), about(T,D');
```

We follow this idea of modelling context augmentation via a link relation, however, since dealing with structured documents, replace the "link" relation with the "part_of" relation denoting the structural layout of documents. Since subcomponents of an element could themselves contain additional subcomponents, we need to traverse the *part_of* relation transitively. To make this more efficient, we assume that the transitive closure of *part_of* has been materialised, and is available to the system as a relation called *part_of_trans*. However, as we discussed in Chapter 4, the process of knowledge augmentation can be performed in two ways: as an aggregation of term *probabilities*, or an aggregation of term *frequencies*, from which the probabilities are derived.

To augment the term frequencies (i.e. the number of occurrences of a term) in a context, it is necessary to collect term frequencies of all subcontexts, and make them local to the augmented context. We can define this knowledge augmentation of a context as:

```
element_about = PROJECT ALL[$4,$1](JOIN[$2=$2](JOIN[$1=$2]
        (elements, part_of_trans ), term));
```

The semantics of the JOIN operations have been discussed previously. The PROJECT operation with the assumption ALL is a non-distinct projection, and has the following semantics:

$$\begin{array}{rcl} (T,P) & = & \text{'PROJECT ALL' '[' targetlist ']' '(' relation ')'} \\ T & := & \{\tau | \tau' \in T_{\text{relation}} \wedge \tau = \tau'[\text{targetlist}]\} \\ \forall \tau \in T : P(\tau) & := & P_{\text{relation}}(\tau) \end{array}$$

The projection thus does not affect tuple probabilities, but only limits the number of attributes present in tuples. This resulting relation of the entire PRA expression is thus a collection of all terms which occur in a context, or in any subcomponent of this context. To weight the terms in such an augmented context, the now augmented term frequencies would be used as a basis for estimating term probabilities. This term-frequency augmentation can be expressed in Probabilistic Datalog as:

```
element_about(T,E) :- elements(E) & part_of_trans (A,E) & term(T,A).
```

A second option for augmenting the knowledge is to first weight terms in all the subcontexts of the context to be augmented, and then to aggregate the probabilities to form term weights in the augmented context.

```
element_terms = Project[$2,$3](JOIN[$1=$2](elements, term));
p_t_d = Project DISJOINT(Bayes[$2](element_terms));
weighted_element_about = PROJECT INDEPENDENT[$4,$1](JOIN[$2=$2](JOIN[$1=$2]
        (elements, part_of_trans ), p_t_d));
```

The second augmentation method makes use of a new relational operator, the relational BAYES, introduced in (Roelleke et al. (2008)). The semantics of the BAYES can be given as

$$P(\tau) \quad := \quad \frac{P(\tau)}{\sum(\{P(\tau')|\tau'[i_1 \cdots i_n] = \tau[i_i \cdots i_n]\})}$$

The Bayes thus weights tuples by their fraction of the overall relation. For a distinct relation such as term, the weight of a tuple is $\frac{1}{L_R}$, where L_R is the length of the relation. The weighted relation produced by the Bayes is given as input to the outer PROJECT, which makes a DISJOINT assumption. The semantics for the DISJOINT are

$$\begin{aligned} (T,P) \quad &= \quad \text{'PROJECT DISJOINT' '[' targetlist ']' '(' relation ')'} \\ P(\tau) \quad &:= \quad \sum_{\tau_{\text{relation}} \in T_{\text{relation}} \wedge \tau_{\text{relation}}[i_1 \cdots i_n] = \tau[i_1 \cdots i_n]} P_{\text{relation}}(\tau_{\text{relation}}) \end{aligned}$$

The entire PRA expression thus produces a distinct relation P_t_d, where individual tuples are weighted by the sum of their Bayes probabilities. As the name indicates, these weights are linear occurrence probabilities, implement $\frac{n_L(t,d)}{N_L(d)}$, where $n_L(t,d)$ is the number of occurrences of term t in document d, and $N_L(d)$ is the total number of terms in d, which correspond to the traditional formulation of the TF-weight (Rölleke et al. (2006)). These TF-weights are aggregated for the probability aggregation in the augmented context, where the aggregation is carried out by the INDEPENDENT projection, which, as the semantics in the previous section indicate, is an implementation of the inclusion-exclusion principle based knowledge augmentation scheme discussed in 6.3.3.5.

Note that although the augmented relation is named "about", it is not supposed to be an indication of the degree of "aboutness" between an element and a context, which forms the basis for the summarisation models developed in Chapter 5. The corresponding PD expression to arrive at a probability-based augmentation is:

```
element_terms(T,E) :- elements(E) & term(T,E).
p_t_d DISJOINT(T,E) :- element_terms(T,E) | (E).
weighted_element_about INDEPENDENT(T,E) :- elements(E) & part_of_trans(X,E) & p_t_d(
    T,X).
```

6.4.3.3 Summary Definition

Following the collection of all potentially summary-forming elements in the relation *elements*, and the knowledge augmentation of these elements, we can now define the PRA expressions which will carry out the weighting of elements. We will discuss the weighting process first for the POLIS-base and POLIS-macroaveraging models, followed by the POLIS-microaveraging model.

6.4.3.3.1 POLIS-Base and -Macroaveraging Models For the POLIS-base and POLIS-macroaveraging models outline in chapter 5, two probabilities are combined in the retrieval function: the document term probability $P(t|d)$, and the collection term frequency $P(t|c)$. For the base- and the macroaveraging-model, $P(t|d)$ is given by the augmented context weights, as outlined in the previous section. The collection term frequency $P(t|c)$ for the base- and macroaveraging-models is estimated as the linear term frequency, rather than the inverse document frequency found in traditional retrieval models. To model this probability, we again use the relational "Bayes" operator, followed by a Project DISJOINT, which provides the linear occurrence-based probability of observing a term in the context

```
p_t_c  = Project  SUM[$1](Bayes(term));
```

where the semantics of the Bayes can be given as

$$P(\tau) \quad := \quad \frac{P(\tau)}{\sum(\{P(\tau')|\tau'[i_1 \cdots i_n] = \tau[i_i \cdots i_n]\})}$$

arriving at a occurrence probability per term location, which is summed per distinct term by the Project DISJOINT:

$$\begin{array}{rcl} (T,P) & = & \text{'PROJECT DISJOINT' '[' targetlist ']' '(' relation ')'} \\ P(\tau) & := & \sum_{\tau_{\text{relation}} \in T_{\text{relation}} \wedge \tau_{\text{relation}}[i_1 \cdots i_n] = \tau[i_1 \cdots i_n]} P_{\text{relation}}(\tau_{\text{relation}}) \end{array}$$

The above PRA statement can be expressed in PD as:

```
p_t_c (T) :- term(T,D) | (T).
```

For the POLIS-base model, we combine these the two probabilities using a BM25-like fractional combination. This can be achieved in PRA using the Join FRACTIONAL operation:

```
element_score = Project INDEPENDENT[$2](Join FRACTIONAL[$1=$1](
    weighted_element_about, p_t_c));
```

The semantics of the JOIN fractional are

$$\begin{aligned}
(T,P) &= \text{JOIN FRACTIONAL '[' condition ']' '(' rel}_a\text{, rel}_b\text{ ')'} \\
T &:= \{(\tau_a[i_1 \cdots i_n], \tau_b[j_1 \cdots j_n]) | \tau_a \in \text{rel}_a \wedge \tau_b \in \text{rel}_b \wedge \forall [i_1 \cdots i_n], [j_1 \cdots j_n] : i_* = j_*\} \\
P(\tau) &:= \frac{P_{\text{rel}_a}(\tau)}{P_{\text{rel}_a}(\tau) + P_{\text{rel}_b}(\tau)}
\end{aligned}$$

The JOIN using a FRACTIONAL assumption thus implements a non-linear combination of weights, similar to the combination found in the BM25 term-weighting function. Combining both $P(t|d)$ and $P(t|c)$ in this way thus implements the summarisation model developed in 5.4.1.1. The aggregation of individual term probabilities into the overall element weight $P(c|e)$ is achieved by the outer projection, using an INDEPENDENT assumption, which results in a distinct relation consisting of all potentially summary-forming elements, where all elements are weighted by the base summarisation model.

The Probabilistic Datalog statement "&" is equivalent to an ordinary Join in PRA. To model the Fractional Join, PD uses a special version of "&", the "FRAC &". The corresponding PRA statement in PD is thus:

```
element_score (E) :- weighted_element_about (T,E) FRAC & p_t_c(T).
```

For the macroaveraging model, we combine two weights: the *element_score* as defined above for the POLIS-base model, and an ad-hoc retrieval based element weight, derived from the given query terms. We will not go into details about the implementation of ad-hoc retrieval models in PRA, however, the PRA expressions for implementing Language Modelling and TF-IDF are given in Appendices A.1 and A.2 for the sake of completeness.

To arrive at a ranking of summary elements, we first need to unite the query independent element weight we calculated using the POLIS-base model with the query-based element weight provided by the ad-hoc model. We will assume that the latter is provided by a relation *ad_hoc_retrieve*. The two element weights in these relations are combined into a single relation, called *query_element_score*

```
query_element_score = Unite DISJOINT(element_score, ad_hoc_retrieve);
```

The Unite DISJOINT combines two relations in a distinct way, adding tuple probabilities. The semantics of the distinct UNITE operator can be given as

$$\begin{aligned} (T,P) &= \text{'UNITE DISJOINT' '(' relation}_a\text{, relation}_b\text{')'} \\ T &:= \{\tau|\tau \in T_{\text{relation}_a} \vee \tau \in T_{\text{relation}_b}\} \\ P(\tau) &:= P_{\text{relation}_a}(\tau) + P_{\text{relation}_b}(\tau) \end{aligned}$$

The aggregation function $\oplus$ for the macroaveraging model is thus implemented as a linear sum of both the base element weight and the ad-hoc retrieval weight. This can also be expressed in Probabilistic Datalog in the following form:

```
query_element_score DISJOINT(E) :- element_score(E) & ad_hoc_retrieve(E).
```

To get the final additive element score, we need to project on the elements:

```
final_element_score = Project ALL[$1](query_element_score);
```

which can alternatively be given in PD as:

```
final_element_score ALL(E) :- query_element_score(E).
```

This concludes the implementation discussion for the POLIS-base and the POLIS-macroaveraging models. We will now turn to the POLIS-microaveraging model.

6.4.3.3.2 POLIS-Microaveraging Model The POLIS-microaveraging model does not make use of the element score provided by the POLIS-base model to rank elements. Instead, it provides an integrated query- and element-based score.

We will start by defining the basic probabilities $P(t|e)$, $P(t|q)$, and $P(c|t)$. Note that we also define a relation *qterm_stemmed*; this is only a helper relation, which provides a stemmed representation of the original query terms:

```
p_t_e = Project DISJOINT(Bayes[$2](element_about));
```

The semantics of both the relational Bayes and the disjoint projection have been discussed in the preceding section. The result of the entire PRA expression is a relation that gives the term probabilities in the knowledge-augmented summary forming elements. Instead of using the aggregated probabilities for augmentation, the term probabilities are derived from augmented term frequencies. An equivalent PD expression can be given as:

```
p_t_e DISJOINT(T,E) :− element_about(T,E) | (E).
```

Like the element-term probability $P(t|e)$, we also create a relation for the query-term probability $P(t|q)$. Assume that the query is represented by its terms in the relation *qterm*. We create an intermediate relation *qterm_stemmed*, which contains a stemmed representation of the original query-terms. Term probabilities for the query are then derived for each query topic using the stemmed representation.

```
qterm_stemmed = Project ALL[$4,$2](Join[$1=$1](qterm, _stem));
p_t_q = Project DISJOINT(Bayes[$2](Select[$2="topicID"](qterm_stemmed)));
```

Both relations can also be defined in Probabilistic Datalog:

```
qterm_stemmed(S,topicID) :− qterm(T,topicID) & _stem(T,S).
p_t_q DISJOINT(T,topicID) :− qterm_stemmed(T,topicID) | (topicID).
```

As we pointed out in Chapter 5, there are two ways how to estimate the probability $P(c|t)$: as a *linear* probability of occurrence, and as an *informativeness* probability based on the inverse element frequency.

To model the occurrence based estimate of $P(c|t)$, we use a frequency augmented context (i.e. term knowledge of all elements is pushed up to the context level). The probability is estimated using a combination of a Bayes and a disjoint projection:

```
p_c_t = Project DISJOINT(Bayes[$1](context_about));
```

As there is only one context to be summarised at any one time, the probability of all terms in this estimate will be 1.0. This occurrence-based linear probability can similarly be expressed in PD as:

```
p_c_t DISJOINT(T) :− context_about(T) | (T).
```

To estimate $P(c|t)$ as an informativeness probability, we model it using the traditional IDF on frequency augmented elements:

```
p_c_t = Bayes MAX_IDF[] (PROJECT[$1](element_about))
```

The semantics of the IDF Bayes follow those of the normal Bayes, with an additional logarithm applied to the the tuple probabilities:

$$P(\tau_a) \quad := \quad \log(P_a(\tau)/\log(P_e(\tau))$$

Here, terms are given a higher weight if they are less frequent across all augmented elements. The corresponding PD expression follows that given for the linear estimate, however, here the assumption *max_ivf* is provided:

```
p_c_t  DISJOINT(T,C) :− context_about(T) | max_ivf T.
```

To calculate a score for elements, we first estimate the probability $P(t|e,q)$, by joining $P(t|e)$, and $P(t|q)$:

```
p_t_e_q   = Project  INDEPENDENT[$1,$2](Join[$1=$1](p_t_e, p_t_q));
```

The aggregation of the two probabilities is performed by means of the INDEPENDENT projection. Here, the $\oplus$ operation detailed in Chapter 5 is implemented via the inclusion-exclusion principle, as provided by the semantics of the project operator.

The final overall element score is provided by the probability $P(c|e,q)$, which we can derive by combining $P(t|e,q)$, and $P(c|t)$:

```
p_c_e_q  = Project  INDEPENDENT[$4](Join[$1=$1](p_c_t,p_t_e_q));
```

This combination is again performed by a Join operation followed by an INDEPENDENT projection, resulting in a distinct relation holding all the elements, with their corresponding weight.

The two relations weighting elements can similarly be expressed in PD, using the following statements:

```
p_t_e_q  INDEPENDENT(T,E) :− p_t_e(T,E) & p_t_q(T,Q).
p_c_e_q  INDEPENDENT(T) :− p_c_t(T,C) & p_t_e_q(T,E).
```

This concludes our discussion for the implementation of the POLIS-microaveraging model.

6.4.3.4 Summary Size Limitation

The final step in the summary generation is a limitation of the size of the summary. POLIS provides two ways in which the summary size can be limited: the number of elements in the summary, and the number of terms in the summary. For both limitations, we will make use of a so-called *top-K* operator, which, when applied to a probabilistic relation, considers only the top *K* tuples, when ordering a relation by tuple probabilities.

To provide an implementation for both limitations, we assume that the scored elements, provided by any of the previously discussed models, will be contained in a relation *summary_element_score*. To limit the number of *elements* used for building the summary, we can directly apply the *top-K* operator to this relation. A summary-size limitation of the form {*at_most n*} is translated into the PRA statement

```
summary = summary_element_score:n;
```

where the tuples in the resulting relation *summary* are the *n* highest scoring tuples in the input relation *summary_element_score*. This can similarly be expressed in PD, by also applying the *top-k* operator to the goal statement:

```
summary(S):n :− summary_element_score(S).
```

For implementing a size limitation based on the number of terms, we first need to join the scored elements in the relation *summary_element_score* with the *term* relation, resulting in a relation providing both element scores and the terms which occur in those elements. We then apply the *top-K* operator to this resulting relation, such that the *top-K* operates on the terms, rather than on the elements directly.

A summary-size limitation of the form *at_most n*} *terms* is thus translated into the PRA statement

```
summary = Join[$1=$2](summary_element_score,term):n;
```

and can be implemented in PD by amending the previously defined PD statement for limiting the number of elements, such that it also first joins *summary_element_score* with the relation *term*:

```
summary(T):n :− summary_element_score(S) & term(T,S).
```

6.4.4 Summary

In this section, we have provided an overview of the implementation of POLIS using the probabilistic relational algebra (PRA). Using the semantics of the PRA expressions, we were able to show that the generated PRA expressions correspond to the semantics of the respective POLIS expressions, and that concepts discussed in the context of the semantics, such as the augmentation of knowledge, can be truthfully implemented in PRA. The implementation, together with the syntax and semantics, completes the formal specification of POLIS.

6.5 Conclusion

In chapter 5 we investigated the application of well-known information retrieval models, such as TF-IDF, Language Modelling, and BM25, to the task of document summarisation. Furthermore, we developed new probabilistic summarisation models, to address some of the problems we discovered when we applied retrieval models to summarisation.

This chapter forms the second main contribution of this dissertation: the development of a *summarisation logic*. The benefit of such a logic is to provide an abstraction layer to the task of summarisation: users of the logic – such as knowledge engineers – do not need to have intimate knowledge of the summarisation models used, or the methadology of summarisation in general, to be able to apply the logic in their work.

In this chapter, we formally specified our summarisation logic – POLIS – in three steps: by providing a syntax, based on the established XPath language,
by giving the logic a well formalised meaning, using denotational "possible worlds" semantics,
and by providing details of an implementation of POLIS, using the probabilistic relational algebra (PRA).

In the next chapter, we will evaluate the summarisation models we developed in chapter 5, relying on actual POLIS expressions to show how the summarisation logic can be used in the test scenarios.

Furthermore, in chapter 8, we will thoroughly investigate how a summarisation logic can be used in the context of enterprise search, by applying POLIS to the problem of expert finding.

Chapter 7

Evaluation

7.1 Introduction

The previous chapters provided a detailed specification of POLIS; specifically, Chapter 5 discussed the derivation of summarisation models from existing ad-hoc retrieval models, and showed the development of an additional, non retrieval based, summarisation model, while Chapter 6 provides details of the syntax, semantics, and implementation of POLIS. Both chapters together provide a full specification of POLIS, however, they independently cover the primary constituents of the summarisation logic: a probabilistic model based on the underlying retrieval task, and the traditional specification of a logic, respectively.

Before we can begin to evaluate POLIS, we first need to define what constitutes an evaluation of a summarisation logic. Any abstraction logic can be evaluated on at least two levels: firstly, by evaluating how well the retrieval models implemented by the logic perform. In the case of POLIS, such an evaluation would focus on how well the summarisation models implemented by POLIS perform on summarising documents.

However, an abstraction logic can additionally be evaluated as a standalone entity, in terms of how easy it can be applied by users to its intended retrieval task. A retrieval logic like POOL would thus be evaluated in a user study, in an attempt to measure how easy it is for users to express their information need. Actual end-users of an information system are usually not available during development, however, they could be emulated by members of a more readily available student population.

Such a user-study could also be performed on POLIS, however, a further complication here is that the intended user population of POLIS consists of knowledge engineers, whose knowledge and expertise would not be present in a more general group of test persons. An emulated user population recruited from students would not be a representative sample of intended POLIS users. POLIS would thus need to be evaluated by actual knowledge engineers, ideally in the context of a real software engineering project, in which a summarisation capability is needed.

While it would have been beneficial to conduct such a user study, we had no access to the required user population. Instead, we focus in this chapter on an evaluation of the summarisation models implemented by POLIS in isolation, and provide examples of how POLIS can be applied to real-world retrieval problems in the next chapter.

In section 7.2 we introduce the evaluation methodology used for evaluating the POLIS summarisation models. This includes a discussion of the used test corpora and the evaluation framework, as well as a short introduction of the significance test applied to all results. Section 7.3 discusses the performance evaluations conducted on the test corpora. Section 7.4 concludes the evaluation of the summarisation models.

7.2 Evaluation Methodology

7.2.1 ROUGE evaluation measure

The evaluation of automatically generated summaries is a non-trivial problem, as we pointed out in Chapter 3. Of the three possible evaluation approaches (i.e. intrinsic, extrinsic, and subjective), the intrinsic evaluation method seems best suited to the task of evaluating the generated summaries in isolation, as it does not rely on a sufficiently large population of human judges to smoothen out a personal bias, which would be needed for a subjective evaluation, and also does not need to be embedded in a carefully designed retrieval experiment, which again would have required access to a user group (which here could have consisted of students, as no knowledge of POLIS would have been required).

By using the intrinsic evaluation methods, we are comparing the generated summaries to “golden standard” reference summaries. The idea is that automatically generated summaries which are very similar to the reference summaries are of higher quality.

To carry out this form of evaluation, we used the ROUGE evaluation framework developed by Lin and Hovy (Lin and Hovy (2003); Lin (2004)). ROUGE is an evaluation framework inspired by the co-occurrence measures employed for machine-translation evaluations, and calculates the overlap between generated summaries and the provided reference summaries. From this calculated overlap, it derives a "ROUGE score" indicative of the quality of the generated summaries. There are several different ways in which ROUGE can calculate the overlap between to documents, differing in certain variables such as the size of fragments compared, which can vary from individual words to multi-word snippets (called ROUGE-1, ROUGE-2, etc.), or the amount of additional words which might appear between overlapping words, which will be skipped for the purpose of calculating the overlap (called ROUGE-SUx, where "x" is the skip-distance). The ROUGE scores are given as both a "recall" and a "precision" value. The recall value is the more meaningful of the two retrieval scores, and measures the extent to which machine summaries capture the content of the reference summaries. The precision score becomes useful in evaluation situations where the automatic summaries are significantly longer than the reference summaries, such that they have an increased chance of achieving a high recall score. In all experiments here we created summaries at the same length (or close to the original length) as the reference summaries, such that the recall score is the more significant measure for the evaluations reported here. Furthermore, for the evaluation discussed here, we only report the ROUGE-1 figures, as they are the commonly used evaluation criterion in summarisation research (see Dang (2006)).

It should also be mentioned that the calculated ROUGE score is somewhat arbitrary, and should not be interpreted as a percentile.

7.2.2 Statistical Significance Testing

When we compare the automatically generated summaries with the manually written reference summaries, the ROUGE score achieved for a specific machine summary / reference summary pair indicates how well the summariser performed. To evaluate a specific summarisation strategy, and to compare its performance to another strategy, we would consider the set of all comparisons, and the average score achieved, and judge the higher scoring strategy the "better" one. Suppose a summarisation approach s_1 would achieve a higher average score than a summariser s_2, one would assume that s_1 is better than s_2.

However, we do not have any information on the validity of this assumption. Since we are only comparing average performance figures, it would be possible that the higher average score for s_1 was only achieved by a few extreme cases which were very much higher than the scores for s_2, while across the whole set of evaluations s_2 performed consistently better than s_1. For any future evaluations, s_1 might be consistently outperformed by s_2, although it was the "better" approach.

Statistical significance testing allows us to evaluate the validity of effectiveness measures, by proving the hypothesis that one summarisation strategy outperforms another summarisation strategy. To do this, significance testing employs a reverse argument: if we can refute the null hypothesis that no difference exists between two strategies, then a significant difference must exist. Suppose we want to show that s_1 performs significantly better than s_2. The null hypothesis H_0 would be that s_2 performs at least as good as s_1. To show that our original hypothesis H_1 – that s_1 is significantly better than s_2 – is correct, we need to show that the probability of H_0 is so small that it should be refuted. To test the probability of a hypothesis, we consider the difference in performance between the two strategies, expressed as the mean difference μ of performance figures across the entire set of comparisons. The two hypotheses can then be expressed as:

$$\begin{aligned} H_0 &:= \mu \leq 0 \\ H_1 &:= \mu > 0 \end{aligned}$$

The purpose of statistical significance testing is thus to calculate the probability of H_0, and to show that it can be refuted. To arrive at these probabilities, statistical inference techniques are applied to the performance figures, such as the well-known paired t-test (Student's t-test). However, many statistical inference techniques, such as the t-test, χ^2, or Kolmogorov-Smirnov, assume that the underlying data is normally distributed, an assumption which has been questioned in the context of IR evaluation (van Rijsbergen (1980)).

For this reason, significance testing of IR evaluations has made use of weaker, non-parametric significance tests, such as the Wilcoxon Signed Rank Test, or the Sign test, which do not consider actual performance figures, but instead estimate the significance of performance tests based on rankings (Wilcoxon), or binary judgements of the form "s_1 performed better than s_2 in test case t_1" (Sign test). However, while these tests overcome the problematic assumptions regarding the distribution of the performance data, Smucker *et al.* showed that significance figures reported by the Wilcoxon and Sign-test were not consistent with each other, and were also inconsistent with significance figures reported by parametric tests, such as the t-test (Smucker et al. (2007)).

Due to the problems with all of these significance tests, we will instead use the *bootstrapping* test (Efron and Tibshirani (1993)). The bootstrap is a non-parametric resampling inference test, and does not make assumptions regarding the distribution of the underlying data. Instead, it only requires the sample to be representative of the whole population. Smucker *et al.* showed that the bootstrapping test performs similar to the t-test, so that both tests agree on the statistical significance of results.

Bootstrapping simulates the underlying distribution by randomly drawing with replacement a large number of samples from the original observations. For each of these *bootstrap samples* a mean is calculated. For the set of means, the standard error is approximated, and both the mean and the standard error of the bootstrap samples allows us to compute confidence intervals. For our tests, we used confidence intervals based on percentiles of the bootstrap samples. To reject the null hypothesis H_0, we calculate 99% and 95% confidence intervals between the 1st or 5th and the 100th percentiles, respectively. A left limit of either confidence interval greater than zero allows us to reject the null hypothesis, and instead accept H_1 with 99% or 95% confidence. The two confidence intervals correspond to *p-scores* of 0.01 and 0.05 respectively. When reporting significance levels, we will report the value of the corresponding quantiles.

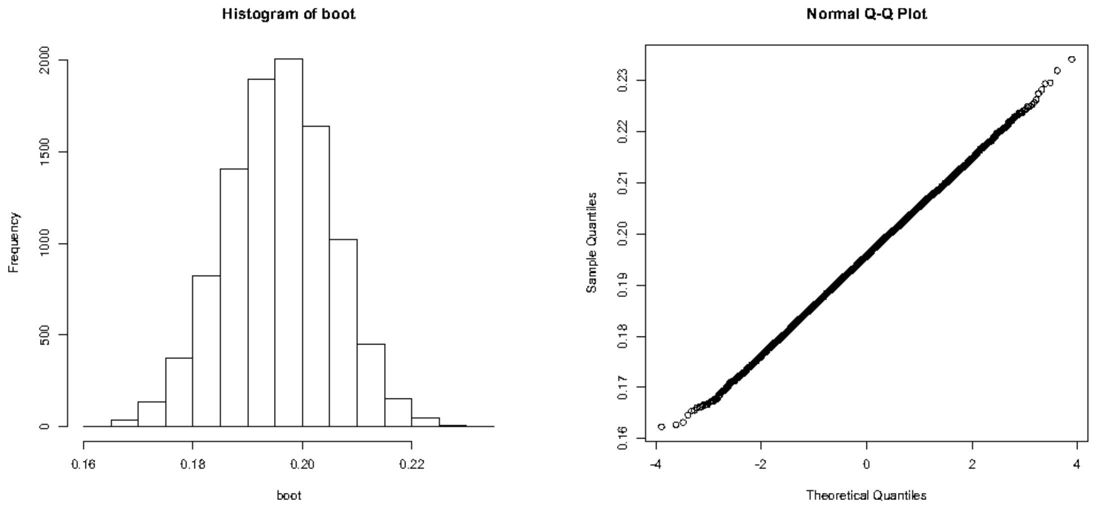


Figure 7.1: Resample histogram and quantile-quantile plot (Q-Q plot) of resampling test.

For each of the significance tests, we also provide the histogram of the resamples, and a quantile-quantile plot (Q-Q plot) of the resample quantiles versus quantiles drawn from a normal distribution. The histogram of the resamples highlights the mean of the resampled distribution, whereas the Q-Q plot indicates how far the resample distribution deviates from a normal distribution (Figure 7.1). Ideally, the resample distribution closely follows the normal distribution, so that all points in the Q-Q plot fall on the center diagonal.

To carry out the actual calculation of the bootstrap significance test we used the R environment[1], and the associated package `boot`, where we set the number of bootstrap samples to $10,000$.

7.2.3 Test Collections

The purpose of a test collection is to aid the evaluation of a specific retrieval model, or any new methodology to improve the quality of retrieval. We discussed the main components of a test collection (documents, queries, and relevance judgements) in Chapter 2, and also presented statistics for a number of test collections.

Test collections for the evaluation of summarisers differ from traditional ad-hoc retrieval test collections, as the focus lies on the generation of "good" summaries, rather than the retrieval of relevant documents. A summarisation test collection therefore consists of a collection of documents, a set of summaries, and –optionally – a set of queries which can be used to generate the summaries. Test collections can be used to specifically evaluate single document summarisation, where a reference summary is provided for every test document, or can be used to evaluate multi-document summarisation approaches, where several documents are covered by a single document.

However, there is no "ideal" test collection for evaluating the summarisation models developed in Chapter 5. As POLIS is a summarisation logic for structured documents, we would need a collection of structured documents, together with (possibly structured) reference summaries. To our knowledge, no such collection exists. Instead, we will use two test collections, which partially cover both aspects of structured summarisation: a test collection for summarisation, and a collection for evaluating structured document retrieval.

7.2.3.1 Document Understanding Conference

The first test collection we use is the Document Understanding Conference AQUAINT corpus, a collection developed for evaluating multi-document summarisation approaches.

The DUC AQUAINT collection consists of newswire texts from three sources: the Xinhua News Service, the New York Times, and the Associated Press. These documents are split into 50 subcollections (called *topics*), each of which contains 25 documents. A title and a narrative in TREC format are provided for each of the 50 topics. The title can be interpreted as a query, and can be used to guide the summarisation process for query based summarisation models. A

[1]www.r-project.org

