

Fall 2011

N-Grams Assisted Long Web Search Query Optimization

Jehann Kersi Irani
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Computer Sciences Commons](#)

Recommended Citation

Irani, Jehann Kersi, "N-Grams Assisted Long Web Search Query Optimization" (2011). *Master's Projects*. 211.
DOI: <https://doi.org/10.31979/etd.t6km-tezz>
https://scholarworks.sjsu.edu/etd_projects/211

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

N-Grams Assisted Long Web Search Query Optimization

A Project Report

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Computer Science

By

Jehaan Kersi Irani

December 2011

© 2011

Jehaan Kersi Irani

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Teng Moh

Dr. Mark Stamp

Mr. Uzair Ali

APPROVED FOR THE UNIVERSITY

ABSTRACT

N-Grams Assisted Long Web Search Query Optimization

By Jehaan Kersi Irani

Commercial search engines do not return optimal search results when the query is a long or multi-topic one [1]. Long queries are used extensively. While the creator of the long query would most likely use natural language to describe the query, it contains extra information. This information dilutes the results of a web search, and hence decreases the performance as well as quality of the results returned. Kumaran et al. [22] showed that shorter queries extracted from longer user generated queries are more effective for ad-hoc retrieval. Hence reducing these queries by removing extra terms, the quality of the search results can be improved. There are numerous approaches used to address this shortfall. Our approach evaluates various versions of the query, thus trying to find the optimal one. This variation is achieved by reducing the query length using a combination of n-grams assisted query selection as well as a random keyword combination generator.

We look at existing approaches and try to improve upon them. We propose a hybrid model that tries to address the shortfalls of an existing technique by incorporating established methods along with new ideas. We use the existing models and plug in information with the help of n-grams as well as randomization to improve the overall performance while keeping any overhead calculations in check.

Acknowledgement

The author is deeply indebted to Dr. Teng Moh for his invaluable knowledge and guidance in the course of this study.

Table of Contents

Introduction	9
Related Work	10
Technologies and Projects used	11
The Clue Web 09 dataset	11
REST Services	12
Microsoft Web N-Gram Service (Public Beta) N-grams data	14
The Lemur Project	15
Indri Project	17
Indri Build Index	19
Query Clarity with retrieval	22
Retrieval User interface (RetUI)	24
The Experiment	26
Technique	26
The Original Approach	26
Our Approach	30
Implementation	31
Benchmarking and results	32
Conclusion	38
Future Scope	39
References	40
Appendix A: Code Snippets	42

List of Figures

Figure 1: Indri index build setup

Figure 2: Indri indexing in progress

Figure 3: Indices created by Indri for search and other retrieval-based applications

Figure 4: Sample Indri indexing parameters

Figure 5: Sample Clarity Parameters

Figure 6: Indri retrieval user interface (index selection phase)

Figure 7: Example of retrieval using Indri's retrieval user interface

Figure 8: Sample parameters used for retrieval in non GUI mode

Figure 9 (a): Average Gain

Figure 9(b): Max Gain

Figure 9(c): Original versus Gains

Figure 10: Sample data structures used

Figure 11: Microsoft's n-gram web service connection class

Figure 12: String Functions used

Figure 13: Code snippet showing original concept implementation

Figure 14: Code snippet showing our implementation

List of Tables

Table 1: TREC-Crowd11 Dataset Stats

Table 2: Sample Soap Request

Table 3: Sample Soap Response

Table 4: Lemur Features

Table 5: Indri Features

Table 6: IndriBuildIndex Parameters

Table 7: Query Clarity with retrieval parameters

Table 8: Results

Introduction

Year over year, a growing number of users are opting for long queries over one and two word search queries [23]. Commercial keyword based search engines, like Google, perform worse with long queries than short ones [1]. Long queries are usually expressed using natural language text, instead of keywords [1]. Due to this limitation on query length, significant improvements in search query performance can be achieved by reducing the length of the query.

While the utilization of single word queries has dropped by 3% [8], queries of length five words or more have increased at a year over year rate of 10% [2]. In the past there have been many works trying to improve upon the original queries by either re-weighting or reducing the original query. The fundamental driving these approaches is that shorter queries perform better than longer ones.

In this report we propose a hybrid concept that builds upon an existing query reduction method. We re-create the query, by trying to capture what the original user generated query intended to. We achieve this by dropping terms that might be unnecessary, thus reducing the length of the query. Dropping a single correct term (a term that dilutes the search results instead of making a positive contribution) can vastly improve query performance [2].

As an example consider the query “My friend would like to know the distance between the Earth and the Sun” Dropping the words “My friend would like to know the” and leaving “distance between the Earth and the Sun” would improve the performance of this query.