```
<topic>
<num> D0606F </num>
<title> impacts of global climate change </title>

<narr>
What are the most significant impacts said to result from global climate change?
</narr>
</topic>
```

Figure 7.2: Sample DUC topic in TREC format.

sample topic can be found in Figure 7.2.

The task of DUC is to generate a 250 word summary for each of the 50 topics. Four human written reference summaries are provided for each topic, to which the generated summaries are compared using the ROUGE evaluation framework.

While the DUC collection is not explicitly intended to be used for evaluating structured document summarisation approaches, its documents use shallow markup, with tags denoting structure up to paragraph level. To allow a more fine-grained extraction of material, we used a sentence splitter to create markup at sentence level, allowing POLIS to extract individual sentences. The sentence splitter we used was developed by Piao at the National Center for Text Mining (NaCTeM[2]), and achieves a reported precision of 0.997352 and recall of 0.995093 on the Genia test corpus (Piao (2008)).

7.2.3.2 INEX

The second collection we used for evaluation is the INEX collection. The INEX IEEE collection was developed by the INitiative for the Evaluation of XML Retrieval[3] (INEX) to evaluate retrieval methods for structured documents. It consists of 12,107 scientific articles of the IEEE Computer Society's publications from 12 magazines and 6 transactions, covering years 1995 to 2001 / 2002. To simplify referencing subcollections in the IEEE collection, they can be referred to by either a shortened name, or a combination of shortened name and year. For example, all articles published in the *IEEE Annals of the History of Computing* are referred to by *an*, and all articles published in *an* in 1995 are referred to by the abbreviation *an/1995*. Table 7.1 shows the short and full titles of all IEEE subcollections, and the years covered.

All articles have a complex XML structure, suitable for demonstrating the application of

[2]www.nactem.ac.uk
[3]inex.is.informatik.uni-duisburg.de

Short Title	Full Title	Years
an	IEEE Annals of the History of Computing	1995 – 2001
cg	IEEE Computer Graphics and Applications	1995 – 2001
co	Computer	1995 – 2001
cs	IEEE Computational Science and Engineering	1995 – 2001
dt	IEEE Design & Test of Computers	1995 – 2001
ex	IEEE Expert	1995 – 2001
ic	IEEE Internet Computing	1997 – 2001
it	IT Professional	1999 – 2001
mi	IEEE Micro	1995 – 2001
mu	IEEE MultiMedia	1995 – 2001
pd	IEEE Parallel & Distributed Technology	1995 – 2000
so	IEEE Software	1995 – 2001
tc	IEEE Transactions on Computers	1995 – 2002
td	IEEE Transactions on Parallel and Distributed Systems	1995 – 2002
tg	IEEE Transactions on Visualization and Computer Graphics	1995 – 2002
tk	IEEE Transactions on Knowledge and Data Engineering	1995 – 2002
tp	IEEE Transactions on Pattern Analysis and Machine Intelligence	1995 – 2002
ts	IEEE Transactions on Software Engineering	1995 – 2002

Table 7.1: Short and Full Titles of the INEX IEEE subcollections.

POLIS to structured documents. In addition to the structural markup provided by the INEX collection, we also applied the sentence splitter used with the DUC collection to gain markup for individual sentences.

To use the INEX collection for summarisation experiments, it is necessary to create reference summaries for the documents. Due to the large collection size, it is not feasible to have human abstractors create such summaries. Instead, we restricted use of the collection to those documents which already contain an abstract.

However, the abstracts provided by the collection cannot easily be compared to the extracts generated by POLIS. Abstracts are a condensed representation of the sentences found in the full document, as authors of abstracts will fuse information found in several of the original sentences to arrive at one more compact sentence to be included in the abstract. As POLIS produces extract-type summaries, it will need to extract *all* those original sentences used for the fused abstract-sentence to cover the same amount of information. To evaluate extract-type summaries to abstracts, the extracts would have to be allowed to grow longer than the abstracts. However, it is not clear how much additional space should be granted to extracts for this purpose.

To overcome this problem, we created two sets of reference summaries. The first set consists of the original abstracts, which were not modified in any way. With this reference set, we created summaries at the same length as the original abstracts (e.g. if an original abstract was 150 words long, the POLIS summary was generated at that length). Figure 7.3 shows the minimum, maximum, and average length of abstracts per subcollection. For example, the smallest abstract in subcollection *mu* has a length of 35 terms, the longest abstract has a length of 93 terms, and the average length of all abstracts in *mu* is 53 terms.

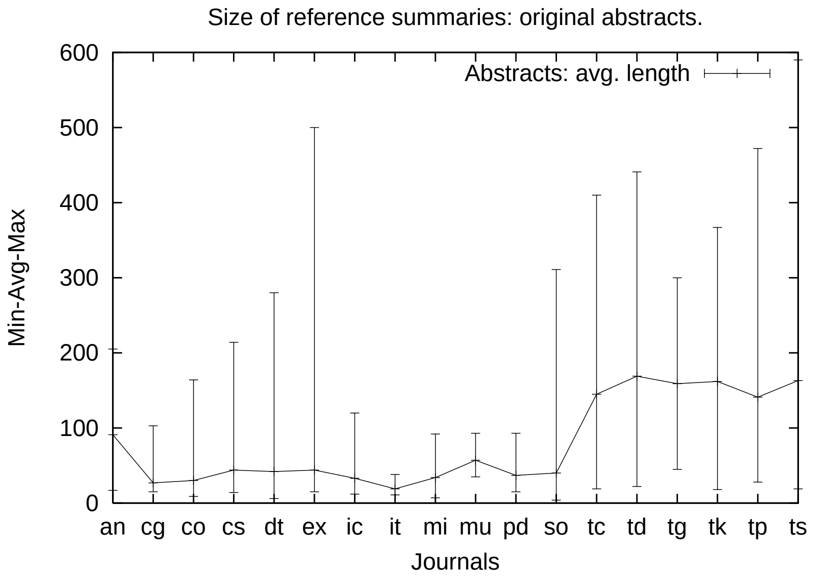


Figure 7.3: Length of original abstracts: minimum-average-maximum length.

Figure 7.4 displays the length of the abstracts as a fraction of the document length. Transactions generally have longer abstracts compared to the other subcollections, while the longest abstracts can be found in *an*, with a summary-to-document fraction of 0.022.

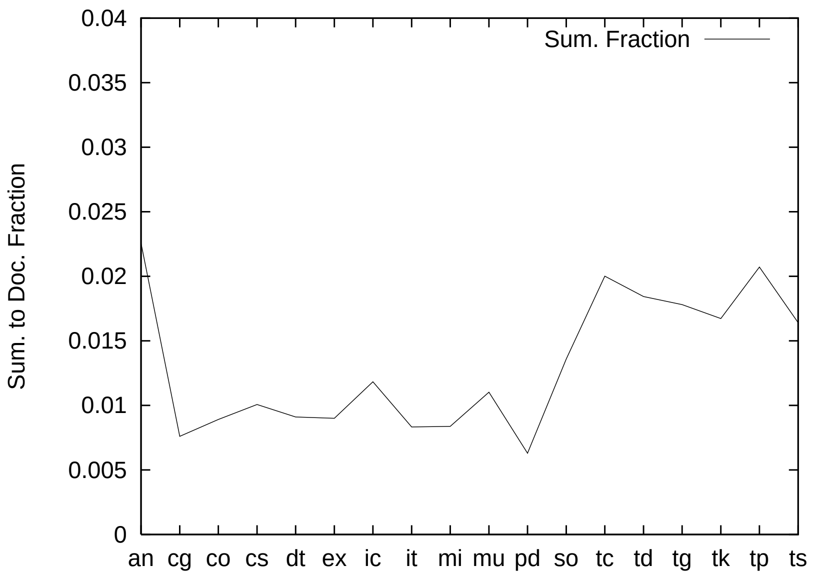


Figure 7.4: Original Abstracts: Summary to Document Fraction

As a second set of reference summaries, we created an extract based on the provided abstract using an algorithm proposed by Marcu (Marcu (1999)), where sentences are successively removed from the original document, until the remainder reaches maximal overlap with the provided abstract. The algorithm we used was implemented by Amini *et al.*, and was also used in their INEX summarisation experiments Amini et al. (2007). For this second set of reference summaries, the size of the POLIS generated summaries was restricted to the average size of extracts for a particular journal and year. For example, if the average length of extracts for a specific journal and year (e.g. *an/1995*) was 500 words, all POLIS summaries for that journal and year were created with a 500 word limit. Figure 7.5 shows the minimum, maximum, and average length of the generated (Marcu) extracts per subcollection. The generated extracts are much longer than the original abstracts. For example, the average size of extracts in subcollection *tk* is $1,045$ terms, while the longest extraxt is $8,683$ terms long.

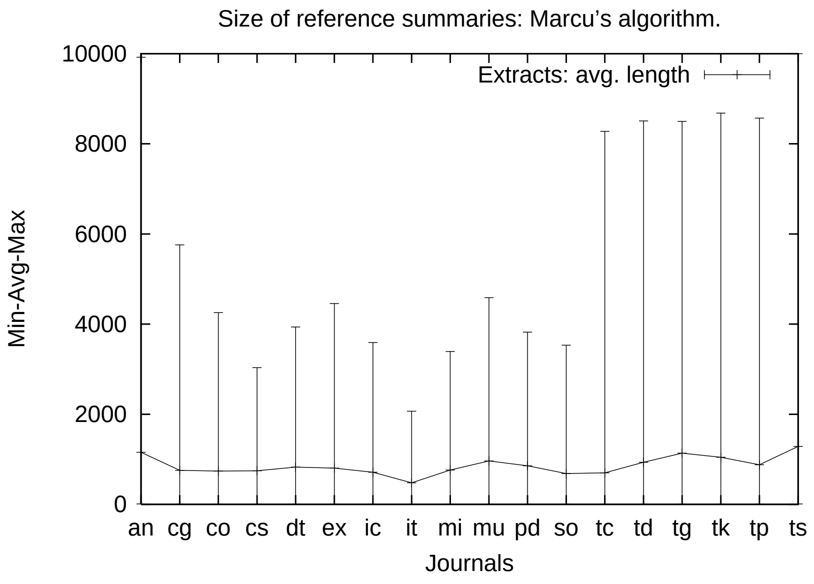


Figure 7.5: Length of Marcu's extracts: minimum-average-maximum length.

This increase in the length of reference summaries can also be observed in the summary-to-document fraction, which is between 0.09 and 0.177 for Marcu's extracts (Figure 7.6)

As POLIS summaries are generated at the same length as the reference summaries, this means that for the evaluation of summarisation models with respect to the generated extracts, the automatic summaries produced by POLIS will also be very large, resulting in a higher chance of achieving a high recall score. We therefore consider the performance evaluation with respect to the original summaries more accurate, although this implies comparing extracts to abstracts.

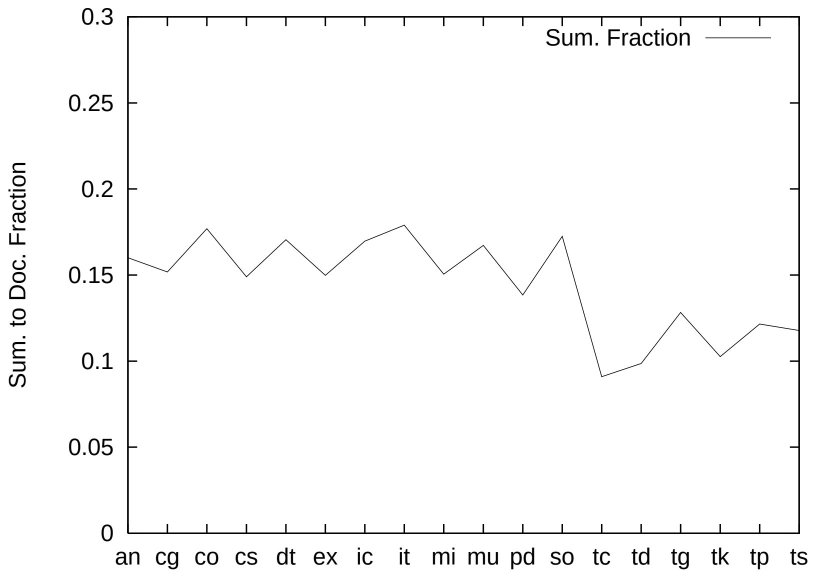


Figure 7.6: Marcu's extracts: Original Abstracts: Summary to Document Fraction

7.3 Results

We will now discuss the performance of the POLIS summarisation models on the test collections outlined previously. For both collections, we start with a short overview of the experiment carried out, followed by a discussion of the ROUGE scores achieved by the different models. For each collection, we conclude with a discussion of the performance figures reported.

7.3.1 Document Understanding Conference AQUAINT corpus

7.3.1.1 Experiment

The DUC corpus was intended to be used with query-based summarisation approaches, and most participant systems at DUC make use of the provided topic descriptions to guide the summarisation process. While the POLIS-base model is a non query-based summarisation approach, we will still compare it to both query-based POLIS models, as well as to the average performance of the DUC participant systems (denoted as "DUC avg." in the remainder of this section).

The documents in the DUC collection have limited markup, consisting of elements to denote documents ("doc"), the document body ("body"), the actual document text ("text"), and individual paragraphs within this text ("p"). We additionally introduced markup to denote sentence level ("sent"). For the non-query-based POLIS-base approach, the following expression was used to generate the summaries:

/doc/body/text/p/sent:{at_most 250 terms}

stating that the summary should be constructed from sentences ("sents") as parts of paragraphs, with up to 250 terms (words).

For the query-based models, we additionally used the topic title as a query to guide the summarisation process. For example, for topic D0601A, the title is "Native American Reservation System - pros and cons". The resulting POLIS expression for topic D0601A, using the POLIS-macroaveraging model, was:

/doc/body/text/p/sent{MACRO "native american reservation system pros and cons"}:{at_most 250 terms}

similar to the non query-based case, with the additional constraint that the summary should focus on "Native American Reservation System".

7.3.1.2 Analysis

For the DUC collection, all POLIS summaries strictly adhered to the limitations set; we therefore only show plots of the ROUGE recall. Figure 7.7 shows a plot of the per-topic performance of the POLIS-base model compared to the average performance of DUC summarisers (Note that DUC average performance figures were available for topics 1 – 35 only). The plot shows that the DUC summarisers outperform the POLIS-base model, with notable exceptions for topics 9, 11, 14, and 26. The average performance across all topics is shown in Table 7.2, and confirms the better performance of the the DUC summarisers.

Figure 7.8 shows the performance of POLIS-base in relation to the first 35 DUC AQUAINT topics, sorted ascending by DUC average performance. The plot shows that there is no clear correlation between DUC average performance and performance of the POLIS-base model.

The difference between the POLIS-base model and the DUC average is significant: the DUC summarisers significantly outperform the POLIS-base approach at 5 % significance level for recall, and at 1 % significance level for precision.

non QB	ROUGE-1-R	ROUGE-1-P
POLIS-base	0.35472 (-0.01669) †	0.35883 (-0.02701) ‡
DUC avg.	**0.37141**	**0.38584**

Table 7.2: Comparison of non query-based POLIS model to DUC average († significant at 5% significance level, ‡ significant at 1% significance level).

The results for the POLIS query-based approaches show a marked improvement over the non-query-based approach (Figure 7.9). The query-based macroaveraging model achieves a recall-

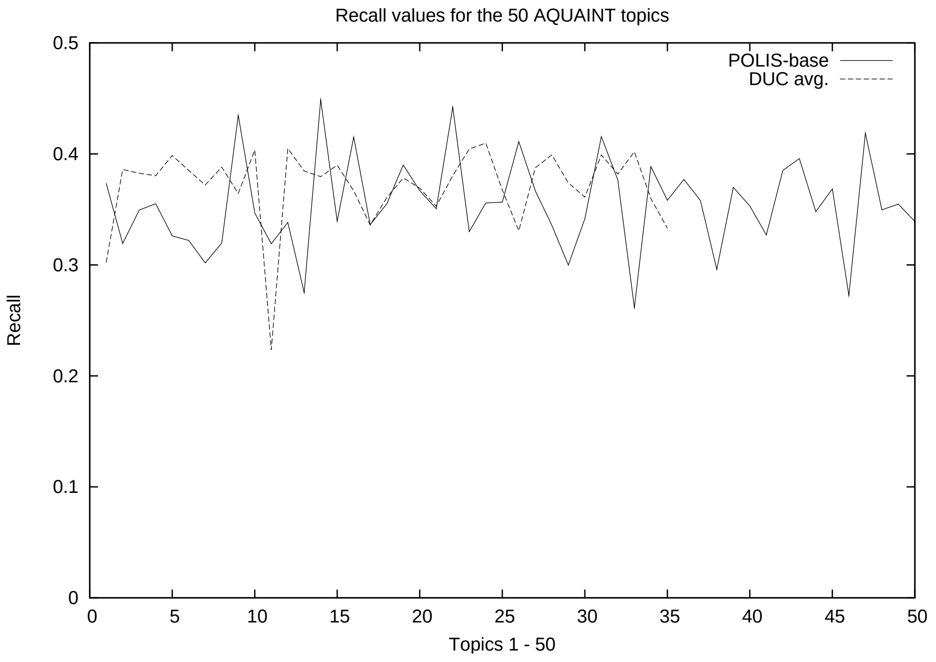


Figure 7.7: DUC collection: Comparison of recall for POLIS-base vs DUC average.

level similar to the DUC average, with no statistically significant difference between the two (the DUC average significantly outperforms POLIS-macro. at 1% significance level for precision). Similarly, there is no significant performance difference between the DUC summarisers and the POLIS-microIDF model for recall (The DUC summarisers significantly outperform the POLIS-microIDF model for precision at 1% significance level). The DUC summarisers significantly outperform the POLIS-micro. approach for both recall and precision at 1 % significance level. The POLIS-macro. approach significantly outperforms the POLIS-micro. approach for both recall and precision at 1% significance level, and significantly outperforms the POLIS-microIDF model for precision at 1% significance level. The POLIS-microIDF model significantly outperforms the POLIS-microaveraging model for recall (at 1% significance level). There is no significant difference for precision between the POLIS-microaveraging model and the POLIS-microIDF model.

There is no statistically significant difference between the POLIS query-based models and the POLIS-base model for both recall and precision. The numerical scores of the significance test for both precision and recall are shown in Tables 7.3 and 7.4.

Figure 7.10 compares the performance of the query-based models with the DUC average performance on the first 35 DUC AQUAINT topics, again sorted by ascending DUC average performance. Again, there is no clear correlation between the POLIS models and the average

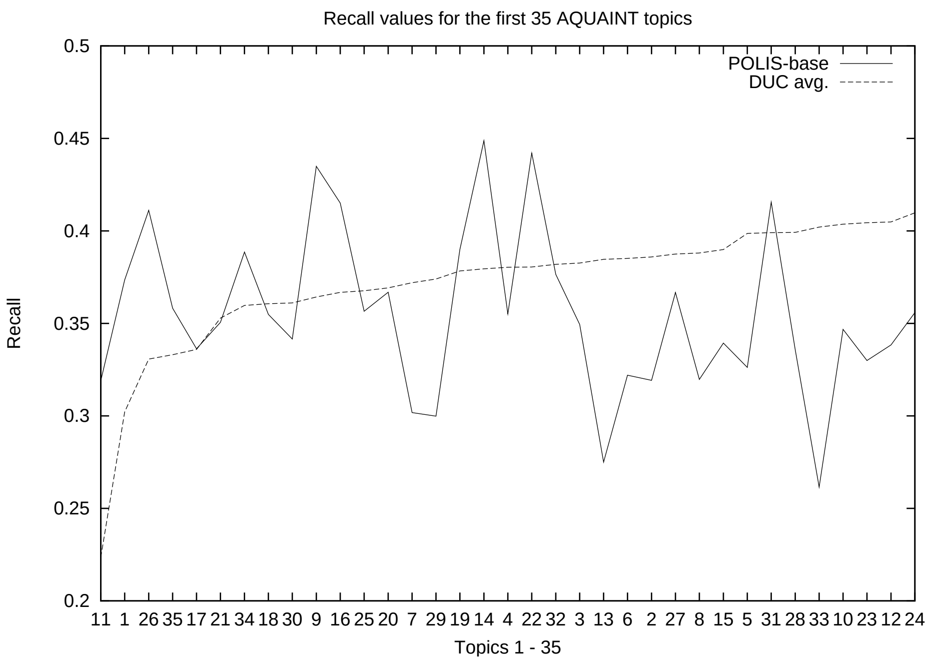


Figure 7.8: DUC collection: Comparison of POLIS-base and DUC average, sorted by DUC average.

Recall	DUC	Base	Macro	Micro	MicroIDF
DUC	–	0.000710†	◇	0.000861‡	◇
Base		–	◇	◇	◇
Macro			–	0.007733‡	◇
Micro				–	0.011284‡
MicroIDF					–

Table 7.3: DUC: Numerical significance test results: Recall.(◇ no significance, † significant at 5% significance level, ‡ significant at 1% significance level).

DUC performance.

7.3.1.3 Discussion

The non query-based POLIS-base model is significantly outperformed by the query-based DUC summarisers.

The query-based approaches show a clear improvement over the non-query-based models for the DUC collection (Table 7.5). The performance of both the macroaveraging model and the microaveraging-IDF model is only marginally below the DUC average, while the microaveraging model performs worse. Although abstract summarisation approaches like POLIS cannot be tailored as closely to the data compared to more traditional summarisation techniques, the results of the macroaveraging model are reasonably close to the other, more dedicated summarisation systems which participated in DUC.

Precision	DUC	Base	Macro	Micro	MicroIDF
DUC	–	0.007393‡	0.005699‡	0.023553‡	0.009628‡
Base		–	◇	◇	◇
Macro			–	0.007885‡	0.009271‡
Micro				–	◇
MicroIDF					–

Table 7.4: DUC: Numerical significance test results: Precision.(◇ no significance, † significant at 5% significance level, ‡ significant at 1% significance level).

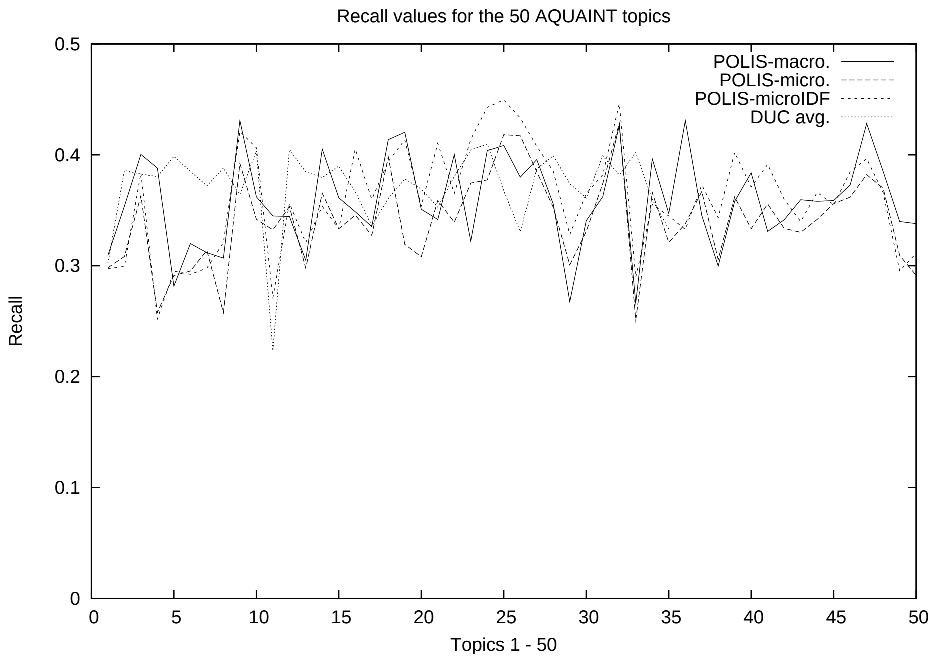


Figure 7.9: DUC collection: Comparison of recall for POLIS-macroaveraging and POLIS-microaveraging models vs DUC average.

A comparison of ROUGE scores for the DUC average to the POLIS models shows that DUC scores are more uniform across topics, while the POLIS models exhibit a higher variance of ROUGE scores across topics (exhibited in the plots in Figures 7.7 and 7.9) . We believe that the uniformity of the DUC average scores across the different topics can be attributed to a smoothing of DUC average values, due to a large number of individual values forming the average: some of the DUC participant systems will perform very well on topics for which other systems achieve a low ROUGE score, such that the average score remains constant. As different systems will excel at different topics, the overall average across all topics remains fairly constant.

Table 7.6 shows the resampling histograms and QQ-plots for the DUC significance tests. Most points in the QQ-plots fall on the center diagonal, showing that the resamples follow a normal distribution. The significance figures reported are thus valid.

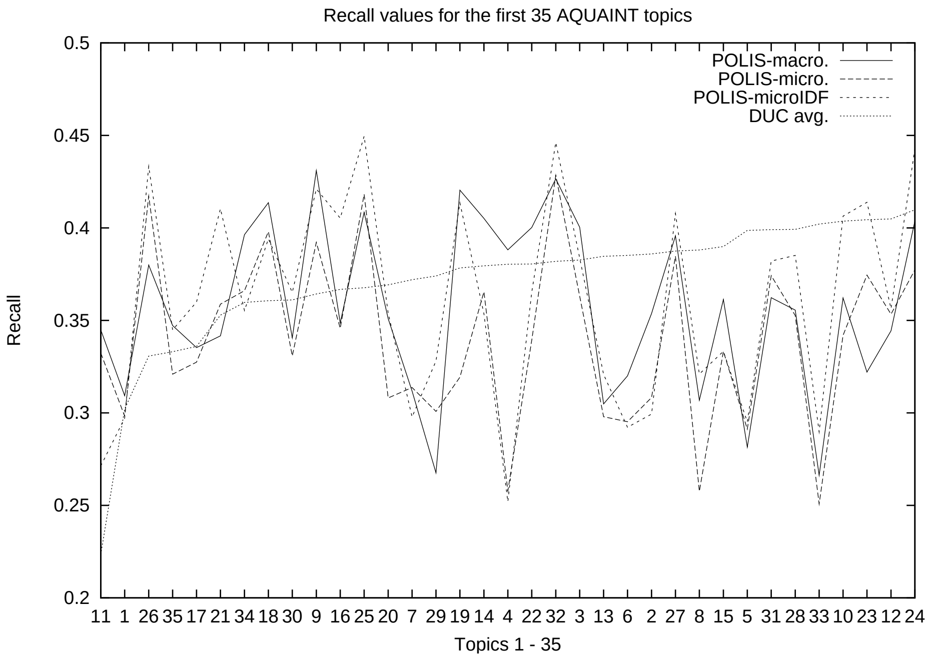


Figure 7.10: DUC collection: Comparison of recall for POLIS-macroaveraging and POLIS-microaveraging models vs DUC average, sorted by DUC average performance.

query based	ROUGE-1-R	ROUGE-1-P
POLIS-macroaveraging	0.3628 (-0.00861)	0.3669 (-0.01894)‡
POLIS-microaveraging	0.34065 (-0.03076)‡	0.34453 (-0.04131)‡
POLIS-microaveraging IDF	0.36071 (-0.0107)	0.34153 (-0.04431)‡
DUC avg.	**0.37141**	**0.38584**

Table 7.5: Comparison of query-based POLIS models to DUC average († significant at 5% significance level, ‡ significant at 1% significance level).

7.3.2 INEX IEEE collection

7.3.2.1 Experiment

The INEX collections was developed to evaluate the performance of Structured Document Retrieval (SDR) systems; the documents in the collection thus provide a much finer level of markup compared to the documents in the DUC collection. Articles in the INEX collection are denoted by an "article" element, and contain a body ("bdy"), sections ("sec"), and paragraphs ("p"). We additionally introduced markup at sentence level ("sent").

For the non query-based POLIS-base model, the following expression was used to generate the summaries:

/article/bdy/sec/p/sent:$\{$**at_most 500 terms**$\}$

stating that sentences as part pf paragraphs need to be extracted, up to a total summary size of

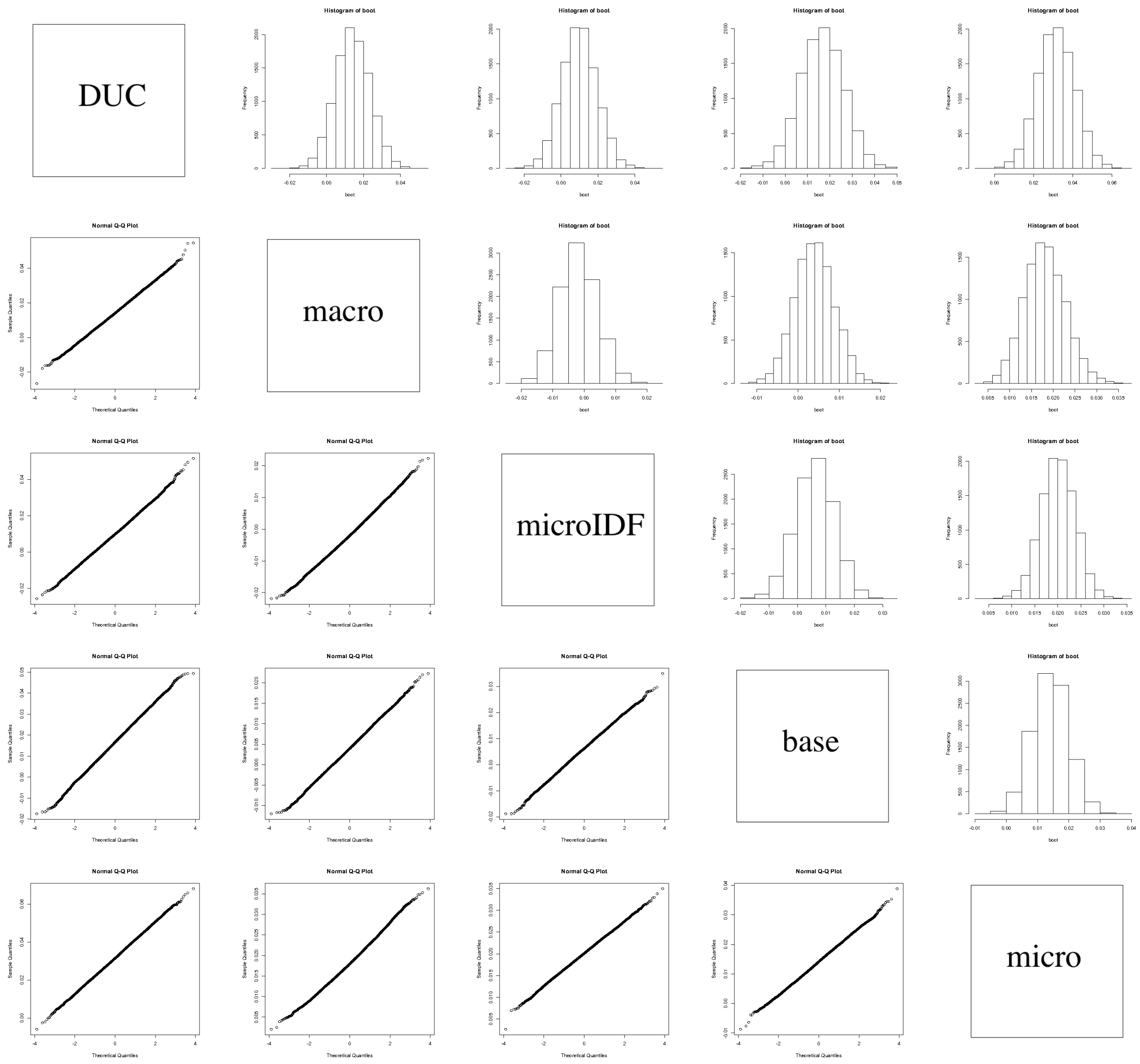


Table 7.6: DUC: QQ-plots and Histograms of bootsamples for summarisation models. Recall.

500 words.

For the query-based models, we used an article's title as the query. For example, for a document with title "Building Dynamic Agent Organizations in Cyberspace", the resulting POLIS expression (assuming use of the macroaveraging model) would be