Finding the correct terms to be dropped is the challenge. Consider a query of length n . An existing approach considers all n sub-queries of length $n-1$ [2]. This method can yield significant gains. But due to the limited pool of sub-queries (of length $n-1$), performance gains are limited. The performance can be vastly improved by increasing this sample space of sub-queries. But due to the exponential number of sub-queries that could be selected (2^n-1 combinations); it becomes impractical to consider all, especially for web search where retrieval time is as critical as the retrieval quality.

Hence we look at ways to optimize sub-query consideration, while still maintaining linear time complexity. We propose a hybrid model that considers not only all sub-queries of length $n-1$ but also more. We first try to select the best possible sub-queries of lengths 1 to 5 using n -grams. For the remaining (from lengths 6 to $n-2$) we randomly select a sub-query from each length category. Then finally we select all the possible $n-1$ combinations as well as the original query. Using this approach we find that our results on an average improve by about 4 times compared to the approach followed by Kumaran et al. [2]. Moreover, queries in which further improvements are not possible our approach returns results identical to the approach referenced above in [2]. Improvements are judged by the predicted quality of the sub-query selected, which would thus result in optimal search results.

Related Work

There are three main approaches used to improve the quality of search results by finding the optimal query based on the original query. They are query segmentation, query substitution and query reduction.

Query segmentation is a technique that segments queries into concepts, and thus search engines retrieve web documents based on the concepts but not tokens [24]. Mutual information based approach was used by Jones et al. to determine segment breaks between pairs of tokens [25]. Tan and Peng's unsupervised machine learning approach tried to discover the underlying concepts of a query based on a generative language model [26]. Since the key concepts are identified, this greatly improves the retrieval performance for long queries [1]. But since segmentation treats all query concepts equally, the focus on key concepts is lost thus degrading long query effectiveness [1].

Query substitution is the replacement of long queries by short relevant keyword based ones [1]. Although this improves the retrieval performance of long search queries, diverse results as well as neighboring information may be obtained as it may ignore contexts from the original long query [1]. Yan Chen et al. [1] proposed the substitution-search result refinement algorithm that would filter non-relevant results, by evaluating the similarities of contexts from the results obtained and the results from the original query. However, this method is not ad-hoc query friendly.

Query reduction is a technique that eliminates noisy and redundant terms from long queries [1]. This is done by extracting key concepts using underlying retrieval models [1]. Carvalho, et al. [2] approached the query reduction problem by considering the effectiveness of a ranking function that scores documents with respect to a query so as to optimize a target measure. Such a measure is an estimate since it cannot be completely specified for every possible query. They suggested performance predictors such as Clarity [7] or Query Scope [10] to obtain the estimates for this target measure. Since the number of reduced queries that need to be evaluated is exponential, it is not feasible to evaluate all the possible combinations, especially in a web environment setting (for search). Hence, query reduction is carried out based on a reduced set of sub-queries. Considering the original query had n words, they only consider n reduced versions, plus the original query. As stated earlier, this approach yields dramatic performance improvements in certain cases [2].

Kumaran et al. observed that on an average the reduced versions were less effective than the original queries' effectiveness. Also, the maximum gains that could be achieved, considering the best possible reduced version of the query is selected, were positive. And lastly, if the original query has poor performance, the reduced versions were more likely to be better than the original query. Conversely, it was difficult to find reduced versions of queries that had high performing original forms. We pursue improvement in the query reduction approach as described by Kumaran et al. [2].

Technologies and Projects used

The Clue Web 09 dataset

The Clue Web 09 dataset was created to support the research on information retrieval and related human language technologies and consists of about a billion web pages in ten languages [3].

The dataset is used by several tracks of the TREC conference [14]. The subset used for this experiment is the TREC 2011 Crowd sourcing Track (TREC-Crowd11). This track contains pages from the TREC 2010 Relevance Feedback, pooled documents submitted by RF participants, TREC 2009 Relevance Feedback and Web Million Query Track [14].

<i>Unique topics</i>	217 topics
<i>Topic-docno pairs</i>	19829
<i>Unique topic-docno pairs</i>	19636
<i>Images present</i>	Jpg: 18512
<i>Images missing</i>	Jpg: 1124
<i>Pdf files present</i>	17243
<i>Pdf files missing</i>	2393
<i>Plain text files present</i>	19636
<i>Unique wget'd pages</i>	19636

Table 1: TREC-Crowd11 Dataset Stats

Source: TREC-Crowd11 Readme file [14]

From Table 1 we see that this dataset has 217 unique topics, which result in about 19636 unique topic document pairs. This gives us a large enough dataset to experiment with. In our search we only index the html files ignoring images, plain text, pdf and other files. We do this as we are only interested in indexing the text between specific tags like body, title etc. This way we can get enough data to build an index as well as filter out information that may not be very relevant. The complete dataset is about 19 GB in size. The datasets are distributed by Carnegie Mellon University for research purposes only [3]. The ClueWeb09-T11 (TREC-2011 Crowdsourcing dataset is available free of charge as a web download only [3].

REST Services

REST or Representational state transfer is an architectural style, based on the existing design of HTTP/1.0 [15]. It consists of clients and servers. The clients initiate their requests and the servers process these requests, giving appropriate responses in return [15]. Information transferred is a representation of a resource which is essentially a document that captures the current or intended state of a resource [15]. It relies on a stateless client-server cacheable communications protocol and in most cases that protocol is HTTP [16].

REST, though initially described in the context of HTTP, is not limited to it. RESTful applications maximize the use of the pre-existing, well-defined interface and other built-in capabilities provided by the chosen network protocol and minimize the addition of new application-specific features on top of it [15]. As an example, the World Wide Web can be viewed as a REST-based architecture [15].

REST is a lightweight alternative to mechanisms like RPC (Remote Procedure Calls) and Web Services (SOAP, WSDL, etc.) [16]. REST is also fully featured. It encompasses all the capabilities of other web based service architectures. REST when used over HTTP, simplifies communication between machines when compared to other complex mechanisms like CORBA, SOAP, etc. [16].

REST services are platform-independent, as well as language-independent. REST offers no built-in security features, encryption, session management, QoS guarantees, etc. but these can be added by building on top of HTTP [16]. For example, for encryption, the REST can be used on top of HTTPS. Consider the following example to understand the difference between REST and Web Services /SOAP. The SOAP request would look like:

```
<? xml version="1.0"?>
<soap: Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:body pb="http://www.acme.com/phonebook">
  <pb:GetUserDetails>
    <pb:UserID>12345</pb:UserID>
  </pb:GetUserDetails>
</soap:Body>
</soap: Envelope>
The REST request would look like :
http://www.acme.com/phonebook/UserDetails/12345
```

Table 2: Sample Soap Request

Source: <http://rest.elkstein.org/> [16]

A server response in REST is often an XML file. For example consider:

```
<parts-list>
<part id="3322">
  <name>ACME Boomerang</name>
  <desc>
    Used by Coyote in <i>Zoom at the Top</i>, 1962
  </desc>
  <price currency="usd" quantity="1">17.32</price>
  <uri>http://www.acme.com/parts/3322</uri>
</part>
<part id="783">
  <name>ACME Dehydrated Boulders</name>
  <desc>
    Used by Coyote in <i>Scrambled Aches</i>, 1957
  </desc>
  <price currency="usd" quantity="pack">19.95</price>
  <uri>http://www.acme.com/parts/783</uri>
</part>
</parts-list>
```

Table 3: Sample Soap Response

Source: <http://rest.elkstein.org/> [16]

Other response formats like CSV, JSON (Java Script Object Notation) and plain text can also be used.

Microsoft Web N-Gram Service (Public Beta) N-grams data

An n-gram is a contiguous sequence of n-terms from a given sequence of text or speech [27]. An n-gram of length 1 is called a unigram, of size 2 a bigram and of size 3 a trigram. N-grams of lengths 4 or more are called as four-grams, five-grams and so on. They can be used to predict the next item in a sequence based on statistics collected from the text corpus [27].

We use Microsoft's n-gram service to predict the performance of sub-queries of lengths 1 to 5. For each sub-query up to length 5 terms, we look up the joint probabilities of the set of words contained in the sub-query. Using this score (joint probability) we select the reduced query with the highest score from each length segment.

This service provides access to petabytes of data via public beta web n-gram Services [11]. These services are hosted on a cloud based platform, highly useful in areas related to language processing, speech and web- search [11]. This service provides access to specific content types like the document body, title and anchor texts and supports smoothed models [11]. The available n-grams are unigram, bigram, trigram, and n-grams with N=4, 5. The Bing en-us market is used to index the documents [11]. These services are hosted and updated by Microsoft. A user token is needed to access these services. Microsoft Research issues this token.