/article/bdy/sec/p/sent{MACRO "building dynamic agent organizations in cyberspace"}:{at_most 500 terms}

similar to the non query-based expression.

We motivate the use of titles for queries as follows: an article is a written representation of an author's idea. Where an abstract provided as part of an article is a condensed representation of the remaining text, the purpose of the title is to get the attention of a reader, and provide an indication of the content. It will thus be pertinent to the content of the article. Elements in the article that contain many title-terms will thus likely also be indicative of the remaining article. We therefore believe that using titles as a guidance for the summarisation process is reasonable.

In addition to the POLIS summarisation models, we also applied a traditional feature-based summariser to the documents in the INEX collection, to be used as a baseline with which the performance of the POLIS models can be compared. The summariser uses the term frequency (key), title, and location features; we will therefore refer to it by the abbreviation KTL. As the document titles are used as summary queries, KTL produces query-based summaries. All features were given equal weights. A more exhaustive discussion of the implementation of the summariser would be beyond the scope of this dissertation, and can be found in Appendix D.

To evaluate the performance of the summariser, we applied it to the DUC collection, and compared it to the ROUGE-1 recall average of the DUC participant systems. Across topics 1 – 35, the averaged ROUGE-1 recall scores of the DUC average was 0.371, while the average of KTL was 0.379. There is no significant performance difference between the DUC average and the feature based summariser, indicating that it is reasonable to use it as a baseline for the INEX IEEE experiments.

7.3.2.2 Analysis

We start the analysis of the INEX experiments with the comparison of POLIS summaries to the generated Marcu extracts. ROUGE-1 recall and precision scores for the POLIS-base model and the two query-based models – POLIS-macroaveraging and POLIS-microaveraging – are shown in Table 7.7, together with the performance figures for the KTL summariser.

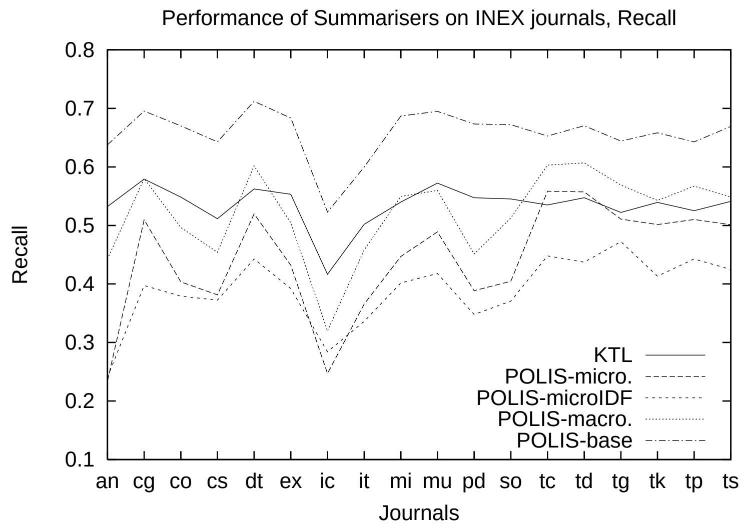


Figure 7.11: INEX collection: Comparison of recall for POLIS summarisation models; comparison with generated extracts.

The recall plot in Figure 7.11 shows that POLIS-base achieves the highest recall scores accross all topics. POLIS-macro. and KTL exhibit a similar performance, while the scores achieved by POLIS-micro. are visibly lower. This is confirmed by the performance figures for the entire collection shown in Table 7.7.

Extracts	ROUGE-1-R	ROUGE-1-P
KTL	0.53693 (-0.12246)‡	**0.46522**
POLIS-base	**0.65939**	0.31101 (-0.15421)‡
POLIS-macroaveraging	0.54246 (-0.11693)‡	0.25352 (-0.2117)‡
POLIS-microaveraging	0.47268 (-0.18671)‡	0.28272 (-0.1825)‡
POLIS-microaveraging IDF	0.38274 (-0.27665)‡	0.40696 (-0.05826)‡

Table 7.7: Comparison of QB and non-QB POLIS models for INEX collection († significant at 5% significance level, ‡ significant at 1% significance level).

The INEX collection is less commonly used for testing summarisation approaches; to facilitate an interpretation of the results we therefore also provide plots of the precision achieved per journal. The precision plot in Figure 7.12 shows that the KTL summariser achieves by far the highest precision, with a visibly lower POLIS-base in the second position. POLIS-micro exhibits a precision similar to POLIS-base, while POLIS-macro. performs worst. This is confirmed by the collection performance figures in Table 7.7

The significance tests underline the performance characteristics shown in the precision and recall plots. POLIS-base significantly outperforms POLIS-macro for recall at 1% significance level, significantly outperforms KTL at 1 % significance level, significantly outperforms POLIS-

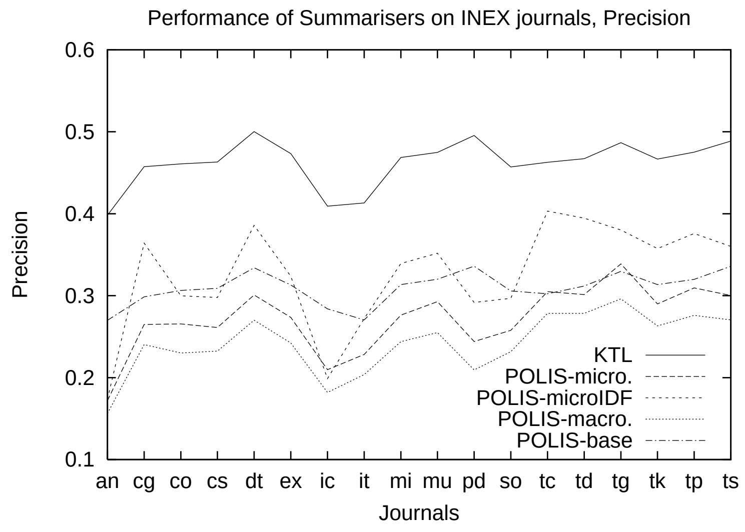


Figure 7.12: INEX collection: Comparison of precision for POLIS summarisation models; comparison with generated extracts.

micro at 1% significance level, and significantly outperforms POLIS-microIDF at 1% significance level. POLIS-macro does not significantly outperform KTL, however, it significantly outperforms POLIS-micro at 1% significance level, and POLIS-microIDF at 1% significance level. KTL significantly outperforms POLIS-micro at 1% significance level, and significantly outperforms POLIS-microIDF at 1% significance level. Finally, POLIS-micro significantly outperforms POLIS-microIDF at 1% significance level.

The significance tests for precision largely follow those for recall, with the exception of POLIS-microIDF, which receives a higher precision than recall score: KTL significantly outperforms POLIS-microIDF at 1% significance level, significantly outperforms POLIS-base at 1% significance level, significantly outperforms POLIS-micro at 1% significance level, and significantly outperforms POLIS-macro at 1% significance level. POLIS-microIDF significantly outperforms POLIS-base at 1% significance level, significantly outperform POLIS-micro at 1% significance level, and significantly outperforms POLIS-macro at 1% significance level. POLIS-base significantly outperforms POLIS-micro at 1% significance level, and significantly outperforms POLIS-macro at 1% significance level. POLIS-micro significantly outperforms POLIS-macro at 1% significance level.

The numerical values of the significance tests for both precision and recall are reported in Tables 7.8 and 7.9.

As a second experiment, we compared the POLIS- and KTL- generated summaries with the original abstracts provided by documents in the INEX collection. Table 7.7 shows the ROUGE-1

Recall	KTL	Base	Macro	Micro	MicroIDF
KTL	–	0.115956‡	◇	0.052162‡	0.271591‡
Base		–	0.1099981‡	0.174693‡	0.016183‡
Macro			–	0.061637‡	0.08721‡
Micro				–	0.040112‡
MicroIDF					–

Table 7.8: INEX, Marcu's extracts: Numerical significance test results: Recall.(◇ no significance, † significant at 5% significance level, ‡ significant at 1% significance level).

Precision	KTL	Base	Macro	Micro	MicroIDF
KTL	–	0.146417‡	0.208155‡	0.177959‡	0.180699‡
Base		–	0.05354766‡	0.023325‡	0.051371‡
Macro			–	0.026302‡	0.102755‡
Micro				–	0.085691‡
MicroIDF					–

Table 7.9: INEX, Marcu's extracts: Numerical significance test results: Precision.(◇ no significance, † significant at 5% significance level, ‡ significant at 1% significance level).

recall and precision scores for all POLIS summarisation models, and for the KTL summariser.

The recall plot in Figure 7.13 shows that KTL performs best for the first journals (an – pd), but is outperformed by the POLIS-base approach for the later journals (tc – ts). As the majority of test documents is contained in these last journals, the overall performance of POLIS-base is better than KTL (see Table 7.10). We believe that the pronounced jump in performance for the later journals is caused by the *nature* of the journals. All journals starting with *t* denote transactions, which due to their scientific nature might have more informative summaries. Informative summaries, as opposed to indicative summaries, inform on the content of an article, rather than try to arouse a reader's interest. To do so, they try to provide the most salient concepts of a text in a very condensed form, and thus resemble the type of summary produced by POLIS (and summarisation systems in general).

The recall scores of both POLIS-macro. and POLIS-micro. are below those of POLIS-base and KTL.

The precision plot in Figure 7.14 shows a similar jump in performance for the "transactions", with KTL clearly outperforming the POLIS models. POLIS-base performs better than both POLIS-macro. and POLIS-micro., however, the performance difference between POLIS-base and POLIS-micro. is very small. The overall collection performance figures in Table 7.10 confirm these results.

Significance tests for the second INEX experiments confirm the results exhibited in the preci-

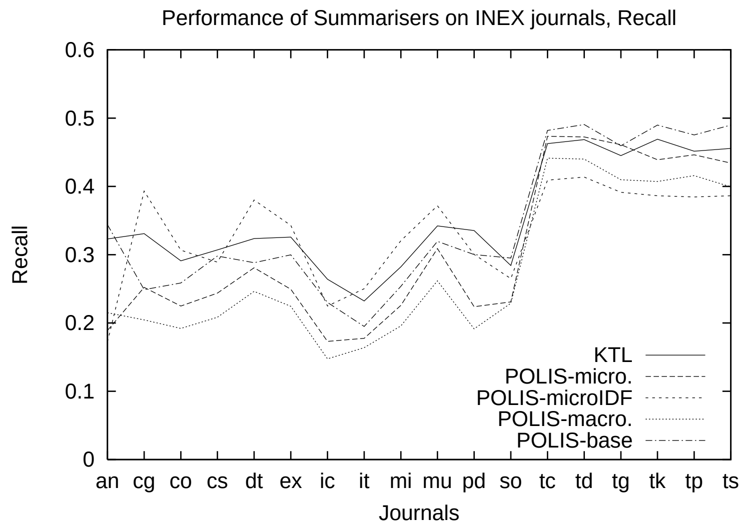


Figure 7.13: INEX collection: Comparison of recall for POLIS summarisation models; comparison with original abstracts.

orig. Abstracts	ROUGE-1-R	ROUGE-1-P
KTL	0.38636 (-0.17507)	**0.34625**
POLIS-base	0.38663 (-0.1748) ‡	0.16384 (-0.18241)‡
POLIS-macroaveraging	0.32280 (-0.23863)‡	0.13787 (-0.20838)‡
POLIS-microaveraging	0.35196 (-0.20947)‡	0.16171 (-0.18454)‡
POLIS-microaveraging IDF	**0.56143**	0.13091 (-0.21534)‡

Table 7.10: Comparison of QB and non-QB POLIS models for INEX collection († significant at 5% significance level, ‡ significant at 1% significance level).

sion and recall plots. POLIS-microIDF significantly outperforms POLIS-base at 1% significance level, however, it does not significantly outperform KTL. It significantly outperforms POLIS-micro at 1% significance level, and significantly outperforms POLIS-macro at 1% significance level. There is no significant performance difference between POLIS-base and KTL. However, POLIS-base significantly outperforms POLIS-macro at 1% significance level, and significantly outperforms POLIS-micro at 1% significance level. The KLT summariser significantly outperforms POLIS-micro at 1% significance level, and significantly outperforms POLIS-macro at 1% significance level. Finally. POLIS-micro significantly outperforms POLIS-macro at 1% significance level.

The significance tests on the precision values follow those for recall, and confirm our findings for the precision plot: the KTL summariser outperforms POLIS-base significantly at 1% significance level, significantly outperforms POLIS-micro at 1% significance level, significantly outperforms POLIS-macro at 1% significance level, and significantly outperforms POLIS-microIDF at 1% significance level. POLIS-base significantly outperforms POLIS-macro at 1% signifi-

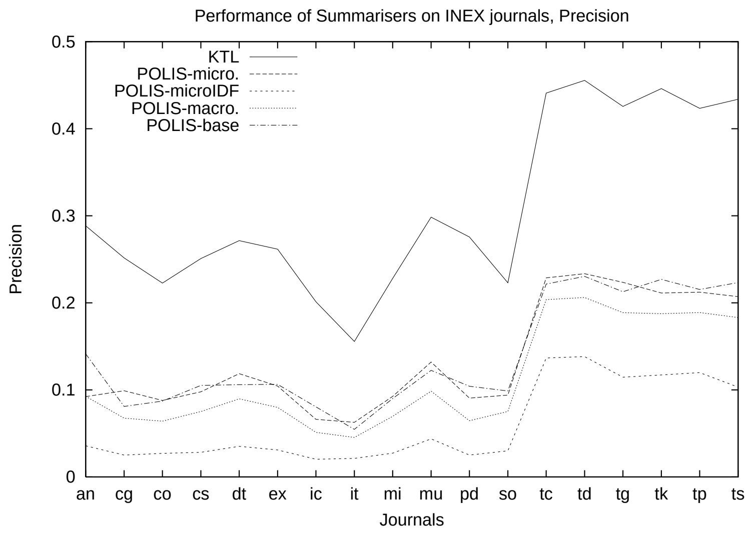


Figure 7.14: INEX collection: Comparison of precision for POLIS summarisation models; comparison with original abstracts.

cance level, significantly outperforms POLIS-microIDF at 5% significance level, but does not significantly outperform POLIS-micro. POLIS-micro significantly outperforms POLIS-macro at 1% significance level, and significantly outperforms POLIS-microIDF at 1% significance level. POLIS-macro does not significantly outperform POLIS-microIDF.

The numerical values of the significance tests for both precision and recall are reported in Tables 7.11 and 7.12.

Recall	KTL	Base	Macro	Micro	MicroIDF
KTL	–	◇	0.05924‡	0.027544‡	◇
Base		–	0.055563‡	0.022266‡	0.133497‡
Macro			–	0.017525‡	0.171151‡
Micro				–	0.144758‡
MicroIDF					–

Table 7.11: INEX, original abstracts: Significance test results: Recall.(◇ no significance, † significant at 5% significance level, ‡ significant at 1% significance level).

Precision	KTL	Base	Macro	Micro	MicroIDF
KTL	–	0.148523‡	0.172952‡	0.150515‡	0.218659‡
Base		–	0.021279‡	◇	0.002177†
Macro			–	0.018838‡	◇
Micro				–	0.00713‡
MicroIDF					–

Table 7.12: INEX, original abstracts: Significance test results: Precision.(◇ no significance, † significant at 5% significance level, ‡ significant at 1% significance level).

7.3.2.3 Discussion

The recall figures for the Marcu extracts shows that the POLIS-base model does not perform worse than a dedicated feature-based summariser, while the query-based POLIS-models perform worse. It is possible that the choice of titles as queries is counterproductive for the INEX collection, as the query-based POLIS-models performed better than POLIS-base on the DUC collection. However, the models exhibit a distinctly different behaviour in the evaluation with the original abstracts. Here, the microaveraging-IDF model performs best, and significantly outperforms the base model. Similarly, the precision scores for the microaveraging-IDF model exhibit very different behaviours for the two different evaluation approaches: while the precision score for the extended extracts is very high, the microaveraging-IDF model achieves the lowest precision score of all models on the original abstracts.

The recall plot for the original abstracts shows that all summarisation approaches perform similar, with a distinct performance increase for the transactions. In comparison, the recall plot for the Marcu extracts exhibits bigger differences between the different summarisation approaches, and no difference between the transactions and the other journals. We believe that the larger size of the Marcu-generated extracts acts as a kind of "catch-all" for sentences in the original documents, equalising any performance differences which otherwise might have been observed between the various approaches.

Finally, the Q-Q plots for the significance tests largely follow the ideal normal distribution, with notable exceptions for the comparison of POLIS-micro. and POLIS-macro., and POLIS-base and POLIS-micro., for the original abstracts. However, the divergence from the center diagonal is not big enough to invalidate the significance tests.

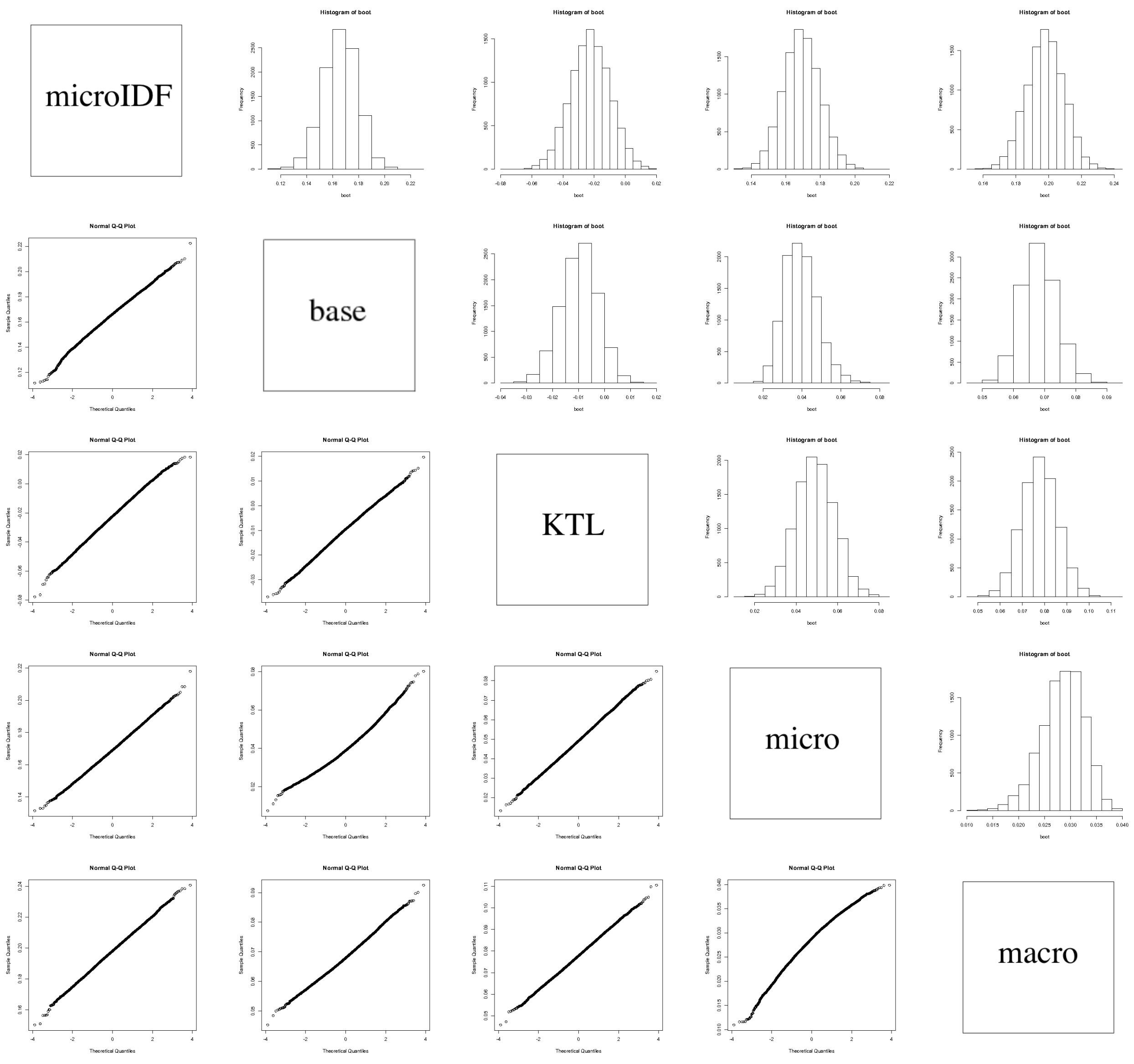


Table 7.13: INEX, original abstracts: QQ-plots and Histograms of bootsamples for summarisation models. Recall.

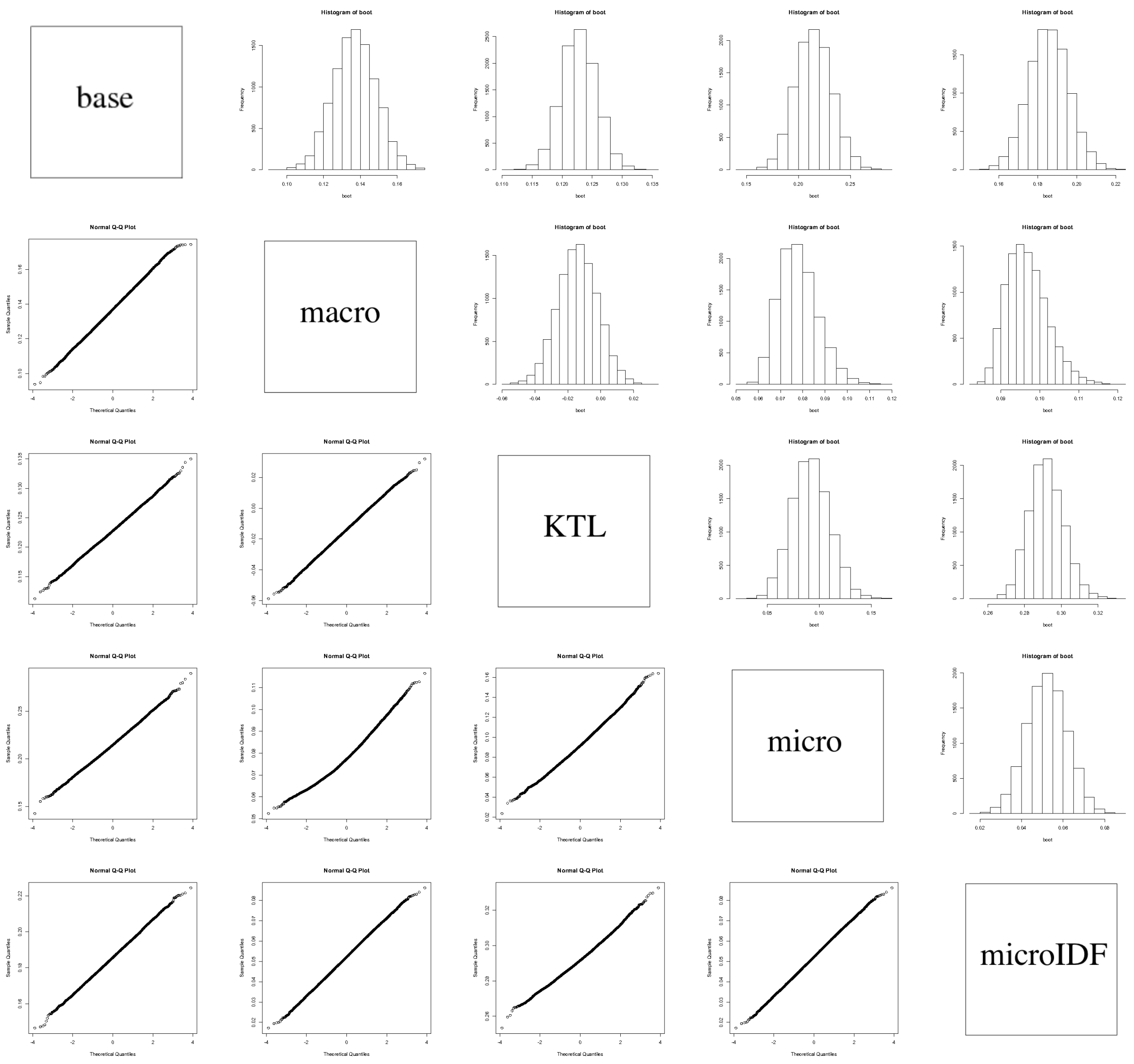


Table 7.14: INEX, Marcu's Extracts: QQ-plots and Histograms of bootsamples for summarisation models. Recall.

7.3.3 Validity Threats to Evaluation

Before concluding the evaluation of the summarisation models implemented by POLIS, we will provide a brief discussion of the threats to the validity of our experimental approaches. This discussion is inspired by similar considerations found in the context of software engineering experiments (c.f. Wohlin et al. (1999)). An evaluation of experimental validity pertains to four main types of validity: conclusion validity, internal validity, construct validity, and external validity.

7.3.3.1 Conclusion Validity

The conclusion validity is concerned with validation of observed causalities, i.e. verifying that experimental observations imply causality, and do not originate from random coincidence. We used statistical significance tests in our experiments to validate that all conclusions drawn are based on statistically significant observations.

7.3.3.2 Internal Validity

Experimental observations need to be causally connected to the experimental setting. Internal validity evaluation tries to ascertain that no factors which are not explicitly controlled in the experimental setting influence the outcome. For our evaluation we selected an established methodology for evaluating summarisation approaches. Any internal validity threats to the evaluation setting would similarly apply to other systems which are used in our comparisons. Consequently, internal threats do not invalidate our comparisons to other summarisation approaches.

7.3.3.3 Construct Validity

The experimental setting has to be chosen such that it is sufficient to evaluate the factors under consideration. For the DUC evaluation, construct validity should be guaranteed by default, as both the test collection and the evaluation strategy were explicitly constructed for the purpose of evaluating summarisation approaches.

However, there are construct validity threats for the INEX collection, as it was not conceived as a collection for evaluating summarisers. We made two assumptions for the INEX experiments which potentially invalidate the conclusions. Firstly, we assumed that the given abstracts could be considered as reference summaries. Should the provided abstracts however not pertain to the main text, a comparison of generated summaries with the given abstracts would be invalid. Secondly, we chose document titles as summarisation queries, assuming that a document title would be indicative of salient concepts in documents. However, it is possible that titles do not provide an indication for a good summary generation strategy.

7.3.3.4 External Validity

The final threat to experimental evaluations is the external validity of experimental results, i.e. whether experimental results obtained from a controlled setting can be generalised to the "real world". The main problem with both the DUC and the INEX collections is the limited set of reference summaries. It is possible that the reference summaries used in the validation of the summarisation approaches differ vastly from the "ideal" summary sought by a general user population. A summariser which would perform very well on the given test collections might thus be considered not useful by general users. However, our experiments attempt to measure the performance of the POLIS summarisation models in comparison to other summarisation approaches;

our conclusion are thus not threatened .

7.4 Conclusion

In this chapter we evaluated the POLIS summarisation models developed in chapter 5. We applied the non query-based POLIS-base model, and the three query-based models to two test collections: the DUC AQUAINT corpus, developed for evaluating multi-document summarisation systems, and the INEX IEEE collection, developed for evaluating structured document retrieval approaches.

The comparison of the POLIS models to both the DUC summarisers and a traditional feature-based summariser show that POLIS performs as good as other, dedicated summarisation approaches. Despite the higher abstraction nature that comes with the logic based approach to summarisation, there is no significant performance penalty associated with POLIS. This underlines the usefulness of POLIS, as it can be applied to real world scenarios without incurring penalties beyond possible problems stemming from the use of summaries as such.

We will elaborate on this in the next chapter, where we apply POLIS to the problem of expert-finding and expert-profiling, and show how POLIS can be used to generate document previews found in the results pages of search engines.

Chapter 8

Application Scenarios of POLIS

As we mentioned in the introduction of the last chapter, the evaluation of any abstraction logic could be concerned with either evaluating the retrieval models implemented by the logic, or evaluating the logic itself in its context of use. We mentioned that the latter was difficult for POLIS, as it would require access to a qualified user group, to evaluate the benefits of POLIS as a summarisation logic in the context of a real knowledge engineering project. The previous chapter therefore evaluated only the summarisation models implemented by POLIS in isolation. In this chapter, we try to show how POLIS can be used beneficially in real world retrieval tasks. While this is not a substitute for an actual user study, it allows us to show that POLIS can be used for solving real problems.

In this chapter, we address two situations in which summaries can provide a solution to an information retrieval problem, and show how POLIS can be applied to provide these summaries. In Section 8.1 we discuss the related problems of expert-finding and expert-profiling, and show how summaries (and specifically POLIS-generated summaries) can be used as expert-profiles. In Section 8.2.2 we show the application of POLIS to documents in the Wikipedia collection, as an example of how POLIS can be used to generate summaries to be displayed in the results page of search engines.

8.1 Expert-finding and Expert-profiling

The first part of this chapter evaluated the probabilistic summarisation models implemented by POLIS on appropriate test collections. In this second part of the POLIS evaluation, we look at how POLIS as a whole can be applied to real world information-seeking problems; specifically, we investigate the possible application of POLIS to the tasks expert-profiling and expert-finding.

In today's competitive corporate environment, employees' expertise is regarded as an important asset on par with capital or goods. However, while stock-keeping and financial record systems allow easy access to information regarding the latter two, it is much more difficult to inventory an organisation workforce's knowledge. Expert-finding and expert-profiling are two strategies proposed to tackle this problem, and can be seen as two sides of the same coin: while expert-finding seeks to answer questions of the form "Who is an expert on topic X?", expert-profiling provides information of the form "Person X is an expert on topic Y.". Both can also be combined in a hybrid-approach, where a set of expert profiles is queried in order to determine who is a possible expert on a specific topic.

We will first discuss prior research into expert-finding, and specifically the very successful multi-stage retrieval approach to expert-finding. This will be followed by a discussion of how POLIS could conceptually be used to conduct expert-profiling, where we also show that performance problems are prohibitive to conducting these experiments at the moment. We overcome these problems by combining the multi-stage retrieval approach with the summarisation-like profiling approach, and conduct an evaluation of the strategy using the Enterprise TREC W3C collection.

8.1.1 Background

A good overview over the specialities and problems encountered in conducting information retrieval in an enterprise environment is given by Hawking (Hawking (2004)). Among the challenges mentioned are the need to handle heterogeneous data sources such as databases, webservers, email repositories and content management systems, to allow for fine-grained access-control rights for different users, to support multiple document formats and to integrate all the results in a seamless fashion.

To solve the problem of expert-finding, Hawking *et al.* developed the expert finding system P@NOPTIC Expert (Craswell et al. (2001b)), which builds fairly simple expert profiles by concatenating all documents published by a person into one long document. Then, in order to retrieve a ranked list of experts for a specific topic, classical document retrieval on all those candidate documents is performed for that topic-query.

Expert finding approaches based on clustering were developed by Foner, and Reichling *et al.*.

In Foner's *Yenta* system (Foner (1997)), users, and their interest, are represented by agents. Agents representing users with similar interest form clusters in the system, which can be used to identify user groups with common interests, or to find an expert for a given problem.

Similarly, Reichling *et al.* use expert profiles to match users in their system (Reichling et al. (2004)). Their architecture is based on a learning platform; user profiles were built from information directly entered by the users (such as education, current occupation, past experience), and information derived from the browsing activities of users, i.e. documents viewed by users on the learning platform.

Yimam-Seid and Kobsa present an architecture, DEMOIR, which covers several aspects of the process of generating and querying expert profiles (Yimam and Kobsa (2000)). Their system covers the three initial stages of building and representing expert-profiles, namely evidence source recognition - using either explicit sources, such as the experts themselves, or implicit sources (e.g. documents authored by expert-candidates), expertise indicator extraction - techniques to extract the salient information from the evidence sources, and expertise models - representations of extracted salient features.

Apart from research and experimentation in academia, some commercial systems have developed expert profiles from implicit data sources. Autonomy[1] has included in its *IDOL Server* the ability to derive user profiles based on the documents that users access and submit on the company intranet. Both *KnowledgeMail* by Tacit[2] and *Discovery Server* by Lotus[3] build user profiles by scanning employees' emails.

8.1.1.1 Expert Finding as Multi-stage Retrieval

The expert finding strategies outlined in the previous section rely on the presence of some form of expert-profile to be queried in order to find the set of most likely experts. Balog and de Rijke proposed an expert finding approach, which tries to determine expertise directly from a knowledge-base of documents, without first generating expert-profiles (Balog et al. (2006b)).

Balog *et al.* model expert finding along the lines of probabilistic document retrieval approaches, such that their retrieval model tries to estimate the probability that a candidate ca is an expert given the topic query q. Instead of directly modelling this probability $P(ca|q)$, Balog *et al.* reformulate the estimate using Bayes' theorem, and thus arrive at

[1] `www.autonomy.com`
[2] `www.tacit.com`
[3] following the acquisition of Lotus by IBM, Discovery Server was discontinued.

$$P(ca|q) = \frac{P(q|ca) \cdot P(ca)}{P(q)}, \quad (8.1)$$

where $P(ca)$ is the probability of a candidate, and $P(q)$ is the probability of a query, both of which are constant for a set of queries and candidates, such that the ranking of candidates is proportional to $P(q|ca)$. Balog *et al.* propose to model this probability $P(q|ca)$ by a combination of two conditional probabilities, which are defined on the set of documents, such that

$$P(q|ca) \propto \sum_d P(q|d) \cdot P(ca|d), \quad (8.2)$$

where $P(q|d)$ is the traditional ad-hoc document retrieval probability, and $P(ca|d)$ is the probability of a candidate given a document (Balog et al. (2006a)). This candidate probability models an association between collection documents and candidates, by associating candidates and documents based on a set of features such as a candidates name, e-mail address, or any other identifying information. This set of identifying information can be regarded as a candidate query, in which case the probability $P(ca|d)$ also reduces to a traditional retrieval probability. Both $P(q|d)$ and $P(ca|d)$ can then be estimated using ad-hoc retrieval models, such as TF-IDF or Language Modelling; expert-finding thus reduces to a multi-stage retrieval tasks, without the need for prior profile generation. Instead, documents are ranked by both their relevance to a topic and their relevance to a candidate, such that a ranking of a candidate for a topic can be derived by integrating the retrieval status values of all documents pertaining to both the candidate and the topic.

8.1.2 Expert Finding via Expert Profiling

We mentioned in the opening section that in addition to the multi-stage retrieval expert-finding approach, it is also possible to model expert-finding via a hybrid approach, where first expert-profiles are derived from the document collection, and where these generated profiles are then queried in order to find experts on a given topic.

8.1.2.1 Summaries as Profiles

The expert-profiling approaches outlined in Section 8.1.1 were either based on the simple concatenation of all documents authored by an expert-candidate, or relied on information provided by the experts themselves. Here, we take the view that expert-profiles are summaries, generated from some form of knowledge-base. Assuming that the knowledge-base consists of documents only, the task of expert-profiling would reduce to a form of multi-document summarisation.

Where the multi-staged retrieval approach used a document-candidate association to determine the importance of specific documents for a candidate, and modelled this association based on a candidate's identifying features, we instead propose to use these identifying features as the query-terms for any of the query-based summarisation approaches.

For example, to generate a 5,000 word profile for Jean-Jacques Moreau (one of the expert candidates in the Enterprise TREC W3C collection, see below), using the POLIS-microIDF approach, the following POLIS-expression would be used to generate an expert profile:

/collection/doc/p/s{MICRO-IDF "Jean-Jacques Moreau
jean-jacques.moreau@crf.canon.fr"}:{at_most 5000 terms}.

This process would be repeated for all possible expert candidates, resulting in a collection of candidate profiles, which could be queried to determine expertise. However, the size of the W3C collection, and the computational complexity of the retrieval models, make a profile generation of this form prohibitively expensive. Instead, we adopted a hybrid approach, which combines some features of the multi-staged retrieval approach with the summarisation-based profiling approach outlined here.

8.1.2.2 Hybrid expert-profiling approach

One strategy to overcome the performance problems encountered in the original summaries-as-profiles approach is to reduce the number of documents which are summarised to form the profile. This can be accomplished by using the candidate-document association stage of the multi-stage expert-finding approach. When sorting all documents according to their retrieval status value with respect to a candidate's identifying features, only a top-ranking fraction of documents are considered in the summarisation stage.

However, using such a hybrid approach also influences the choice of summarisation model. In the pure summaries-as-profiles approach we used a candidate's identifying features as the query terms. Here, however, these features were already used to form the document-candidate association. Furthermore, since only the top-ranking documents of this association are considered, those candidate features will be particularly prevalent in this set of documents, exhibiting a strong bias towards the summarisation query.

A modification of the pure profiling strategy could consist of modifying the query to not any longer contain a candidate's features, but instead represent a topic's query. For example, topic 57 in the 2006 Enterprise TREC expert finding task is "OWL Lite Specification"; the corresponding POLIS expression would therefore be

/collection/doc/p/s{MICRO-IDF "OWL Lite Specification"}:{at_most 5000 terms}.

Unfortunately, this approach again suffers from serious drawbacks. Firstly, this approach would require the generation of an expert-profile for each candidate for every topic, which would be counter-intuitive to the required performance increase, the main motivation for the hybrid approach. More seriously, this strategy also exhibits a bias, in this case towards the topic queries. To find relevant experts for a topic, the set of candidate profiles is queried using the given topic query. However, this very query was already used to generate the profiles.

Consequently, the only valid summarisation approach in such a hybrid setting is to use non query-based summarisation models on the restricted set of candidate documents. The only parameters which can be modified in the experimental setting are therefore the fraction of document to be considered in the document-candidate association, and the size of the summary generate from this restricted document set.

For every candidate, the restricted set of associated documents is therefore summarised using the non query-based POLIS-base summarisation approach, using the POLIS expression

/collection/doc/p/s:{at_most 5000 terms},

where the summary size of $5,000$ terms is one of the parameters to be set in the experimental setting.

8.1.3 Evaluation

In the previous section we established that a hybrid expert-finding approach, which combines the candidate-document association of the multi-stage retrieval approach with a summaries-as-profiles strategy, has to rely on a non query-based summarisation method. In this section we will evaluate this hybrid approach on an appropriate test collection, the Enterprise TREC W3C (World Wide Web Consortium) collection.

8.1.3.1 Collection

A test-collection for the evaluation of enterprise-search strategies, such as expert-finding, needs to recreate the problems and difficulties as outlined by Hawking (Hawking (2004)). One such collection is the Enterprise TREC W3C collection.

The W3C collection consists of a crawl of the internal documentation of the World Wide Web Consortium, which was conducted in 2004. The collection consists of a heterogeneous document repository, containing emails, personal web pages, CVS repositories, etc. An overview of the collection is presented in Table 8.1 (from (Balog (2008))).

Subcollection	Description	no. of docs	size (in GB)	avg. doc. length (in terms)
lists	e-mails	198,394	1.855	376
dev	code documentation	62,509	2.578	593
www	web	45,975	1.043	1128
esw	wiki	19,605	0.181	1.5
other	misc.	3,538	0.047	335
people	personal webpages	1,016	0.003	82

Table 8.1: Statistics of W3C subcollections.

For our experiments, we only used the *lists* subcollection, which consists of emails sent through W3C mailing lists.

The documents in the collections are formatted using XHTML, where the finest level of granularity for textual elements was at paragraph level, however, this was not provided for all elements. In order to gain a more homogeneous structural layout, and to allow the extraction of material at sentence level, like for the INEX and DUC experiments, we applied a sentence-splitter to introduce markup for both paragraphs and sentences.

In addition to the documents in the W3C collection, the Enterprise TREC also provides 55 topics (queries), a set of expert-candidates (i.e. potential experts for the topics), and relevance judgements (i.e. sets of actual experts for the topics). The topics in the Enterprise TREC collection are provided in TREC formatting, similar to DUC (a sample topic can be found in Figure 8.1).

8.1.3.2 Experiments

To evaluate the hybrid expert-profiling approach, we first generated expert-profiles from the documents in the test collection. We then ran the topic queries provided with the Enterprise TREC collection on the set of expert-profiles, to retrieve the most likely expert-candidates for each topic. The output of the expert-profiling stage is a collection of expert profiles, while the output of the

```
<top>
<num> Number: EX54
<title> xml digital signature </title>
<desc> Description:
Find experts on XML digital signatures.
</desc>
<narr> Narrative:
The very features that make XML so powerful for business transactions
(e.g., semantically rich and structured data, text-based, and
Web-ready nature) provide both challenges and opportunities for the
application of encryption and digital signature operations to
XML-encoded data. XML Signature is a joint effort between the World
Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF).
In this query, we are looking for people related to research and
projects on XML digital signature within W3C.
</narr>
</top>
```

Figure 8.1: Sample Enterprise TREC topic in TREC format.

retrieval on the expert-profiles is a list of topics and the most likely experts for that topic.

We mentioned in the previous discussion that performance problems render our initial idea, to apply a query-based multi-document summarisation – with a candidate's identifying features as query terms – to the entire collection of documents to create expert-profiles, unfeasible. To overcome these performance problems, the hybrid approach proposed in section 8.1.2.2 uses the candidate-document association found in the multi-staged retrieval approach to rank the collection of document with respect to the candidate first, and then apply the actual summariser only to a fraction of the document ranking.

Following the multi-staged retrieval approach, the association between candidates and documents is modelled using a traditional ad-hoc retrieval approach, where the query terms are a candidate's identifying features. For our experiment, we implemented the candidate-document association using a Language Modelling (LM) approach, which in an earlier experiment had exhibited the best performance in a comparison of various ad-hoc retrieval models (Roelleke et al. (2009)).

As we pointed out, the summarisation stage in the hybrid expert-profiling approach cannot make use of queries, to avoid a bias towards either topics or candidates. Consequently, we do not have to consider the choice of what query to use for summarisation. However, there are two further parameters which affect the summary to be generated as an expert-profile: the fraction of documents to be summarised, and the actual summary size.

Finding a value for the first of these parameters, the fraction of documents to be summarised, is difficult. Prior research into expert-profiling uses *all* documents associated with a candidate, thus setting the fraction of documents to be used to 1.0. However, we were unable to summarise a document collection of this size, as the memory requirements of POLIS and system limitations lead to unacceptable processing times.

Instead, we applied a very limiting restriction on the actual fraction ranked documents to be used, setting the parameter to values between 0.001 and 0.008 (i.e. summarising between 0.1% and 0.8% of the ranked documents). While these settings are very restrictive, they are supported by the retrieval status values (RSVs) of documents in the ranking: a few top-ranking documents exhibit high RSVs, while the majority of documents has low RSVs. Inspecting the lower ranking documents reveals the most likely explanation for this behaviour: common query terms (such as common first- or surnames) retrieve documents which are not relevant for the candidate at hand, and thus appear at low ranks in the document ranking. The upper limit of 0.008 was again determined by system constraints, as larger document fractions caused the summarisation models to use excessive amounts of memory, and thus could not be processed.

To set the summary size, we considered the typical size of generated summaries reported in the literature on summarisation research. Common sizes for summaries reported are in the region of 10% to 20% of the original document size (e.g. Tombros and Sanderson (1998)). We chose the lower boundary of this size range, setting the summary-size to 10% of the documents to be summarised, measured as the number of terms. Given the large number of expert-candidates (1,092), smaller summaries help keep the overall amount of data manageable.

8.1.3.3 Results

Following the guidelines for the Enterprise TREC track, we retrieved the 100 most likely expert-candidates for each of the given topics, by running the respective topic queries on the set of profiles. To evaluate the quality of the retrieved expert-candidates, we compared the retrieved candidates with the provided reference experts using the official TREC-eval script.

TREC-eval takes as input a set of reference experts per topic (the so called *qrels*), and the ranked set of experts delivered by an expert-finding system, and outputs performance figures, such as –among others– mean average precision (MAP), precision at 10 (P@10), precision, and recall.

In addition to the hybrid expert-profiling approach we also applied a multi-staged retrieval approach to the Enterprise TREC collection, where both the candidate-document association and the document-topic association are implemented using a Language Modelling retrieval approach (Table 8.2).

Approach	MAP	P@10	P@5	Total Size of Profiles
Hybrid, 0.01 Document Fraction	0.0641	0.1867	0.1867	195 MB
Hybrid, 0.05 Document Fraction	0.0496	0.1739	0.1826	848 MB
Hybrid, 0.08 Document Fraction	0.0536	0.1630	0.1913	1.3 GB
LM-based Multi-staged retrieval	**0.1079**	**0.3327**	**0.2857**	N/A

Table 8.2: E-TREC: Mean Average Precision (MAP), Precision at 10 (P@10), and Precision at 5 (P@5) for the Hybrid-approach and a Language Modelling-based multi-staged retrieval approach.

The results show that the multi-stage retrieval approach clearly outperforms the hybrid approach. However, given the small size of the expert profiles, the retrieval quality for the hybrid approach is surprisingly good. An indication of this can be found in the Precision at 5 metric (P@5): despite the small size of expert profiles, the hybrid approach will on average retrieve two relevant experts in the top five ranked expert-candidates.

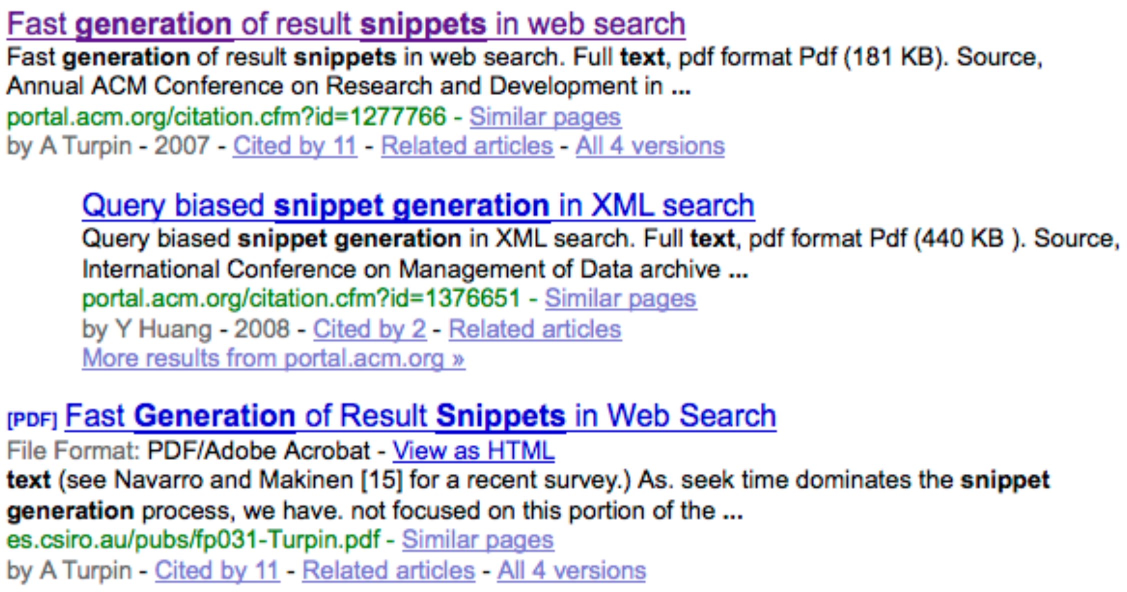
8.1.4 Expert-finding: Summary

In the first part of this chapter we used POLIS to generate expert-profiles from a collection of evidence documents. Although the purpose of this chapter is to show how POLIS can be applied in the context of real-world problems, and not to evaluate the effectiveness of the implemented summarisation models, the application of POLIS to the Enterprise TREC collection showed that the POLIS-generated expert-profiles achieve a decent performance. In the next section we will explore another possible usage scenario, where POLIS is used to summarise documents retrieved by an ad-hoc search engine.

8.2 Summaries as Text Snippets

In the first section we discussed the application of POLIS to the related problems of expert-finding and expert-profiling, and showed that summaries generated by POLIS can be used as profiles to find possible experts for given topics. In this section we will look at another application scenario for POLIS: the generation of text snippets.

Text snippets are short fragments of a longer text, e.g. individual sentences, or sub-sentence fragments, which can be presented to a user to convey an idea of the content of a longer document. As such, they are similar to indicative summaries. Text snippets are commonly found on the result pages of web search engines, where they display portions of returned pages which contain the query terms. Figure 8.2 shows how the Google search engine displays occurrences of the query terms in returned documents using text snippets, where the query "text snippet generation" was used.


Fast **generation** of result **snippets** in web search
Fast **generation** of result **snippets** in web search. Full **text**, pdf format Pdf (181 KB). Source, Annual ACM Conference on Research and Development in ...
portal.acm.org/citation.cfm?id=1277766 - Similar pages
by A Turpin - 2007 - Cited by 11 - Related articles - All 4 versions

Query biased **snippet generation** in XML search
Query biased **snippet generation** in XML search. Full **text**, pdf format Pdf (440 KB). Source, International Conference on Management of Data archive ...
portal.acm.org/citation.cfm?id=1376651 - Similar pages
by Y Huang - 2008 - Cited by 2 - Related articles
More results from portal.acm.org »

[PDF] Fast **Generation** of Result **Snippets** in Web Search
File Format: PDF/Adobe Acrobat - View as HTML
text (see Navarro and Makinen [15] for a recent survey.) As. seek time dominates the **snippet generation** process, we have. not focused on this portion of the ...
es.csiro.au/pubs/fp031-Turpin.pdf - Similar pages
by A Turpin - Cited by 11 - Related articles - All 4 versions

Figure 8.2: The Google search engine uses text snippets to highlight occurrences of query terms in returned documents.

Research into efficient text snippet generation algorithms has been influenced by other summarisation approaches, such that snippet generation can be seen as a more specialised form of summary generation. Here, we propose to use POLIS – a more general summarisation approach – to generate one- or two-sentence query-based summaries of documents, which could be displayed by search engines instead of the more specialised text snippets. However, instead of embedding POLIS in an actual search engine, we will discuss the application of POLIS on selected documents, and show how the generated summaries convey information similar to text snippets.

In the remainder of this section, we will first discuss traditional text snippet generation algorithms, and how they differ from traditional summarisation approaches. We will then show the application of POLIS to sample documents, which we extracted from the Wikipedia collection.

8.2.1 Background

Snippet generation algorithms have been derived or inspired by traditional summarisation approaches. Summarisation approaches, such as the feature-based weighting of text elements, can similarly be used in the generation of text snippets.

Turpin *et al.* observe that, assuming the minimal unit of information to be extracted is a sentence, snippet generation reduces to ranking sentences with respect to a given query (Turpin et al. (2007)). Consequently, they focus more on the *efficiency* of snippet generation algorithms, rather than on finding more *effective* methods, and analyses the benefits of various compression and caching strategies with respect to generating snippets.

Ko *et al.* propose a snippet generation algorithm based on pseudo-relevance feedback (Ko et al. (2007)). Sentences containing query terms are considered to be relevant, and all their terms become candidates for a query-expansion strategy. Snippets are generated from the top-ranking sentences with respect to this expanded query. Initial experimental results indicate that snippets generated in this way are beneficial in identifying relevant documents, however, long execution times make the approach impractical.

Huang *et al.* address the issue of snippet generation with respect to XML data, and argue that text snippets should satisfy a number of criteria in order to be useful (Huang et al. (2008)). Snippets should be self-contained, distinguishable from each other, representative of the returned document, and should be small. Their snippet generation approach tries to satisfy these goals, by first identifying and extracting the most significant information in the query results, followed by the generation of a snippet which is maximally informative with respect to the extracted information.

While snippet generation strategies are derived from summarisation approaches, there are differences between general summaries and snippets. Firstly, snippets are much shorter than summaries, usually being only a few words to a few sentences long (e.g. the snippets in Figure 8.2 are less than one sentence long). Secondly, snippets are indicative of the contents of documents, with a specific focus on the occurrences of query terms, to the point where query terms are explicitly highlighted. However, we believe that a general summarisation approach like POLIS can nonetheless be used to create snippet-like summaries, by applying query-based algorithms to the source documents, and setting the target size to one or two sentences.

8.2.2 Wikipedia

To demonstrate how POLIS-generated short summaries could be used in place of traditional snippets, we extracted two sample documents from Wikipedia. Wikipedia is a free, collaborative online encyclopaedia, which is written and maintained by volunteers. Due to the exhaustive topics covered by articles in the Wikipedia, and the prominence of the website[4], Wikipedia articles often occupy high ranks on result pages of web searches; we therefore believe that Wikipedia articles are suitable for demonstrating the application of POLIS in the context of a web search engine.

We selected two general interest Wikipedia articles to demonstrate the application of POLIS: the article on U.S. president Barack Obama (Wikipedia (2008a)), and an article about sailing (Wikipedia (2008b)). Both articles were selected randomly, and are not meant to convey a deeper agenda.

In order to generate snippet-like summaries, both documents first had to be preprocessed. Embedded data, such as images and audio-files, were removed from the documents. Additionally, textual content not related to the documents, such as banners or navigation menus, was removed, resulting in an unstructured "core" document. Like in the previous DUC and INEX experiments we applied Piao's sentence splitter (Piao (2008)), to arrive at a structural markup denoting individual sentences. We will apply the POLIS query-based summarisation models to each of the documents, and compare the summaries generated in this way to the text snippets displayed by Google.

8.2.2.1 Barack Obama

We will start our discussion with the Wikipedia article on U.S. president Barack Obama.

As a query, we considered an information need concerning a piece of biographical data, Obama's place of birth. Obama claims to have been born on Hawaii. However, during the 2008 presidential election campaign, commentators were questioning Obama's eligibility to be nominated as a presidential candidate, as it was not clear whether Obama was a natural-born citizen of the United States.

A web-searcher trying to find more information on this topic might try to find more information regarding the claim that Obama was born on Hawaii. We therefore used the query "Barack Obama Hawaii" in our demonstration. For this query, Google returns the Wikipedia article on

[4]According to an ArsTechnica interview given in December 2008, Jimmy Wales, Chair of the Wikimedia Foundation, claims Wikipedia is the 4th most popular site on the web.

Barack Obama in the top rank. Figure 8.3 shows the text snippet presented by Google for this document.

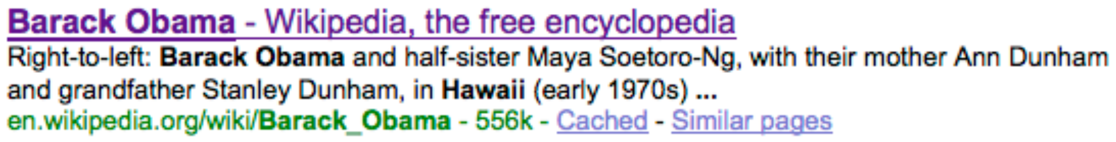

Barack Obama - Wikipedia, the free encyclopedia
Right-to-left: **Barack Obama** and half-sister Maya Soetoro-Ng, with their mother Ann Dunham and grandfather Stanley Dunham, in **Hawaii** (early 1970s) ...
en.wikipedia.org/wiki/**Barack_Obama** - 556k - Cached - Similar pages

Figure 8.3: Text snippet presented by Google for the Wikipedia article on Barack Obama.

To generate a similar summary with POLIS, we applied the POLIS-macro., POLIS-micro., and POLIS-microIDF models to the structure document generated using the sentence splitter, allowing summaries to be up to three sentences long. We used the same query as for the Google search. The resulting POLIS expression for the macroaveraging model thus was

/doc/sent{MACRO ”Barack Obama Hawaii”}:{at_most 3}

The expressions for the microaveraging and microaveraging-IDF model follow analogously. All three summarisation models produced the same three-sentence summary, shown in Figure 8.4. A direct comparison with the Google text-snippet shows that the POLIS summaries are much longer. Furthermore, the Google snippet and the POLIS summaries do not overlap in content (Comparing the Google snippet to the actual Wikipedia article shows that the snippet was generated from an image caption, which was removed in our preprocessing stage).

Barack Obama was born at the Kapi’olani Medical Center for Women and Children in Honolulu, Hawaii, to Ann Dunham, a White American from Wichita, Kansas of English and Irish descent.
Obama’s mother returned to Hawaii in 1972 for several years, and then in 1977 went back to Indonesia, where she worked as an anthropological field worker.
Obama’s father was Barack Obama, Sr., a Luo from Nyangoma Kogelo, Nyanza Province, Kenya.

Figure 8.4: POLIS-generated three-sentence summary of Wikipedia article on Barack Obama.

However, the POLIS summary is informative with respect to the stated information need, as it provides information about Barack Obama’s place of birth.

8.2.2.2 Sailing

The second Wikipedia article used for demonstrating the generation of snippet-like summaries with POLIS is about sailing. For this article, we assume that a searcher wants to find out more about the history of sailing. Consequently, the query used is “Sailing History”. For this query, Google produces a ranking which contains the Wikipedia article on sailing on third rank. The text snippet accompanying the article is shown in Figure 8.5.

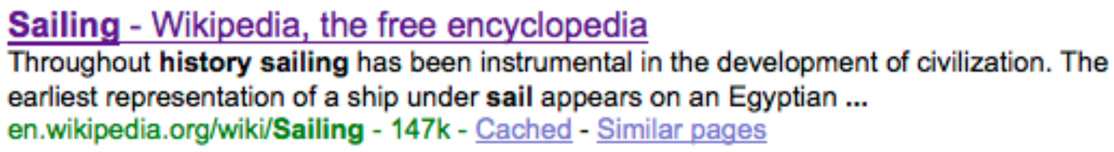

Sailing - Wikipedia, the free encyclopedia
Throughout **history sailing** has been instrumental in the development of civilization. The earliest representation of a ship under **sail** appears on an Egyptian ...
en.wikipedia.org/wiki/**Sailing** - 147k - Cached - Similar pages

Figure 8.5: Text snippet presented by Google for the Wikipedia article on Sailing.

Again, the same query was used with POLIS to generate a three-sentence summary of the article. The resulting POLIS expression for the macroaveraging model was

/doc/sent{MACRO "Sailing History"}:{at_most 3}

with analogous expressions for the two microaveraging models. While the three query-based POLIS models produced the same summary for the article on Barack Obama, the summaries for the article on sailing differed. The summary generated by the macroaveraging approach can be seen in Figure 8.6, and seems to favour longer sentences. There is no overlap with the Google snippet, and the summary focuses more on sailing in general rather than on the historical aspect of sailing (the term "history" does not at all appear in the generated summary).

Sailing is the art of controlling a sailing vessel.
By changing the rigging, rudder and dagger or centre board, a sailor manages the force of the wind on the sails in order to change the direction and speed of a boat.
Mastery of the skill requires experience in varying wind and sea conditions, as well as knowledge concerning sailboats.

Figure 8.6: POLIS-macro. three-sentence summary of Wikipedia article on Sailing.

There are several possible explanations for this behaviour. Firstly, the term "sailing" is much more prevalent in the article than the term "history": "sailing" occurs 64 times (ignoring similar terms with the same stem, such as "sail", "sailor" etc.), whereas "history" only occurs four times. Furthermore, the Wikipedia article on sailing is longer than the article on Barack Obama ($6,000$ terms vs $5,000$ terms), which might influence term probabilities in the macroaveraging model to favour the term "sailing". Finally, "sailing" occurs twice in the first summary sentence, giving it a high probability of importance with respect to other sentences.

Both microaveraging models produced the same summary, shown in Figure 8.7. Both summaries seem to favour short sentences, with the first sentence in both microaveraging models overlapping with the Google snippet, while the two remaining sentences do not focus on the history of sailing, and seem to originate from section headings.

Throughout history sailing has been instrumental in the development of civilization.
Points of sail
Sailing terminology

Figure 8.7: POLIS-micro. and POLIS-microIDF three-sentence summary of Wikipedia article on Sailing.

For this second Wikipedia article, the microaveraging models produced the more suitable summaries, better capturing and catering for the intended information need. Additionally, the first sentence in the summaries produced by the microaveraging models was also occurring in the Google snippet for the Wikipedia article.

8.2.3 Snippet Generation: Summary

The second application scenario for POLIS we looked at in this chapter was the generation of text snippets, which can be found in search engine result pages. We argued that, instead of using special generation algorithms to produce these snippets, POLIS could be used to create short summaries based on the query used to retrieve documents. To demonstrate this, we picked two articles from the Wikipedia, and demonstrated that short, query-based summaries generated by POLIS could be used as snippets on search engine results pages, as the generated summaries either overlapped with the actual snippets found in search engines, or were more informative than the original snippets.

8.3 Conclusion

We pointed out in the introduction of Chapter 7 that an abstraction logic like POLIS could be evaluated on several levels: on the one hand, it would be possible to evaluate the models implemented by POLIS in isolation, but furthermore, it would also be possible to evaluate POLIS as a whole (e.g. by evaluating its usefulness in user studies). While Chapter 7 focused on performing the isolated model evaluation, evaluating POLIS as a whole is more difficult, as it requires access to a suitable software engineering project, in which to perform user studies.

We did not have access to such an engineering project, so this chapter instead focuses on two possible application scenarios in which POLIS could be used, and demonstrates the usefulness of

POLIS. For the case of expert-finding and expert-profiling, we showed how POLIS can be used to produce summaries which act as expert-profiles. Although the summaries-as-profiles did not perform as well as a traditional expert-finding approach, the profiles were successful in finding relevant experts for given topics. Similarly, for the application scenario of text snippet generation, we demonstrated how POLIS could successfully be used to generate summaries which could be included in the results pages of search engines. This concludes our discussion of possible application scenarios for POLIS. The next chapter will summarises this thesis' contributions, and outlines possible future research directions.

Chapter 9

Contributions and Future Work

This thesis has outlined the development of a probabilistic logic for structured document summarisation called POLIS. In this chapter, we discuss the contributions made by this thesis, and outline possible directions for future research originating from the work presented here.

9.1 Contributions

Past research has established the usefulness of summaries for various retrieval tasks. The abstraction logic POLIS developed in this thesis provides a high level point of view of summarisation, which allows users with an intermediate level of knowledge of summarisation approaches (such as knowledge engineers) to easily integrate summarisation facilities into new or existing retrieval systems. The development and specification of POLIS was carried out in several stages; we will follow these stages in the discussion of the contributions made by this thesis. Section 9.1.1 details the parallelisms between summarisation and ad-hoc retrieval tasks, which motivates the summarisation models discussed in Section 9.1.2. Section 9.1.3 summarises the main contributions of the actual specification and implementation of POLIS.

9.1.1 Parallels between Summarisation and Retrieval Models

We started our discussion of POLIS by showing that summarisation can be parallelled to various ad-hoc retrieval tasks. Using the related concepts of knowledge augmentation and knowledge propagation, we showed that retrieval tasks such as structured document retrieval and anchor-text retrieval are instances of the same class of concepts as the non-retrieval task of summarisation, namely instances of the concept of knowledge augmentation.

9.1.2 Probabilistic Summarisation Models

Exploiting the parallels between summarisation and ad-hoc retrieval, we started the development of the probabilistic summarisation models provided by POLIS by showing how summarisation is another instance of the implication-probability $P(d \to q)$, which is the basis for ad-hoc retrieval models. However, we also showed that a direct application of ad-hoc retrieval models to summarisation yields problems with respect to the estimation of collection statistics.

Instead, we developed new probabilistic summarisation models, based on the conjunctive and disjunctive decomposition of the term-space of the documents to be summarised. These models follow the ad-hoc retrieval models derived from such a decomposition of the term-space, however, they use a new estimate for collection statistics, derived from the sublinear TF-component found in the BM25 retrieval model.

9.1.3 POLIS

Following the development of the probabilistic summarisation models accessible through POLIS, we provided a formal definition of the summarisation logic. Specifically, this formal definition consists of a syntax, the semantics, and a possible implementation.

9.1.3.1 Syntax

The POLIS syntax should allow developers and knowledge engineers to quickly and easily define the element-granularity at which summaries should be generated, to specify a query for query-based summarisation, and to limit the size of the generated summaries. We derived the POLIS syntax from the established XPath language, which provides access to elements in structured documents based on paths in the document trees. To accommodate the additional features required by POLIS, i.e. query-based summarisation and summary-size limitation, we modified the XPath syntax to include specifications for summarisation models and queries, and a size quantification as an upper restriction on the size of summaries.

9.1.3.2 Semantics

The second main component of the POLIS specification is the semantics. While the syntax only provides a means of interacting with the logic, and could take various possible forms, the semantics defines the “meaning” of logical expressions, and is the core of the logic.

For POLIS we used a possible worlds semantics. Contexts in structured documents are interpreted as agents, which reason about possible states of the world they are in. Using this possible worlds approach, we defined a framework which allows term-probabilities in contexts to emerge. Additionally, the possible worlds semantics provides a framework in which to formally define the concepts of evidence-propagation and knowledge-augmentation, on which the parallelism between summarisation and ad-hoc retrieval is based.

9.1.3.3 Implementation

The final element of the formal specification of POLIS provided by this thesis is one possible implementation. Syntax and semantics completely specify a language, however, we believe that a language implementation should be considered part of the formal specification, as the usability of a language relies on a stable and efficient implementation.

For a given combination of syntax and semantics, several possible implementations are possible, based on different usage scenarios and techniques. Here, we provide an implementation of POLIS based on the Probabilistic Relational Algebra (PRA). PRA, like POLIS, has well defined syntax and semantics, which allows us to demonstrate the equivalence of POLIS statements and their respective PRA implementation. Furthermore, PRA can be executed efficiently by existing retrieval frameworks, allowing for an efficient implementation of POLIS.

9.1.4 Evaluation

We contributed in this thesis two ways in which we evaluated POLIS: firstly, we evaluated the effectiveness of the summarisation models provided by POLIS in isolation; secondly, we demonstrated the usage of POLIS in two possible application scenarios.

To demonstrate the effectiveness of the POLIS summarisation models, we applied the models to two test collections: the Document Understanding Conference (DUC) AQUAINT corpus, and the INEX IEEE corpus. The evaluation based on these two corpora demonstrates that despite the difficulty in tailoring summarisation approaches to actual data which comes with higher abstraction, the performance of the POLIS summarisation models is similar to other, dedicated summarisation systems.

In addition to the purely effectiveness-based evaluation of the summarisation models, we demonstrated the applicability of POLIS in two usage scenarios. We showed how it is possible to use POLIS to generate expert-profiles for the task of expert-finding, by creating summaries of all documents associated with a particular expert-candidate. As a second usage scenario, we explained how POLIS could be included in a search engine, to generate short text snippets of retrieved documents based on the entered search queries.

9.2 Future Work

In this section we discuss how the work presented in this dissertation could be extended in the future. We will focus on two main aspects: the inclusion of additional summarisation models, and the support for heterogeneous document collections as well as the inclusion of vague predicates.

9.2.1 Models

In chapter 5, we developed probabilistic summarisation models derived from existing retrieval models, and additionally presented an entirely new summarisation model not based on existing retrieval models. However, all the summarisation models developed as part of our research differ significantly from other, more established summarisation approaches, which rely on linear combinations of weighting features, the linguistic classification of sentences, or the rhetorical analysis of documents. One possible direction for future research would be to investigate how these more traditional summarisation approaches could be modelled in a probabilistic framework.

Modelling feature-based summarisation in a probabilistic framework is a direct extension of the models developed here. The summarisation models developed in this dissertation are based on the frequency of terms in elements and in the context to be summarised, and thus bear a resemblance to the term-frequency feature found in the early work on summarisation by Luhn (Luhn (1958)). Similarly, features such as the location of elements in the context, or the presence of title terms in the elements (which could be considered an additional query) could be integrated reasonably easy.

However, integrating linguistically motivated summarisation approaches, or approaches based on the rhetoric structure of documents, into the logic would be more challenging. While it would be possible to model the weights derived from these approaches in a probabilistic framework, it would be very challenging to integrate such approaches with the document structure information provided by both structured documents and the POLIS expression.

9.2.2 Heterogeneous Collections and Vague Elements

POLIS, the summarisation logic developed in this thesis, uses XPath-like expressions to guide the summarisation process, such that elements pointed at by the path are those considered for inclusion in the summary. While such POLIS expressions are easy to formulate and understand, they rely on two premises: a homogeneous collection, in which all documents follow the same structure, and a POLIS-user who has knowledge of the structure of those documents to be summarised. Both premises could be weakened using vague element paths and vague element names.

9.2.2.1 Vague Paths

In a homogeneous collection, all documents follow a common structural layout. However, in large collections, or collections that were built over longer periods of time, documents will most likely use heterogeneous structural markup, either because new markup was introduced after documents were already included in the collection, or because documents in the collection originate from different data sources.

Such heterogeneous collections, however, require caution by users wishing to summarise documents in the collection. A user wanting to create sentence level summaries, knowing that sentences are always embedded in paragraphs, might issue a POLIS expression of the form:

/doc/p/sent{**MICRO ”former president carters international activities”**}**:**{**at_most 2000**}.

Such an expression might not match actual sentences in a heterogeneous collection, however, as additional markup might be present (e.g. “section” markup). While it would be possible to still find sentences, using a POLIS expression of the form

//sent{**MICRO ”former president carters international activities”**}**:**{**at_most 2000**},

this would require attention by the user. Ideally, the logic should automatically match path expressions in the POLIS statement to the actual structure of documents, and should return elements even if there is no exact match. This could be achieved by introducing an additional probability, such that path expressions matching completely would receive a “matching probability” of 1.0, while other elements would receive a probability based on the edit-distance necessary to match the given path expression.

9.2.2.2 Vague Elements

Similarly to the vague path expressions, it would also be possible to probabilistically model the matching of element names in the POLIS-expression to the actual element names found in the collection. Starting again with the example expression

/doc/p/sent{MICRO "former president carters international activities"}:{at_most 2000},

such an approach should model the given element name "sent" to the actual element name "sentence". Again, this could be modelled probabilistically based on the edit-distance necessary to match given element names to the actual names found in documents. An element named "sentence" in the user-specified POLIS expression would thus be matched with a higher probability on the actual element "sent" than the actual element "s".

9.2.3 Efficiency and Scalability

We briefly touched on the issue of performance and scalability in section 6.4, mentioning that the use of indexes and other database technologies would afford a highly efficient implementation of POLIS. While we did not mention any retrieval times for the experiments conducted in chapter 7, this assertion held true for the work reported there: summarising the documents in the DUC and INEX collection was almost instantaneous. However, summarising the documents collected for a candidate in the E-TREC expert-finding experiments to generate a candidate profile did take significant amounts of time. While traditional single- or multi-document summarisers would not normally be used to summarise those documents associated with a candidate in the context of expert-finding, it is conceivable that a summarisation logic like POLIS could be applied in retrieval scenarios where such documents need to be processed.

Another possible direction of future research would be concerned with finding more efficient implementations of the outlined retrieval models. It could also be possible to derive new, more efficient summarisation models from the ones developed here.

9.2.4 Ideal Summary Size

The abstraction provided by POLIS makes it easy to adapt the size of generated summaries to their application-specific context. Furthermore, it is possible to easily adjust the size should this be requested by a user, or be necessitated by changing requirements. However, these adjustments require a POLIS user to manually set the summary size in the POLIS expressions, either as the number of elements included in the summary, or as the maximal number of terms in the summary.

Ideally, the logic should be able to automatically determine the ideal size of a summary. While no one summary will ever be ideal in any given situation, there are probably ways to determine the "ideal" size given a specific upper boundary. For example, a utility function could be used to determine the benefit of including an element in a summary (measured as the increase of informativeness of the overall summary, or as the novelty introduced by that element), opposed to the cost of including the element.

9.3 Conclusion

While ad-hoc retrieval functionalities have pushed into numerous applications and electronic devices, automatic text summarisation has not yet seen the same level of exposure. Abstraction layers for the task of summarisation, such as the summarisation logic POLIS developed in this thesis, are beneficial, as they allow developers not familiar with either the mathematical models or the methodologies of summarisation approaches to include summarisation functionalities in their products.

Effective and efficient retrieval with abstraction logics is difficult, as the higher abstraction approaches cannot be tailored as closely to specific collections as traditional approaches. Despite the challenge in performance that comes with higher abstraction approaches, our evaluation results for both the DUC AQUAINT corpus and the INEX IEEE corpus have shown that the models implemented in POLIS perform at least as good as traditional "low-level" summarisation approaches. However, the abstract summarisation layer provided by POLIS additionally contributes a new, flexible approach through which summarisation methodologies can be accessed.

Bibliography

Massih R. Amini, Anastasios Tombros, Nicolas Usunier, and Mounia Lalmas. Learning-based summarisation of XML documents. *Inf. Retr.*, 10(3):233–255, 2007. ISSN 1386-4564.

Chinatsu Aone, Mary Ellen Okurowski, James Gorlinsky, and Bjornar Larsen. A scalable summarization system using robust NLP. In Inderjeet Mani and Mark Maybury, editors, *Advances in Automatic Text Summarization*, pages 71 – 80. MIT press, 1998. ISBN 0-262-13359-8.

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*, 2003. Cambridge University Press. ISBN 0-521-78176-0.

Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999. ISBN 0-201-39829-X.

Krisztian Balog. *People Search in the Enterprise*. PhD thesis, University of Amsterdam, June 2008.

Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. Formal models for expert finding in enterprise corpora. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 43–50, New York, NY, USA, 2006a. ACM Press. ISBN 1-59593-369-7.

Krisztian Balog, Edgar Meij, and Maarten de Rijke. The University of Amsterdam at the TREC 2006 Enterprise Track. In *TREC 2006 Working Notes*, pages 694–698, November 2006b.

Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. Headline generation based on statistical translation. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 318–325, Morristown, NJ, USA, 2000. Association for Computational Linguistics.

Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. In *In Pro-*

ceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97), ACL, Madrid, Spain, pages 10–17, 1997.

Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 550–557, Morristown, NJ, USA, 1999. Association for Computational Linguistics. ISBN 1-55860-609-3.

P.B. Baxendale. Machine-Made Index for Technical Literature - An Experiment. *IBM Journal of Research and Development*, 2(Nontopical Issue):354 – 361, 1958.

Adam L. Berger and Vibhu O. Mittal. Ocelot: a system for summarizing web pages. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 144–151, New York, NY, USA, 2000. ACM. ISBN 1-58113-226-3.

Ronald Brandow, Karl Mitze, and Lisa F. Lau. Automatic condensation of electronic publishing publications by sentence selection. *Inf. Process. Manage.*, 31(5):675–685, 1995.

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, 1998. ISSN 0169-7552.

Deng Cai, Shipeng Yu, Ji rong Wen, and Wei ying Ma. Extracting content structure for web pages based on visual representation. In *Proceedings of the fifth Asia Pacific Web Conference*, pages 406–417, 2003.

Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336, New York, NY, USA, 1998. ACM. ISBN 1-58113-015-5.

James Clark and Steven DeRose. XML path language (XPath) 1.0, 1999.

Cyril Cleverdon, Jack Mills, and Michael Keen. *Factors Determining the Performance of Indexing Systems: ASLIB Cranfield Research Project. Volume 1: Design.* ASLIB Cranfield Research Project, Cranfield, 1966.

Paul Clough and Mark Sanderson. Measuring pseudo relevance feedback & CLIR. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 484–485, New York, NY, USA, 2004. ACM. ISBN 1-58113-881-4.

Thomas M. Connolly and Carolyn Begg. *Database Systems: A Practical Approach to Design, Implementation, and Management*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001. ISBN 0201708574.

John M. Conroy and Dianne P. O'Leary. Text summarization via hidden markov models. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 406–407, New York, NY, USA, 2001. ACM. ISBN 1-58113-331-6.

Nick Craswell, David Hawking, and Stephen Robertson. Effective site finding using link anchor information. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 250–257, New York, NY, USA, 2001a. ACM. ISBN 1-58113-331-6.

Nick Craswell, David Hawking, Anne-Marie Vercoustre, and Peter Wilkins. Panoptic Expert: Searching for experts not just for documents. In *Ausweb Poster Proceedings*, Queensland, Australia, 2001b.

W. Bruce Croft. Knowledge-based and statistical approaches to text retrieval. *IEEE Expert: Intelligent Systems and Their Applications*, 8(2):8–12, 1993. ISSN 0885-9000.

W. Bruce Croft and John Lafferty. *Language Modeling for Information Retrieval*. Kluwer Academic Publishers, Norwell, MA, USA, 2003. ISBN 1402012160.

Hercules Dalianis and Eduard H. Hovy. Aggregation in natural language generation. In *EWNLG '93: Selected papers from the Fourth European Workshop on Trends in Natural Language Generation, An Artificial Intelligence Perspective*, pages 88–105, London, UK, 1996. Springer-Verlag. ISBN 3-540-60800-1.

Hoa Trang Dang. Overview of DUC2006. In *Proceedings of DUC2006*, pages 1 – 10, Gaithersburg, MD, USA, 2006. National Institute of Standard and Technology.

Brian D. Davison. Topical locality in the web. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 272–279, New York, NY, USA, 2000. ACM. ISBN 1-58113-226-3.

Robert L. Donaway, Kevin W. Drummey, and Laura A. Mather. A comparison of rankings produced by summarization evaluation measures. In *NAACL-ANLP 2000 Workshop on Automatic summarization*, pages 69–78, Morristown, NJ, USA, 2000. Association for Computational Linguistics.

Harold P. Edmundson. New Methods in Automatic Extracting. *Journal of the ACM*, 16(2): 264–285, 1969. ISSN 0004-5411.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1993. ISBN 978-0-412-04231-7.

Ramez A. Elmasri and Shankrant B. Navathe. *Fundamentals of Database Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999. ISBN 0-8053-1755-4.

Ronald Fagin and Joseph Y. Halpern. Reasoning About Knowledge and Probability. *Journal of the ACM*, 41(2):340–367, March 1994.

Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, Massachusetts, 1995. ISBN 0-262-06162-7.

Elisabetta Fersini, Enza Messina, and Francesco Archetti. Enhancing web page classification through image-block importance analysis. *Information Processing and Management*, 44(4): 1431–1447, 2008.

Therese Firmin and Mike J. Chrzanowski. An evaluation of text summarization systems. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*, pages 325–335. MIT press, 1998. ISBN 0-262-13359-8.

Leonard N. Foner. Yenta: a multi-agent, referral-based matchmaking system. In *AGENTS '97: Proceedings of the first international conference on Autonomous agents*, pages 301–307, New York, NY, USA, 1997. ACM Press. ISBN 0-89791-877-0.

Jan Frederik Forst. A summarisation logic for structured documents. In Wessel Kraaij, Arjen P.

de Vries, Charles L. A. Clarke, Norbert Fuhr, and Noriko Kando, editors, *SIGIR*, page 919. ACM, 2007. ISBN 978-1-59593-597-7.

Jan Frederik Forst, Anastasios Tombros, and Thomas Roelleke. Solving the enterprise trec task with probabilistic data models. In *Proceedings of TREC 2006*, 2006.

Jan Frederik Forst, Thomas Roelleke, and Anastasios Tombros. Modelling a summarisation logic in probabilistic datalog. In Alexander Hinneburg, editor, *LWA*, pages 221–228. Martin-Luther-University Halle-Wittenberg, 2007a. ISBN 978-3-86010-907-6.

Jan Frederik Forst, Anastasios Tombros, and Thomas Roelleke. POLIS: A Probabilistic Logic for Document Summarisation. In *ICTIR 2007 1st International Conference on the Theory of Information Retrieval*, pages 201 – 212. Foundation for Information Society, 2007b. ISBN 963063237-3.

Ingo Frommholz. Annotation-based document retrieval with probabilistic logics. In Norbert Fuhr, Laszlo Kovacs, and Carlo Meghini, editors, *Research and Advanced Technology for Digital Libraries. Proc. of the 11th European Conference on Digital Libraries (ECDL 2007)*, Lecture Notes in Computer Science, pages 321–332, Heidelberg et al., September 2007. Springer.

Norbert Fuhr. Probabilistic Datalog—a logic for powerful retrieval methods. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 282–290, New York, NY, USA, 1995. ACM. ISBN 0-89791-714-6.

Norbert Fuhr and Thomas Rölleke. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. *ACM Transactions on Information Systems*, 15 (1):32–66, 1997. ISSN 1046-8188.

Norbert Fuhr and Thomas Rölleke. HySpirit — A probabilistic inference engine for hypermedia retrieval in large databases. *Lecture Notes in Computer Science*, 1377:24–42, 1998.

Norbert Fuhr, Norbert Gövert, and Thomas Rölleke. Dolores: A system for logic-based retrieval of multimedia objects. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia*, pages 257–265. ACM, 1998.

Danilo Fum, Giovanni Guida, and Carlo Tasso. Evaluating importance: A step towards text summarization. In *IJCAI*, pages 840–844, 1985.

Julia Rose Galliers and Karen Sparck Jones. Evaluating Natural Language Processing Systems. In *Lecture Notes in Artificial Intelligence*. Springer, 1995.

Niyu Ge, John Hale, and Eugene Charniak. A statistical approach to anaphora resolution. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 161–170, 1998.

Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, and Jaime Carbonell. Summarizing text documents: sentence selection and evaluation metrics. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 121–128, New York, NY, USA, 1999. ACM. ISBN 1-58113-096-1.

Udo Hahn and Ulrich Reimer. Knowledge-based text summarization: salience and generalization operators for knowledge base abstraction. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*, pages 215–232. MIT Press, 1998. ISBN 0-262-13359-8.

Micheline Hancock-Beaulieu. Query expansion: advances in research in online catalogues. *J. Inf. Sci.*, 18(2):99–103, 1992. ISSN 0165-5515.

Donna Harman, R. Baeza-Yates, Edward Fox, and W. Lee. *Inverted files*, pages 28–43. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992. ISBN 0-13-463837-9.

Donna K. Harman. The first text retrieval conference (TREC-1) Rockville, MD, U.S.A., 4–6 November, 1992. *Inf. Process. Manage.*, 29(4):411–414, 1993. ISSN 0306-4573.

David Hawking. Challenges in enterprise search. In *ADC '04: Proceedings of the 15th Australasian database conference*, pages 15–24, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

David Hawking, Ellen M. Voorhees, Nick Craswell, and Peter Bailey. Overview of the TREC-8 Web Track. In *TREC*, 1999.

Philip J. Hayes and Steven P. Weinstein. CONSTRUE/TIS: A System for Content-Based Indexing of a Database of News Stories. In *IAAI '90: Proceedings of the The Second Conference* *on Innovative Applications of Artificial Intelligence*, pages 49–64. AAAI Press, 1991. ISBN 0-262-68068-8.

Djoerd Hiemstra. A probabilistic justification for using tf.idf term weighting in information retrieval. *International Journal on Digital Libraries*, 3(2):131–139, 2000.

Graeme Hirst, Chrysanne DiMarco, Eduard H. Hovy, and K. Parsons. Authoring and Generating Health-Education Documents That are Tailored to the Needs of the Individual Patient. In *Proceedings of the 6th International Conference on User Modeling*, pages 107–118, Italy, 1997.

Eduard Hovy. Automated Text Summarization. In R. Mitkov, editor, *The Oxford Handbook of Computational Linguistics*, pages 583–598, Oxford, 2005. Oxford University Press. ISBN 978-0-19-927634-9.

Eduard Hovy and Chin-Yew Lin. Automated text summarization in SUMMARIST. In Inderjeet Mani and Mark Maybury, editors, *Advances in Automatic Text Summarization*, pages 81 – 94. MIT press, 1998. ISBN 0-262-13359-8.

Eduard Hovy and Daniel Marcu. Experiments in evaluating summarization, 1998. URL `http://www.isi.edu/~marcu/coling-acl98-tutorial.html`.

Meishan Hu, Aixin Sun, and Ee-Peng Lim. Comments-oriented document summarization: understanding documents with readers' feedback. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–298, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-164-4.

Yu Huang, Ziyang Liu, and Yi Chen. Query biased snippet generation in XML search. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 315–326, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-102-6.

Hongyan Jing and Kathleen R. McKeown. The decomposition of human-written summary sentences. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 129–136, New York, NY, USA, 1999. ACM. ISBN 1-58113-096-1.

Hongyan Jing, Regina Barzilay, Kathleen McKeown, and Michael Elhadad. Summarization evaluation methods experiments and analysis. In *In AAAI Intelligent Text Summarization Workshop (Stanford, CA, Mar. 1998)*, pages 60 – 68. AAAI, 1998.

F. C. Johnson, Chris D. Paice, W. J. Black, and A. P. Neal. The application of linguistic processing to automatic abstract generation. In *Readings in information retrieval*, pages 538–552, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-55860-454-5.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11 – 21, 1972.

Min-Yen Kan and Kathleen McKeown. Information extraction and summarization: Domain independence through focus types. Technical Report Tech. Rep. CUCS-030-99, Columbia University, 1999.

Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999. ISSN 0004-5411.

Kevin Knight and Daniel Marcu. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press / The MIT Press, 2000. ISBN 0-262-51112-6.

Youngjoong Ko, Hongkuk An, and Jungyun Seo. An effective snippet generation method using the pseudo relevance feedback technique. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 711–712, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-597-7.

Julian Kupiec, Jan Pedersen, and Francine Chen. A trainable document summarizer. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 68–73, New York, NY, USA, 1995. ACM. ISBN 0-89791-714-6.

F. Wilfrid Lancaster. *Information retrieval systems: characteristics, testing, and evaluation.* Wiley, New York, 1968. ISBN 978-0471512400.

Anton Leuski, Chin-Yew Lin, and Eduard Hovy. ineats: interactive multi-document summarization. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational*

Linguistics, pages 125–128, Morristown, NJ, USA, 2003. Association for Computational Linguistics. ISBN 0-111-456789.

David D. Lewis and Karen Spärck Jones. Natural language processing for information retrieval. *Commun. ACM*, 39(1):92–101, 1996. ISSN 0001-0782.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A New Benchmark Collection for Text Categorization Research. *J. Mach. Learn. Res.*, 5:361–397, 2004. ISSN 1533-7928.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July 2004.

Chin-Yew Lin. Training a Selection Function for Extraction. In *CIKM '99: Proceedings of the eighth international conference on Information and knowledge management*, pages 55–62, New York, NY, USA, 1999. ACM. ISBN 1-58113-146-1.

Chin-Yew Lin and Eduard Hovy. From single to multi-document summarization: a prototype system and its evaluation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 457–464, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 71–78, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. Identifying topics by position. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP–97)*, pages 283 – 290, 1997.

Hans P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958. ISSN 0018-8646.

Saadia Malik, Andrew Trotman, Mounia Lalmas, and Norbert Fuhr. Overview of INEX 2006. In Norbert Fuhr, Mounia Lalmas, and Andrew Trotman, editors, *INEX*, volume 4518 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 2006. ISBN 978-3-540-73887-9.

Inderjeet Mani. Summarization evaluation: An overview. In *In NAACL 2001.*, 2001.

Inderjeet Mani and Eric Bloedorn. Summarizing similarities and differences among related documents. *Inf. Retr.*, 1(1-2):35–67, 1999. ISSN 1386-4564.

Inderjeet Mani, Barbara Gates, and Eric Bloedorn. Improving summaries by revising them. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 558–565, Morristown, NJ, USA, 1999. Association for Computational Linguistics. ISBN 1-55860-609-3.

Daniel Marcu. The automatic construction of large-scale corpora for summarization research. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 137–144, New York, NY, USA, 1999. ACM. ISBN 1-58113-096-1.

Daniel Marcu. The rhetorical parsing of natural language texts. In *Proceedings of the 35th annual meeting on Association for Computational Linguistics*, pages 96–103, Morristown, NJ, USA, 1997a. Association for Computational Linguistics.

Daniel Marcu. Improving summarization through rhetorical parsing tuning. In *Proceedings of the COLING-ACL Workshop on Very Large Corpora*, pages 206–215, 1998.

Daniel Marcu. From Discourse Structures to Text Summaries. In Inderjeet Mani and Mark T. Maybury, editors, *ACL/EACL97-WS*, pages 82–88, Madrid, Spain, July 11 1997b.

Mark T. Maybury. Generating summaries from event data. *Inf. Process. Manage.*, 31(5):735–751, 1995. ISSN 0306-4573.

Oliver A. McBryan. GENVL and WWWW: Tools for taming the web. In *In Proceedings of the First International World Wide Web Conference*, pages 79–90, 1994.

Kathleen R. McKeown and Dragomir R. Radev. Generating summaries of multiple news articles. In *Proceedings, 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74–82, New York, NY, USA, 1995. ACM.

Kathleen R. McKeown, Judith L. Klavans, Vasileios Hatzivassiloglou, Regina Barzilay, and Eleazar Eskin. Towards multidocument summarization by reformulation: progress and prospects. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference* *innovative applications of artificial intelligence*, pages 453–460, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence. ISBN 0-262-51106-1.

Carlo Meghini, Fabrizio Sebastiani, Umberto Straccia, and Costantino Thanos. A model of information retrieval based on a terminological logic. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 298–307, New York, NY, USA, 1993. ACM. ISBN 0-89791-605-0.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet:An On-Line Lexical Database. *International Journal of Lexicography*, 3(4):235 – 312, 1990.

Mandar Mitra, Amit Singhal, and Chris Buckley. Automatic text summarization by paragraph extraction. In *Proceedings of the Workshop on Intelligent Scalable Summarization at the ACL / EACL Conference*, pages 39 – 46, 1997.

Mandar Mitra, Amit Singhal, and Chris Buckley. Improving automatic query expansion. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–214, New York, NY, USA, 1998. ACM. ISBN 1-58113-015-5.

Nils J. Nilsson. Probabilistic logic. *Artif. Intell.*, 28(1):71–88, 1986. ISSN 0004-3702.

Kenji Ono, Kazuo Sumita, and Seiji Miike. Abstract generation based on rhetorical structure extraction. In *Proceedings of the 15th conference on Computational linguistics*, pages 344–348, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

Scott Piao. Personal website. World Wide Web electronic publication, 2008. URL `http://personalpages.manchester.ac.uk/staff/scott.piao/`.

Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, New York, NY, USA, 1998. ACM. ISBN 1-58113-015-5.

Martin F. Porter. An algorithm for suffix stripping. In *Readings in information retrieval*, pages 313–316, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-55860-454-5.

Yonggang Qiu and Hans-Peter Frei. Concept based query expansion. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–169, New York, NY, USA, 1993. ACM. ISBN 0-89791-605-0.

Dragomir Radev, Timothy Allison, Sasha Blair-goldensohn, John Blitzer, Arda elebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Adam Winkel, and Zhu Zhang. MEAD - a platform for multidocument multilingual text summarization. In *Proceedings of LREC 2004*, 2004.

Dragomir R. Radev. *Generating natural language summaries from multiple on-line source: language reuse and regeneration.* PhD thesis, Columbia University, 1999.

Dragomir R. Radev and Kathleen R. McKeown. Generating natural language summaries from multiple on-line sources. *Comput. Linguist.*, 24(3):470–500, 1998. ISSN 0891-2017.

Lisa F. Rau, Paul S. Jacobs, and Uri Zernik. Information extraction and text summarization using linguistic knowledge acquisition. *Inf. Process. Manage.*, 25(4):419–428, 1989. ISSN 0306-4573.

Tim Reichling, Andreas Becks, Oliver Bresser, and Volker Wulf. Kontaktanbahnung in Lernplattformen. In *Mensch & Computer 2004: Allgegenwärtige Interaktion*, pages 179–188. Oldenbourg Verlag, 2004.

Ulrich Reimer and Udo Hahn. Text condensation as knowledge base abstraction. In *Proc. IEEE-88*, pages 338–344, 1988.

Stephen Robertson. The methodology of information retrieval experiment. In K. Sparck-Jones, editor, *Information retrieval experiment*, pages 9 – 31, Newton, MA, USA, 1981. Butterworth-Heinemann.

Stephen Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 232–241, New York, NY, USA, 1994. Springer-Verlag New York, Inc. ISBN 0-387-19889-X.

Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple bm25 extension to multiple weighted fields. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49, New York, NY, USA, 2004. ACM. ISBN 1-58113-874-1.

Stephen E. Robertson. Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation*, 60:503–520, 2004.

Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at TREC. In *Text REtrieval Conference*, pages 21–30, 1992.

J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, 1971.

Thomas Roelleke. *POOL: Probabilistic Object-Oriented Logical Representation and Retrieval of Complex Objects*. Shaker Verlag, Aachen, 1999. Dissertation.

Thomas Roelleke. *Probabilistic Data Models for Information and Knowledge Management: The Integration of Database and Information Retrieval Technologies*. Probably Cambridge University Press, XXX, 2009. ISBN 0000000000000. In preparation.

Thomas Roelleke and Jun Wang. A parallel derivation of probabilistic information retrieval models. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 107–114, New York, NY, USA, 2006. ACM. ISBN 1-59593-369-7.

Thomas Roelleke, Hengzhi Wu, Jun Wang, and Hany Azzam. Modelling retrieval models in a probabilistic relational algebra with a new operator: the relational bayes. *The VLDB Journal*, 17(1):5–37, 2008. ISSN 1066-8888.

Thomas Roelleke, Jun Wang, Jan Frederik Forst, and Hengzhi Wu. Informativeness and Probability Mixtures: On the Symmetry of TF-IDF and Language Modelling. *ACM Transactions on Information Systems*, 2009. submitted.

Thomas Rölleke and Norbert Fuhr. Retrieval of complex objects using a four-valued logic. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in*

Information Retrieval, SIGIR'96, August 18-22, 1996, Zurich, Switzerland (Special Issue of the SIGIR Forum), pages 206–214. ACM, 1996. ISBN 0-89791-792-8.

Thomas Rölleke, Theodora Tsikrika, and Gabriella Kazai. A general matrix framework for modelling information retrieval. *Inf. Process. Manage.*, 42(1):4–30, 2006. ISSN 0306-4573.

Tetsuya Sakai and Karen Sparck-Jones. Generic summaries for indexing in information retrieval. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 190–198, New York, NY, USA, 2001. ACM. ISBN 1-58113-331-6.

Gerard Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988. ISSN 0306-4573.

Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986. ISBN 0070544840.

Gerard Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975. ISSN 0001-0782.

Gerard Salton, Amit Singhal, Mandar Mitra, and Chris Buckley. Automatic text structuring and summarization. *Inf. Process. Manage.*, 33(2):193–207, 1997. ISSN 0306-4573.

Roger C. Schank and Robert P. Abelson. *Scripts, Plans, Goals, and Understanding: An Enquiry into Human Knowledge Structures*. Lawrence Erlbaum Associates, 1977.

Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 21–29, New York, NY, USA, 1996. ACM. ISBN 0-89791-792-8.

Mark D. Smucker, James Allan, and Ben Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 623–632, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-803-9.

Karen Sparck-Jones. Automatic summarising: factors and directions. In Inderjeet Mani and Mark Maybury, editors, *Advances in Automatic Text Summarization*. MIT press, 1998. ISBN 0-262-13359-8.

Karen Sparck-Jones and Keith Van Rijsbergen. Information retrieval test collections. *Journal of Documentation*, 32(1):59 – 75, 1976.

Tomek Strzalkowski, Gees Stein, Jin Wang, and Bowden Wise. A robust practical text summarizer. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*, pages 26–33, 1998.

Simone Teufel and Marc Moens. Summarizing scientific articles: experiments with relevance and rhetorical status. *Comput. Linguist.*, 28(4):409–445, 2002. ISSN 0891-2017.

Simone Teufel and Marc Moens. Sentence extraction as a classification task. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*, pages 58–65. MIT Press, 1998a. ISBN 0-262-13359-8.

Simone Teufel and Marc Moens. Argumentative classification of extracted sentences as a first step towards flexible abstracting. In Inderjeet Mani and Mark Maybury, editors, *Advances in Automatic Text Summarization*, pages 155 – 176. MIT press, 1998b. ISBN 0-262-13359-8.

Anastasios Tombros and Mark Sanderson. Advantages of query biased summaries in information retrieval. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 2–10, New York, NY, USA, 1998. ACM. ISBN 1-58113-015-5.

Andrew Turpin, Yohannes Tsegay, David Hawking, and Hugh E. Williams. Fast generation of result snippets in web search. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 127–134, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-597-7.

Cornelis J. van Rijsbergen. A non-classical logic for information retrieval. *Comput. J.*, 29(6): 481–485, 1986.

Cornelis J. van Rijsbergen. *Information retrieval*. Butterworths, London, 1979.

Cornelis J. van Rijsbergen. *Information Retrieval*. Butterworths, 2nd edition, 1980.

Ellen M. Voorhees and Donna Harman. Overview of the sixth text REtrieval conference (TREC-6). *Information Processing and Management*, 36(1):3–35, 2000.

Dingding Wang, Tao Li, Shenghuo Zhu, and Chris Ding. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 307–314, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-164-4.

Furu Wei, Wenjie Li, Qin Lu, and Yanxiang He. Query-sensitive mutual reinforcement chain and its application in query-oriented multi-document summarization. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 283–290, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-164-4.

Wikipedia. Information retrieval — wikipedia, the free encyclopedia, 2009. URL `http://en.wikipedia.org/w/index.php?title=Information_retrieval&oldid=294247024`. [Online; accessed 3-June-2009].

Wikipedia. Barack Obama — Wikipedia, The Free Encyclopedia, 2008a. URL `http://en.wikipedia.org/w/index.php?title=Barack_Obama&oldid=259046549`. [Online; accessed 19-December-2008].

Wikipedia. Sailing — Wikipedia, The Free Encyclopedia, 2008b. URL `http://en.wikipedia.org/w/index.php?title=Sailing&oldid=259583377`. [Online; accessed 2-January-2009].

Michael J. Witbrock and Vibhu O. Mittal. Ultra-summarization: A statistical approach to generating highly condensed non-extractive summaries. In *SIGIR'99: Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 315–316, New York, NY, USA, 1999. ACM.

Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering: An Introduction*. Kluwer International / Springer, 1999. ISBN 978-0792386827.

Catherine G. Wolf, Sherman R. Alpert, John G. Vergo, Lev Kozakov, and Yurdaer Doganata. Summarizing technical support documents for search: expert and user studies. *IBM Syst. J.*, 43(3):564–586, 2004. ISSN 0018-8670.

Hengzhi Wu, Gabriella Kazai, and Thomas Roelleke. Modelling Anchor Text Retrieval in Book Search based on Back-of-Book Index. In *Proceedings of the SIGIR 2008 Workshop on Focused Retrieval*, pages 51–58, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-164-4.

Richard S. Wurman, David Sume, and Loring Leifer. *Information Anxiety 2*. Que, 2000. ISBN 0789724103.

Yiming Yang, Tom Pierce, and Jaime Carbonell. A study of retrospective and on-line event detection. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 28–36, New York, NY, USA, 1998. ACM. ISBN 1-58113-015-5.

Dawit Yimam and Alfred Kobsa. Demoir: A hybrid architecture for expertise modeling and recommender systems. In *WETICE '00: Proceedings of the 9th IEEE International Workshops on Enabling Technologies*, pages 67–74, Washington, DC, USA, 2000. IEEE Computer Society. ISBN 0-7695-0798-0.

Hongyuan Zha. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 113–120, New York, NY, USA, 2002. ACM. ISBN 1-58113-561-0.

Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342, New York, NY, USA, 2001. ACM. ISBN 1-58113-331-6.

Appendix A

Implementation of Ad-hoc retrieval models in PRA

A.1 Language Modelling (LM) Implementation in PRA

Language Modelling in PRA:

```
# docModel(Term, DocId):
docModel =
  Project  DISJOINT(
    Bayes[$2](term)) ;

# collModel(Term):
collModel  =
  Project  DISJOINT[$1](
    Bayes[](term)) ;

0.8  mixture(docModel);
0.2  mixture(collModel);

lambda_doc =
  Project [](
    Select [$1=docModel](mixture));
```

```
lambda_coll =
 Project [](
   Select [$1=collModel](mixture)) ;
# lambda_coll = Project COMPLEMENT(lambda_doc);


# retrieved (DocId):
 retrieved =
 Project DISTINCT[$4](
   Join[$1=$1](qterm, term)) ;


lambda_docModel =
 Join []( lambda\_doc, docModel);


lambda_collModel =
 Join []( lambda_coll , collModel) ;


# lambda_collModel_retrieved (Term, DocId):
 lambda_collModel_retrieved =
 Join []( lambda_collModel, retrieved ) ;


# lm1_term_weight(Term, DocId):
lm1_term_weight =
 Project DISJOINT(
 Unite ALL(lambda_docModel,
             lambda_collModel_retrieved )) ;


# lm1_retrieve (DocId, QueryId):
 lm1_retrieve =
 Project PROD[$4,$2](
   Join[$1=$1](qterm,
               lm1_term_weight)) ;
```

A.2 TF-IDF Implementation in PRA

TF-IDF in PRA:

```
# TF−IDF Retrieval
# TFIDF/main.pra

# tf_sum(Term, DocId):
tf_sum =
 Project DISJOINT(
  Bayes[$2](term));

# tf_max(Term, DocId):
tf_max =
 Bayes SUBSUMED[$2](tf_sum);

#  tf_frac (Term, DocId): see  text .

# idf(Term):
idf =
 Bayes MAX_IDF[](
  Project[$1](term));

# Query term weighting:
# qterm_idf(Term, QueryId):
qterm_idf =
 Project[$1,$2](
  Join[$1=$1](qterm, idf));

# Normalisation:
qterm_norm_idf =
 Bayes[$2](qterm_idf);
```

```
# Customisable setting of tf:
tf = tf_frac ;

# tfidf_retrieve (DocId, QueryId):
 tfidf_retrieve =
 Project DISJOINT[$4,$2](
  Join[$1=$1](qterm_norm_idf,
               tf));
```

Appendix B

DUC: ROUGE-1 Scores for Topics 1 – 50

The following tables contains the ROUGE-1 recall scores for the different summarisation models on the DUC collection. The first table provides the average ROUGE-1 scores achieved by DUC summarisers. The remaining tables show performance figures for the POLIS summarisation models, where the first column holds the topic number, the third column the measure (here: recall), the fourth column the average score, and the fifth column the 95% confidence interval for the reported score.

Topic	ROUGE score
1	0.30217
2	0.38594
3	0.38265
4	0.38033
5	0.39862
6	0.38511
7	0.37201
8	0.38807
9	0.36429
10	0.40361
11	0.22340
12	0.40488
13	0.38462
14	0.37947
15	0.38991
16	0.36681
17	0.33595
18	0.36062
19	0.37838
20	0.36921
21	0.35286
22	0.38047
23	0.40440
24	**0.40980**
25	0.36765
26	0.33071
27	0.38751
28	0.39922
29	0.37398
30	0.36105
31	0.39908
32	0.38196
33	0.40206
34	0.35973
35	0.33305

Table B.1: DUC average ROUGE-1 recall scores.

Topic	ROUGE test	Measure	ROUGE score	Confidence Interval
1	ROUGE-1	Average_R:	0.37353	(95%-conf.int. 0.35963 - 0.38299)
2	ROUGE-1	Average_R:	0.31924	(95%-conf.int. 0.31118 - 0.32651)
3	ROUGE-1	Average_R:	0.34940	(95%-conf.int. 0.33901 - 0.35977)
4	ROUGE-1	Average_R:	0.35513	(95%-conf.int. 0.34887 - 0.36601)
5	ROUGE-1	Average_R:	0.32621	(95%-conf.int. 0.31522 - 0.33724)
6	ROUGE-1	Average_R:	0.32201	(95%-conf.int. 0.30974 - 0.33548)
7	ROUGE-1	Average_R:	0.30179	(95%-conf.int. 0.29721 - 0.30636)
8	ROUGE-1	Average_R:	0.31975	(95%-conf.int. 0.30048 - 0.33514)
9	ROUGE-1	Average_R:	0.43494	(95%-conf.int. 0.43055 - 0.44015)
10	ROUGE-1	Average_R:	0.34678	(95%-conf.int. 0.34216 - 0.35113)
11	ROUGE-1	Average_R:	0.31900	(95%-conf.int. 0.31134 - 0.32789)
12	ROUGE-1	Average_R:	0.33835	(95%-conf.int. 0.33132 - 0.34594)
13	ROUGE-1	Average_R:	0.27498	(95%-conf.int. 0.26440 - 0.28552)
14	ROUGE-1	Average_R:	**0.44880**	(95%-conf.int. 0.43349 - 0.46190)
15	ROUGE-1	Average_R:	0.33935	(95%-conf.int. 0.33401 - 0.34469)
16	ROUGE-1	Average_R:	0.41509	(95%-conf.int. 0.40860 - 0.42150)
17	ROUGE-1	Average_R:	0.33627	(95%-conf.int. 0.33355 - 0.33903)
18	ROUGE-1	Average_R:	0.35484	(95%-conf.int. 0.34273 - 0.37284)
19	ROUGE-1	Average_R:	0.39001	(95%-conf.int. 0.38160 - 0.39844)
20	ROUGE-1	Average_R:	0.36685	(95%-conf.int. 0.35310 - 0.37867)
21	ROUGE-1	Average_R:	0.35060	(95%-conf.int. 0.34281 - 0.35842)
22	ROUGE-1	Average_R:	0.44210	(95%-conf.int. 0.43732 - 0.44511)
23	ROUGE-1	Average_R:	0.32995	(95%-conf.int. 0.32473 - 0.33509)
24	ROUGE-1	Average_R:	0.35584	(95%-conf.int. 0.34806 - 0.36364)
25	ROUGE-1	Average_R:	0.35657	(95%-conf.int. 0.34765 - 0.36673)
26	ROUGE-1	Average_R:	0.41113	(95%-conf.int. 0.38670 - 0.43701)
27	ROUGE-1	Average_R:	0.36674	(95%-conf.int. 0.35720 - 0.37324)
28	ROUGE-1	Average_R:	0.33570	(95%-conf.int. 0.32963 - 0.34661)
29	ROUGE-1	Average_R:	0.29984	(95%-conf.int. 0.29502 - 0.30466)
30	ROUGE-1	Average_R:	0.34156	(95%-conf.int. 0.33008 - 0.35767)
31	ROUGE-1	Average_R:	0.41560	(95%-conf.int. 0.40514 - 0.42608)
32	ROUGE-1	Average_R:	0.37647	(95%-conf.int. 0.36176 - 0.39037)
33	ROUGE-1	Average_R:	0.26148	(95%-conf.int. 0.23508 - 0.29131)
34	ROUGE-1	Average_R:	0.38855	(95%-conf.int. 0.37516 - 0.41103)
35	ROUGE-1	Average_R:	0.35823	(95%-conf.int. 0.35571 - 0.36074)
36	ROUGE-1	Average_R:	0.37706	(95%-conf.int. 0.37124 - 0.38619)
37	ROUGE-1	Average_R:	0.35809	(95%-conf.int. 0.35085 - 0.36469)
38	ROUGE-1	Average_R:	0.29596	(95%-conf.int. 0.28287 - 0.30511)
39	ROUGE-1	Average_R:	0.36992	(95%-conf.int. 0.36104 - 0.38385)
40	ROUGE-1	Average_R:	0.35313	(95%-conf.int. 0.34276 - 0.35950)
41	ROUGE-1	Average_R:	0.32699	(95%-conf.int. 0.31507 - 0.33643)
42	ROUGE-1	Average_R:	0.38511	(95%-conf.int. 0.37773 - 0.39248)
43	ROUGE-1	Average_R:	0.39586	(95%-conf.int. 0.38120 - 0.41183)
44	ROUGE-1	Average_R:	0.34792	(95%-conf.int. 0.34054 - 0.35354)
45	ROUGE-1	Average_R:	0.36856	(95%-conf.int. 0.35721 - 0.37661)
46	ROUGE-1	Average_R:	0.27244	(95%-conf.int. 0.26189 - 0.28121)
47	ROUGE-1	Average_R:	0.41865	(95%-conf.int. 0.40219 - 0.43507)
48	ROUGE-1	Average_R:	0.34968	(95%-conf.int. 0.33510 - 0.36331)
49	ROUGE-1	Average_R:	0.35477	(95%-conf.int. 0.34925 - 0.35978)
50	ROUGE-1	Average_R:	0.33903	(95%-conf.int. 0.33245 - 0.34552)

Table B.2: ROUGE-1 recall scores on DUC collection for the POLIS-base summarisation model.

Topic	ROUGE test	Measure	ROUGE score	Confidence Interval
1	ROUGE-1	Average_R:	0.29820	(95%-conf.int. 0.29078 - 0.30511)
2	ROUGE-1	Average_R:	0.30831	(95%-conf.int. 0.30529 - 0.31090)
3	ROUGE-1	Average_R:	0.36309	(95%-conf.int. 0.35859 - 0.36763)
4	ROUGE-1	Average_R:	0.25782	(95%-conf.int. 0.24919 - 0.26641)
5	ROUGE-1	Average_R:	0.29124	(95%-conf.int. 0.28219 - 0.30034)
6	ROUGE-1	Average_R:	0.29518	(95%-conf.int. 0.28402 - 0.31209)
7	ROUGE-1	Average_R:	0.31375	(95%-conf.int. 0.30871 - 0.31691)
8	ROUGE-1	Average_R:	0.25758	(95%-conf.int. 0.24768 - 0.26750)
9	ROUGE-1	Average_R:	0.39224	(95%-conf.int. 0.38278 - 0.40172)
10	ROUGE-1	Average_R:	0.34195	(95%-conf.int. 0.33386 - 0.34949)
11	ROUGE-1	Average_R:	0.33234	(95%-conf.int. 0.32993 - 0.33534)
12	ROUGE-1	Average_R:	0.35333	(95%-conf.int. 0.33796 - 0.36769)
13	ROUGE-1	Average_R:	0.29790	(95%-conf.int. 0.28938 - 0.30667)
14	ROUGE-1	Average_R:	0.36549	(95%-conf.int. 0.35290 - 0.37809)
15	ROUGE-1	Average_R:	0.33331	(95%-conf.int. 0.32666 - 0.34046)
16	ROUGE-1	Average_R:	0.34574	(95%-conf.int. 0.33637 - 0.35589)
17	ROUGE-1	Average_R:	0.32740	(95%-conf.int. 0.32258 - 0.33224)
18	ROUGE-1	Average_R:	0.39799	(95%-conf.int. 0.39019 - 0.40803)
19	ROUGE-1	Average_R:	0.31926	(95%-conf.int. 0.31678 - 0.32173)
20	ROUGE-1	Average_R:	0.30806	(95%-conf.int. 0.29493 - 0.32120)
21	ROUGE-1	Average_R:	0.35879	(95%-conf.int. 0.34884 - 0.36830)
22	ROUGE-1	Average_R:	0.33931	(95%-conf.int. 0.32269 - 0.35596)
23	ROUGE-1	Average_R:	0.37447	(95%-conf.int. 0.36666 - 0.38129)
24	ROUGE-1	Average_R:	0.37739	(95%-conf.int. 0.36268 - 0.38940)
25	ROUGE-1	Average_R:	0.41821	(95%-conf.int. 0.41661 - 0.42012)
26	ROUGE-1	Average_R:	0.41718	(95%-conf.int. 0.39267 - 0.44244)
27	ROUGE-1	Average_R:	0.38474	(95%-conf.int. 0.37433 - 0.39236)
28	ROUGE-1	Average_R:	0.35263	(95%-conf.int. 0.33984 - 0.36260)
29	ROUGE-1	Average_R:	0.30081	(95%-conf.int. 0.29046 - 0.31216)
30	ROUGE-1	Average_R:	0.33078	(95%-conf.int. 0.32036 - 0.34286)
31	ROUGE-1	Average_R:	0.37420	(95%-conf.int. 0.36803 - 0.38129)
32	ROUGE-1	Average_R:	**0.42845**	(95%-conf.int. 0.41748 - 0.43726)
33	ROUGE-1	Average_R:	0.25081	(95%-conf.int. 0.23220 - 0.27612)
34	ROUGE-1	Average_R:	0.36604	(95%-conf.int. 0.35140 - 0.38336)
35	ROUGE-1	Average_R:	0.32093	(95%-conf.int. 0.31389 - 0.32861)
36	ROUGE-1	Average_R:	0.33790	(95%-conf.int. 0.32846 - 0.34432)
37	ROUGE-1	Average_R:	0.36692	(95%-conf.int. 0.35559 - 0.38236)
38	ROUGE-1	Average_R:	0.30559	(95%-conf.int. 0.30161 - 0.31194)
39	ROUGE-1	Average_R:	0.36207	(95%-conf.int. 0.35304 - 0.37396)
40	ROUGE-1	Average_R:	0.33336	(95%-conf.int. 0.32311 - 0.34235)
41	ROUGE-1	Average_R:	0.35574	(95%-conf.int. 0.34274 - 0.37308)
42	ROUGE-1	Average_R:	0.33370	(95%-conf.int. 0.32561 - 0.34181)
43	ROUGE-1	Average_R:	0.33010	(95%-conf.int. 0.31266 - 0.34845)
44	ROUGE-1	Average_R:	0.34176	(95%-conf.int. 0.33024 - 0.35436)
45	ROUGE-1	Average_R:	0.35591	(95%-conf.int. 0.34786 - 0.36425)
46	ROUGE-1	Average_R:	0.36185	(95%-conf.int. 0.35601 - 0.36770)
47	ROUGE-1	Average_R:	0.38217	(95%-conf.int. 0.37518 - 0.38976)
48	ROUGE-1	Average_R:	0.36993	(95%-conf.int. 0.36106 - 0.38000)
49	ROUGE-1	Average_R:	0.30930	(95%-conf.int. 0.30653 - 0.31261)
50	ROUGE-1	Average_R:	0.29129	(95%-conf.int. 0.28543 - 0.29538)

Table B.3: ROUGE-1 recall scores on DUC collection for the POLIS-microaveraging summarisation model.

Topic	ROUGE test	Measure	ROUGE score	Confidence Interval
1	ROUGE-1	Average_R:	0.29721	(95%-conf.int. 0.28955 - 0.30128)
2	ROUGE-1	Average_R:	0.29939	(95%-conf.int. 0.29682 - 0.30263)
3	ROUGE-1	Average_R:	0.38370	(95%-conf.int. 0.37752 - 0.39012)
4	ROUGE-1	Average_R:	0.25202	(95%-conf.int. 0.24296 - 0.26361)
5	ROUGE-1	Average_R:	0.29512	(95%-conf.int. 0.28478 - 0.30552)
6	ROUGE-1	Average_R:	0.29222	(95%-conf.int. 0.28160 - 0.30580)
7	ROUGE-1	Average_R:	0.29781	(95%-conf.int. 0.29466 - 0.30104)
8	ROUGE-1	Average_R:	0.32097	(95%-conf.int. 0.31107 - 0.33086)
9	ROUGE-1	Average_R:	0.42107	(95%-conf.int. 0.41417 - 0.42639)
10	ROUGE-1	Average_R:	0.40633	(95%-conf.int. 0.40561 - 0.40752)
11	ROUGE-1	Average_R:	0.27130	(95%-conf.int. 0.26317 - 0.27866)
12	ROUGE-1	Average_R:	0.35636	(95%-conf.int. 0.34298 - 0.37170)
13	ROUGE-1	Average_R:	0.32080	(95%-conf.int. 0.31244 - 0.32755)
14	ROUGE-1	Average_R:	0.35356	(95%-conf.int. 0.34785 - 0.35787)
15	ROUGE-1	Average_R:	0.33332	(95%-conf.int. 0.32988 - 0.33738)
16	ROUGE-1	Average_R:	0.40531	(95%-conf.int. 0.39214 - 0.42168)
17	ROUGE-1	Average_R:	0.35993	(95%-conf.int. 0.35562 - 0.36399)
18	ROUGE-1	Average_R:	0.39410	(95%-conf.int. 0.38446 - 0.40380)
19	ROUGE-1	Average_R:	0.41357	(95%-conf.int. 0.40835 - 0.41880)
20	ROUGE-1	Average_R:	0.35392	(95%-conf.int. 0.34287 - 0.36496)
21	ROUGE-1	Average_R:	0.41038	(95%-conf.int. 0.40553 - 0.41456)
22	ROUGE-1	Average_R:	0.36525	(95%-conf.int. 0.34730 - 0.38323)
23	ROUGE-1	Average_R:	0.41388	(95%-conf.int. 0.40155 - 0.42718)
24	ROUGE-1	Average_R:	0.44286	(95%-conf.int. 0.43575 - 0.44988)
25	ROUGE-1	Average_R:	**0.44954**	(95%-conf.int. 0.44467 - 0.45571)
26	ROUGE-1	Average_R:	0.43327	(95%-conf.int. 0.40920 - 0.46075)
27	ROUGE-1	Average_R:	0.40776	(95%-conf.int. 0.39389 - 0.41851)
28	ROUGE-1	Average_R:	0.38516	(95%-conf.int. 0.37555 - 0.39307)
29	ROUGE-1	Average_R:	0.32808	(95%-conf.int. 0.32286 - 0.33336)
30	ROUGE-1	Average_R:	0.36515	(95%-conf.int. 0.35278 - 0.37756)
31	ROUGE-1	Average_R:	0.38205	(95%-conf.int. 0.37666 - 0.38569)
32	ROUGE-1	Average_R:	0.44621	(95%-conf.int. 0.42988 - 0.46236)
33	ROUGE-1	Average_R:	0.28969	(95%-conf.int. 0.27196 - 0.31172)
34	ROUGE-1	Average_R:	0.35529	(95%-conf.int. 0.33973 - 0.37447)
35	ROUGE-1	Average_R:	0.34488	(95%-conf.int. 0.33897 - 0.34970)
36	ROUGE-1	Average_R:	0.33275	(95%-conf.int. 0.32370 - 0.34169)
37	ROUGE-1	Average_R:	0.37280	(95%-conf.int. 0.36504 - 0.38069)
38	ROUGE-1	Average_R:	0.34331	(95%-conf.int. 0.33439 - 0.35424)
39	ROUGE-1	Average_R:	0.40220	(95%-conf.int. 0.38767 - 0.41740)
40	ROUGE-1	Average_R:	0.37095	(95%-conf.int. 0.35981 - 0.37997)
41	ROUGE-1	Average_R:	0.39140	(95%-conf.int. 0.37559 - 0.40741)
42	ROUGE-1	Average_R:	0.35987	(95%-conf.int. 0.35390 - 0.36581)
43	ROUGE-1	Average_R:	0.33991	(95%-conf.int. 0.32572 - 0.35468)
44	ROUGE-1	Average_R:	0.36616	(95%-conf.int. 0.35382 - 0.38001)
45	ROUGE-1	Average_R:	0.35496	(95%-conf.int. 0.33868 - 0.36684)
46	ROUGE-1	Average_R:	0.38463	(95%-conf.int. 0.37554 - 0.39168)
47	ROUGE-1	Average_R:	0.39657	(95%-conf.int. 0.39160 - 0.39975)
48	ROUGE-1	Average_R:	0.36592	(95%-conf.int. 0.35671 - 0.37627)
49	ROUGE-1	Average_R:	0.29550	(95%-conf.int. 0.28967 - 0.30125)
50	ROUGE-1	Average_R:	0.31114	(95%-conf.int. 0.30790 - 0.31454)

Table B.4: ROUGE-1 recall scores on DUC collection for the POLIS-microaveragingIDF summarisation model.

Topic	ROUGE test	Measure	ROUGE score	Confidence Interval
1	ROUGE-1	Average_R:	0.36952	(95%-conf.int. 0.35367 - 0.38165)
2	ROUGE-1	Average_R:	0.33002	(95%-conf.int. 0.32716 - 0.33289)
3	ROUGE-1	Average_R:	0.38569	(95%-conf.int. 0.37652 - 0.39443)
4	ROUGE-1	Average_R:	0.28310	(95%-conf.int. 0.27508 - 0.29111)
5	ROUGE-1	Average_R:	0.34368	(95%-conf.int. 0.33592 - 0.35414)
6	ROUGE-1	Average_R:	0.29917	(95%-conf.int. 0.28536 - 0.31409)
7	ROUGE-1	Average_R:	0.31573	(95%-conf.int. 0.31051 - 0.32176)
8	ROUGE-1	Average_R:	0.31697	(95%-conf.int. 0.30430 - 0.32671)
9	ROUGE-1	Average_R:	0.39126	(95%-conf.int. 0.38278 - 0.39975)
10	ROUGE-1	Average_R:	0.33427	(95%-conf.int. 0.32733 - 0.34120)
11	ROUGE-1	Average_R:	0.30128	(95%-conf.int. 0.29980 - 0.30330)
12	ROUGE-1	Average_R:	0.36631	(95%-conf.int. 0.35130 - 0.37872)
13	ROUGE-1	Average_R:	0.30985	(95%-conf.int. 0.30124 - 0.31867)
14	ROUGE-1	Average_R:	0.43102	(95%-conf.int. 0.41461 - 0.44616)
15	ROUGE-1	Average_R:	0.35929	(95%-conf.int. 0.35535 - 0.36166)
16	ROUGE-1	Average_R:	0.36722	(95%-conf.int. 0.35258 - 0.37785)
17	ROUGE-1	Average_R:	0.32152	(95%-conf.int. 0.31393 - 0.32709)
18	ROUGE-1	Average_R:	0.40290	(95%-conf.int. 0.39100 - 0.41866)
19	ROUGE-1	Average_R:	0.39291	(95%-conf.int. 0.38638 - 0.39778)
20	ROUGE-1	Average_R:	0.33201	(95%-conf.int. 0.31321 - 0.34612)
21	ROUGE-1	Average_R:	0.41570	(95%-conf.int. 0.40465 - 0.42662)
22	ROUGE-1	Average_R:	0.35926	(95%-conf.int. 0.34198 - 0.37659)
23	ROUGE-1	Average_R:	0.43596	(95%-conf.int. 0.42753 - 0.44439)
24	ROUGE-1	Average_R:	0.39691	(95%-conf.int. 0.38831 - 0.40687)
25	ROUGE-1	Average_R:	0.40153	(95%-conf.int. 0.39553 - 0.40756)
26	ROUGE-1	Average_R:	**0.44538**	(95%-conf.int. 0.42016 - 0.46920)
27	ROUGE-1	Average_R:	0.37476	(95%-conf.int. 0.36347 - 0.38162)
28	ROUGE-1	Average_R:	0.36639	(95%-conf.int. 0.35769 - 0.37918)
29	ROUGE-1	Average_R:	0.33793	(95%-conf.int. 0.32597 - 0.34997)
30	ROUGE-1	Average_R:	0.37692	(95%-conf.int. 0.36706 - 0.38680)
31	ROUGE-1	Average_R:	0.35541	(95%-conf.int. 0.35178 - 0.36085)
32	ROUGE-1	Average_R:	0.42653	(95%-conf.int. 0.40856 - 0.44052)
33	ROUGE-1	Average_R:	0.29054	(95%-conf.int. 0.26724 - 0.31933)
34	ROUGE-1	Average_R:	0.39052	(95%-conf.int. 0.37256 - 0.40635)
35	ROUGE-1	Average_R:	0.35243	(95%-conf.int. 0.33952 - 0.36584)
36	ROUGE-1	Average_R:	0.38825	(95%-conf.int. 0.38134 - 0.39488)
37	ROUGE-1	Average_R:	0.35519	(95%-conf.int. 0.34483 - 0.36760)
38	ROUGE-1	Average_R:	0.34036	(95%-conf.int. 0.33171 - 0.35639)
39	ROUGE-1	Average_R:	0.35327	(95%-conf.int. 0.34675 - 0.36509)
40	ROUGE-1	Average_R:	0.33632	(95%-conf.int. 0.32583 - 0.34301)
41	ROUGE-1	Average_R:	0.34485	(95%-conf.int. 0.33222 - 0.36023)
42	ROUGE-1	Average_R:	0.33952	(95%-conf.int. 0.33131 - 0.35406)
43	ROUGE-1	Average_R:	0.33501	(95%-conf.int. 0.31919 - 0.35206)
44	ROUGE-1	Average_R:	0.38044	(95%-conf.int. 0.37408 - 0.38744)
45	ROUGE-1	Average_R:	0.40342	(95%-conf.int. 0.38931 - 0.41943)
46	ROUGE-1	Average_R:	0.33900	(95%-conf.int. 0.33520 - 0.34425)
47	ROUGE-1	Average_R:	0.43214	(95%-conf.int. 0.42471 - 0.43955)
48	ROUGE-1	Average_R:	0.38493	(95%-conf.int. 0.38010 - 0.39202)
49	ROUGE-1	Average_R:	0.35572	(95%-conf.int. 0.35088 - 0.36008)
50	ROUGE-1	Average_R:	0.37182	(95%-conf.int. 0.36506 - 0.37798)

Table B.5: ROUGE-1 recall scores on DUC collection for the POLIS-macroaveraging summarisation model.

Appendix C

INEX: ROUGE-1 Scores; Average per Journal

Journal	ROUGE-1
an	0.34389207547169814
cg	0.2488990196078432
co	0.25854114285714286
cs	0.298127614213198
dt	0.28833311787072236
ex	0.2997072881355932
ic	0.230067
it	0.1948703124999999
mi	0.25363199999999997
mu	0.31956434210526313
pd	0.30013701492537304
so	0.2950734120171676
tc	0.48195892259414297
td	**0.49054788331071814**
tg	0.45947553299492366
tk	0.48975546816479404
tp	0.47534956256358096
ts	0.48979393442622937

Journal	ROUGE-1
an	0.6377633962264151
cg	0.695362450980393
co	0.6701058345864659
cs	0.6426657360406088
dt	**0.712043954372623**
ex	0.683579423288131
ic	0.522959
it	0.5996228125
mi	0.6869784333333337
mu	0.6950623684210528
pd	0.6734108208955223
so	0.6722167381974251
tc	0.6526951778242678
td	0.6704279647218461
tg	0.6441375634517764
tk	0.6583094943820226
tp	0.6427836419125124
ts	0.669254774590164

Table C.1: INEX-Collection. Recall: POLIS-base vs original abstracts; POLIS-base vs Marcu's extracts

Journal	ROUGE-1
an	0.14147976415094352
cg	0.08096624183006536
co	0.08710593984962399
cs	0.10513898477157359
dt	0.10604216730038032
ex	0.1064160338983051
ic	0.08052409090909092
it	0.05469487500000001
mi	0.09007793333333337
mu	0.12241322368421056
pd	0.1042164925373134
so	0.09876989270386266
tc	0.2215873221757326
td	**0.23041687924016302**
tg	0.21274923857868025
tk	0.22697220973782783
tp	0.21536057985757878
ts	0.2232866393442624

Journal	ROUGE-1
an	0.2699928773584907
cg	0.29851885620915064
co	0.3063960601503761
cs	0.30898492385786785
dt	0.3341155133079847
ex	0.3130567796610171
ic	0.28393122727272724
it	0.27040587500000013
mi	0.3135486000000002
mu	0.32006230263157887
pd	**0.3359692537313434**
so	0.3056566523605151
tc	0.3022665690376569
td	0.31179972862957944
tg	0.32972035532994926
tk	0.3135327153558057
tp	0.31993610376398784
ts	0.3358795286885247

Table C.2: INEX-Collection. Precision: POLIS-base vs original abstracts; POLIS-base vs Marcu's extracts

Journal	ROUGE-1
an	0.1890062264150943
cg	0.25207297385620914
co	0.2247172781954888
cs	0.24366700507614206
dt	0.2809469201520916
ex	0.24940308474576242
ic	0.17301518181818173
it	0.17744725
mi	0.22578513333333328
mu	0.30903460526315785
pd	0.22390962686567173
so	0.2308838626609441
tc	**0.4734808263598333**
td	0.47236377204884666
tg	0.4611667005076139
tk	0.43905187265917617
tp	0.44619381485249215
ts	0.43427559426229495

Journal	ROUGE-1
an	0.2343016509433961
cg	0.509615980392157
co	0.4034913984962399
cs	0.3814208629441624
dt	0.519719391634981
ex	0.43190793220338974
ic	0.24706186363636365
it	0.36558337500000004
mi	0.4471382333333334
mu	0.4889377631578946
pd	0.3884671641791046
so	0.4046725536480685
tc	**0.5585637866108791**
td	0.5572870420624153
tg	0.5108751269035533
tk	0.5014790262172281
tp	0.5101416683621567
ts	0.5011666393442625

Table C.3: INEX-Collection. Recall: POLIS-micro vs original abstracts; POLIS-micro vs Marcu's extracts

Journal	ROUGE-1
an	0.09225495283018864
cg	0.09896767973856217
co	0.0878240751879699 7
cs	0.09773527918781727
dt	0.11864771863117869
ex	0.10443454237288138
ic	0.06620436363636364
it	0.0627249375
mi	0.09258313333333332
mu	0.1319455921052631
pd	0.09053902985074629
so	0.09404532188841208
tc	0.22865742677824255
td	**0.2333963500678429**
tg	0.2235339593908629
tk	0.21121739700374537
tp	0.21230906408952185
ts	0.2071182377049181

Journal	ROUGE-1
an	0.1720137735849056
cg	0.26481307189542475
co	0.2654888270676693
cs	0.2611629441624365
dt	0.30091741444866915
ex	0.2734905084745763
ic	0.20970954545454545
it	0.2282473125000001
mi	0.2762057333333333
mu	0.2927291447368422
pd	0.244105223880597
so	0.2577259442060086
tc	0.30490086820083645
td	0.30132396200814116
tg	**0.3388368020304567**
tk	0.2899071348314607
tp	0.3095150152594094
ts	0.30024489754098377

Table C.4: INEX-Collection. Precision: POLIS-micro vs original abstracts; POLIS-micro vs Marcu's extracts

Journal	ROUGE-1
an	0.17231463836477978
cg	0.3930914705882351
co	0.30663950375939897
cs	0.28905825719120165
dt	0.3801755133079846
ex	0.3428977288135599
ic	0.2247700454545456
it	0.2502495
mi	0.3204727777777772
mu	0.371696951754386
pd	0.30043373134328377
so	0.26499386981402007
tc	0.4091903242677829
td	**0.4136996426956134**
tg	0.3915023519458547
tk	0.38627681647940054
tp	0.3844730077992542
ts	0.3862930942622952

Journal	ROUGE-1
an	0.24050433962264148
cg	0.3974744771241828
co	0.3791332180451128
cs	0.3722195939086296
dt	0.4427400380228133
ex	0.3922057288135594
ic	0.2841429090909091
it	0.33610875000000007
mi	0.40171316666666673
mu	0.4179426315789473
pd	0.3478614925373135
so	0.3706750214592277
tc	0.4479087133891213
td	0.4372163636363634
tg	**0.4723798477157362**
tk	0.41322146067415755
tp	0.44268483214649024
ts	0.4248040368852459

Table C.5: INEX-Collection. Recall: POLIS-microIDF vs original abstracts; POLIS-microIDF vs Marcu's extracts

Journal	ROUGE-1
an	0.035738836477987396
cg	0.025174564270152484
co	0.027094952380952393
cs	0.028318206429780026
dt	0.03522645120405578
ex	0.03096744632768365
ic	0.020299121212121206
it	0.021334812500000015
mi	0.027306888888888905
mu	0.043781228070175426
pd	0.025070547263681606
so	0.03004911301859798
tc	0.13663715829846554
td	**0.13819085481682497**
tg	0.11458314720812186
tk	0.11712127340823955
tp	0.11982093591047822
ts	0.10313979508196736

Journal	ROUGE-1
an	0.17364547169811326
cg	0.3640929084967318
co	0.2999345263157899
cs	0.29781203045685267
dt	0.3858223574144486
ex	0.32370952542372883
ic	0.1988307272727272
it	0.2719899999999999
mi	0.3392816
mu	0.3518180263157894
pd	0.2916663432835821
so	0.2967220815450645
tc	**0.4032701255230121**
td	0.3945467842605157
tg	0.38013477157360404
tk	0.3574511235955054
tp	0.3758466531027467
ts	0.36017233606557403

Table C.6: INEX-Collection. Precision: POLIS-microIDF vs original abstracts; POLIS-microIDF vs Marcu's extracts

Journal	ROUGE-1
an	0.21509943396226422
cg	0.20456816993464041
co	0.19208445112781974
cs	0.20841213197969546
dt	0.24622505703422068
ex	0.22450911864406778
ic	0.1474709545454545
it	0.1639028749999999
mi	0.1957546333333334
mu	0.2616644736842104
pd	0.19134462686567164
so	0.22882757510729615
tc	**0.44138849372385003**
td	0.44000465400271327
tg	0.4097475126903551
tk	0.4071582022471908
tp	0.4158732756866733
ts	0.399394569672131

Journal	ROUGE-1
an	0.4428047169811323
cg	0.5800016339869287
co	0.49638139849624047
cs	0.4541225380710658
dt	0.6017020152091255
ex	0.5043637288135594
ic	0.3195923636363636
it	0.4580986875
mi	0.5496283666666667
mu	0.559822697368421
pd	0.45113380597014935
so	0.5125700643776827
tc	0.603080711297071
td	**0.6067400135685218**
tg	0.5690524365482231
tk	0.5427543445692877
tp	0.5670457070193284
ts	0.5484321311475407

Table C.7: INEX-Collection. Recall: POLIS-macro vs original abstracts; POLIS-macro vs Marcu's extracts

Journal	ROUGE-1
an	0.09263613207547168
cg	0.06750957516339869
co	0.0641466165413534
cs	0.07515411167512688
dt	0.08974889733840312
ex	0.0799206779661017
ic	0.05124572727272728
it	0.045331437499999995
mi	0.06958696666666667
mu	0.09846993421052634
pd	0.06453671641791045
so	0.07523347639484979
tc	0.20364007322175764
td	**0.20613540027137045**
tg	0.18873472081218282
tk	0.18754029962546803
tp	0.18878784333672433
ts	0.1830691393442624

Journal	ROUGE-1
an	0.15634547169811322
cg	0.2403576797385622
co	0.23010827067669176
cs	0.23229167512690344
dt	0.27004836501901136
ex	0.242178813559322
ic	0.18187663636363638
it	0.2036113125
mi	0.24393363333333326
mu	0.2549630921052631
pd	0.20938119402985075
so	0.23172128755364813
tc	0.2784496129707114
td	0.2784205970149256
tg	**0.29608395939086285**
tk	0.2630443632958803
tp	0.27589408952187183
ts	0.27035704918032805

Table C.8: INEX-Collection. Precision: POLIS-macro vs original abstracts; POLIS-macro vs Marcu's extracts

Journal	ROUGE-1
an	0.32295853773584915
cg	0.33076225490196065
co	0.29104230075187976
cs	0.3070127918781725
dt	0.32370923954372616
ex	0.32576128813559313
ic	0.264137818181818
it	0.2322894374999999
mi	0.2822254333333334
mu	0.34226407894736843
pd	0.33536246268656705
so	0.28423023605150255
tc	0.46272995815899565
td	0.46850862957937556
tg	0.445160456852792
tk	**0.46918249063670403**
tp	0.4516266632756866
ts	0.4555679713114755

Journal	ROUGE-1
an	0.5321981603773586
cg	**0.5791096405228755**
co	0.5484226917293228
cs	0.5114591878172591
dt	0.5623119391634983
ex	0.5531133898305085
ic	0.4163977272727273
it	0.5019038750000001
mi	0.5399204666666666
mu	0.5723775
pd	0.547249626865672
so	0.5452550214592274
tc	0.534902029288703
td	0.5472337042062421
tg	0.522197157360406
tk	0.5393693820224716
tp	0.52527022380468
ts	0.5411078073770493

Table C.9: INEX-Collection. Recall: KTL vs original abstracts; KTL vs Marcu's extracts

Journal	ROUGE-1
an	0.2885173584905659
cg	0.2516337581699345
co	0.22263790977443576
cs	0.2509069543147208
dt	0.2714074144486694
ex	0.2616018644067796
ic	0.20128431818181822
it	0.15563962499999995
mi	0.22808170000000005
mu	0.2984611842105263
pd	0.2755928358208955
so	0.2230779613733908
tc	0.44088719665271875
td	**0.45554339213025763**
tg	0.4256608629441621
tk	0.44612395131086136
tp	0.423412472024415
ts	0.43376342213114755

Journal	ROUGE-1
an	0.39821580188679234
cg	0.4573753921568628
co	0.4608189624060147
cs	0.4631294923857869
dt	**0.5002965019011408**
ex	0.473315254237288
ic	0.4092153636363638
it	0.41307343750000014
mi	0.46864409999999973
mu	0.4748613815789476
pd	0.4955298507462687
so	0.457082103004292
tc	0.4627897384937239
td	0.46714099050203545
tg	0.4866605076142133
tk	0.46674979400749034
tp	0.475222797558494
ts	0.4886447950819669

Table C.10: INEX-Collection. Precision: KTL vs original abstracts; KTL vs Marcu's extracts

Appendix D

KTL Feature-based Summariser

To evaluate the POLIS summarisation models on the INEX collection, we additionally applied a feature based summariser to the documents in the collection, so that the performance of the POLIS models could be judged against a traditional summarisation approach. The feature based summariser makes use of the term frequency (key method), title, and location features, which are combined in a linear fashion, such that all features are given equal weights. The score of a sentence *S* is thus given by a linear combination of the sentence's feature scores:

$$RSV(S) \quad := \quad K(S) + T(S) + L(S) \tag{D.1}$$

We discuss the individual features in the remainder of this chapter.

D.1 Key Method

The Key feature was first introduced by Luhn in his seminal paper on summarisation, and has – in some form – been used in almost all feature-based summarisation systems since. The idea behind this feature is that important sentences contain a larger number of important terms. For our system, we define the importance of a term as its term-weight given by the traditional IR *TF* weighting function. For the Key method, we modelled TF in the traditional *BM25* approach. We define the key-weight of a sentence *S* as the sum of its term-weights, divided by the number of terms:

$$KW(S) \quad := \quad \frac{\sum_{t \in S} (P_L(t,d))}{|S|} \tag{D.2}$$

where $P_L(t,s)$ is the TF-weight of a term t in sentence s.

D.2 Title Method

The Title features makes use of the fact that the topic of a document is either hinted at or is explicitly stated in its title. Sentences which largely overlap with the title can thus be assumed to be more important. To measure this overlap, we divide the number of terms co-occurring in both sentence and title by the total number of terms in a sentence:

$$TW(S) \quad := \quad \frac{|t \in S \wedge t \in T|}{|S|} \tag{D.3}$$

where T is the title, $|t \in S \wedge t \in T|$ is the number of terms overlapping in T and S, and $|S|$ is the number of terms in S.

D.3 Location Method

The Location feature takes advantage of the notion that the position of a sentence in a document conveys information about its importance. However, there is no general consensus on which locations are most useful. Tombros and Sanderson report that in their experiments, the leading paragraph of articles conveys much of the content, in agreement with results by Brandow *et al.* (Tombros and Sanderson (1998); Brandow et al. (1995)). On the other hand, Edmundson reports that the most salient locations are found at the beginning and the end of documents (Edmundson (1969)), however, without providing details of where the beginning ends and the end begins. We therefore take an agnostic approach to the location feature, assuming that it is not a sharp cut-off, but rather a continuous "importance function" over the entire document.

We model this using an inverse Gaussian distribution. Consider the sentences of a document placed along the x-axis of a graph. The Gaussian distribution then measures the "level of unimportance", peaking at the center of the document, and sloping off towards the beginning and end of the document. For a sentence S at position P_s, the location weight is defined as:

$$LW(S) \quad := \quad 1 - \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{((P_S - \frac{|S|}{2}) - \mu)^2}{2\sigma^2}} \tag{D.4}$$

where $|S|$ is the total number of sentences, σ is the standard deviation, and μ is the expected value. To realise a smooth transition over all sentences in the document, we set σ to 5.0, μ to 0.0.

Appendix E

Manipulation of a Document-summary by varying POLIS expressions

E.1 Introduction

The discussion of this appendix follows closely the applicability scenario outlined in Section 8.2; here, we will give a more detailed example of how different POLIS expressions applied to the same input document will result in different summaries.

We will use the Wikipedia article on "Information Retrieval" as our sample document (Wikipedia (2009)). As a preliminary step, we again apply Piao's sentence splitter to the document, resulting in structural markup at sentence level.

Our discussion will be limited to the effects of varying the summarisation model used; as manipulating the summary-size quantifier (and the changes to the summary effected) is self-explanatory, it will remain fixed to "at_most 5" during our trials (i.e. up to five sentences can be extracted to form the summary).

E.2 Models

We will in turn apply all summarisation models provided by POLIS to the sample document, and discuss the summary produced. Where a summary query could be specified (i.e. for all the query-based summarisation models), we used “information retrieval” as our query.

E.2.1 POLIS-base model

To produce a non-query-biased summary of the sample document, we applied the following POLIS expression:

/doc/sent:{at_most 5}

The resulting summary is shown in Figure E.1.

Information retrieval (IR) is the science of searching for documents, for information within documents and for metadata about documents, as well as that of searching relational databases and the World Wide Web.
First implementations of information retrieval systems were introduced in the 1950s and 1960s.
Automated information retrieval systems are used to reduce what has been called ”information overload”.
There is overlap in the usage of the terms data retrieval, document retrieval, information retrieval, and text retrieval, but each also has its own body of literature, theory, praxis and technologies.
IR is interdisciplinary, based on computer science, mathematics, library science, information science, information architecture, cognitive psychology, linguistics, statistics and physics.

Figure E.1: Five sentence summary of the Wikipedia article on “Information Retrieval” produced using the POLIS-base model.

Although there was no query-bias in the creation of this summary, all sentences in the summary focus on either “information” or “retrieval”.

E.2.2 POLIS-microaveraging model

The first of the query-based summarisation approaches is the POLIS-microaveraging model. To apply this model to the sample document, we used the following POLIS expression:

/doc/sent{MICRO ”Information Retrieval”}:{at_most 5}

The resulting summary is shown in Figure E.2.

The summary indicates that the microaveraging approach favours shorter sentences over longer ones, compared to the (non query-based) POLIS-base model. Furthermore, the query-

* Information need
* Adversarial information retrieval
* Relevance (Information Retrieval)
* Information Retrieval Facility
o Fuzzy retrieval

Figure E.2: Five sentence summary of the Wikipedia article on “Information Retrieval” produced using the POLIS-microaveraging model.

term “information” is given a higher importance compared to “retrieval”, as “retrieval” is not featured in the top-ranking first sentence.

E.2.3 POLIS-microaveragingIDF model

Similarly to the POLIS-microaveraging model, the following expression results in a summary using the POLIS-microaveraging-IDF model:

/doc/sent{MICRO-IDF ”Information Retrieval”}:{at_most 5}

The resulting summary is shown in Figure E.3.

* Information need
* Adversarial information retrieval
* Relevance (Information Retrieval)
* Information Retrieval Facility
o Fuzzy retrieval

Figure E.3: Five sentence summary of the Wikipedia article on “Information Retrieval” produced using the POLIS-microaveragingIDF model.

The summary produced by the microaveraging-IDF approach is identical to the summary generated by the POLIS-microaveraging model. The same discussion thus applies.

E.2.4 POLIS-macroaveraging model

The query we used for the macroaveraging model was:

/doc/sent{MACRO ”Information Retrieval”}:{at_most 5}

Applying this query to the sample document resulted in the five sentence summary shown in Figure E.4.

Similar to the POLIS-base model, the POLIS-macroaveraging model favours longer sentences over shorter ones.

First implementations of information retrieval systems were introduced in the 1950s and 1960s.
1950: The term ”information retrieval” may have been coined by Calvin Mooers.
1959: Hans Peter Luhn published ”Auto-encoding of documents for information retrieval.”
In information retrieval a query does not uniquely identify a single object in the collection.
The idea of using computers to search for relevant pieces of information was popularized in an article As We May Think by Vannevar Bush in 1945.

Figure E.4: Five sentence summary of the Wikipedia article on “Information Retrieval” produced using the POLIS-macroaveraging model.

Index