These services can be invoked via SOAP or REST requests. For example a GET call on <http://web-ngram.research.microsoft.com/rest/lookup.svc/> would return a list of supported models in path-form which can then be plugged into the various lookup methods. The general format is `http://web-ngram.research.microsoft.com/REST/lookup.svc/{catalog}/{version}/{order}/ {operation}? {parameters}`

The catalog determines the dataset to be queried, like the Bing-Body. The version identifier determines the version of the dataset to be used. Jun09 is an example of a version. Order states the order of the n-grams from one to five to be queried. The operation specifies the type of probability to return. The choices for operation are conditional and joint probabilities. Other parameters include the user token which uniquely identifies the user accessing the web service. This token is generated and distributed by Microsoft Research. P is the phrase to be queried. The format of the result returned can be specified as well. These could be JSON, text or xml. When no format is specified text is assumed.

The Lemur Project

The Lemur Project, best known for its Indri search engine, Lemur Toolbar, and ClueWeb09 dataset, develops tools to support research and development of information retrieval as well as text mining software [17]. Some of their products include search engines, browser toolbars, text analysis tools, and data resources [17].

Their software development is based on the pillars of state-of-the-art accuracy, flexibility, and efficiency [17]. For example Indri search engine provides search solutions as is and also stores data in a manner accessible to support further development in the field of information retrieval [17].

The Lemur Project was begun by the Center for Intelligent Information Retrieval (CIIR) at the University of Massachusetts, Amherst, and the Language Technologies Institute (LTI) at Carnegie Mellon University [17].

The Lemur Toolkit is designed to facilitate research in language modeling and information retrieval (IR), where IR is broadly interpreted to include such technologies as ad hoc and distributed retrieval with structured queries, cross-language IR, summarization, filtering and categorization [5]. The system's underlying architecture was built to support the technologies above [5].

- Sophisticated structured query languages (using InQuery and Indri)
- Support for XML and structured document retrieval
- Used commonly with a wide range of research test collections (e.g., TREC CDs 1-5, wt10g, RCV1, gov, gov2)
- Index your web pages with an "out-of-the-box" site search capability
- Interactive interfaces for Windows, Linux, and Web
- Distributed information retrieval and document clustering applications
- Cross-platform, fast and modular code written in C++
- C++, Java and C# APIs
- Free and open-source software
- In use for over 6 years by a large and growing user community
- Indexing
- Multiple indexing methods for small, medium and large-scale (terabyte) collections
- Built-in support for English, Chinese and Arabic text
- Porter and Krovetz word stemming
- Incremental indexing
- Out-of-the-box indexing support for TREC Text, TREC Web, plain text, HTML, XML, PDF, MBox, Microsoft Word, and Microsoft PowerPoint
- Indexes inline and offset text annotations (e.g., part-of-speech and named entities)
- Indexes document attributes
- Retrieval
- Supports major language modeling approaches such as Indri and KL-divergence, as well as vector space, tf.idf, Okapi and InQuery
- Relevance- and pseudo-relevance feedback
- Wildcard term expansion (using Indri)
- Passage and XML element retrieval
- Cross-lingual retrieval
- Smoothing via Dirichlet priors and Markov chains
- Supports arbitrary document priors (e.g., Page Rank, URL depth)

Table 4: Lemur Features

Source: <http://lemurproject.org/lemur.php> [13]

Indri Project

Indri is a component of the Lemur Project. It is a text search engine, developed at UMass [18]. It is freely available with a flexible BSD-inspired license [18]. The Indri search engine provides accurate search for large text collections ‘out of the box’ [17]. It also stores the data in an accessible manner to support development of new retrieval strategies [17].

- Powerful Query Interface
 - Supports popular structured query operators from INQUERY
 - Suffix-based wildcard term matching
 - Field retrieval
 - Passage retrieval
- Flexible Indexing and Document Support
 - Supports UTF-8 encoded text
 - Language independent tokenization of UTF-8 encoded documents.
 - Parses PDF, HTML, XML, and TREC documents
 - Word and PowerPoint parsing (Windows only)
 - Text Annotations
 - Document Metadata
- Package Versatility
 - Open source, with a flexible BSD-inspired license
 - Includes both command line tools and a Java user interface
 - API can be used from Java, PHP, or C++
 - Works on Windows, Linux, Solaris and Mac OS X
- Scalability and Efficiency
 - Best-in-class ad hoc retrieval performance
 - Can be used on a cluster of machines for faster indexing and retrieval
 - Scales to terabyte-sized collections

Table 5: Indri Features

Source: <http://www.lemurproject.org/indri.php> [4]

Indri is built up of many sub applications.

IndriBuildIndex:

This application can build Indri repositories from various data sources [18]. The data sources include TREC formatted documents, HTML documents, text documents, and PDF files [18]. On Windows it has the added capability of indexing Word and PowerPoint documents [18]. The IndriBuildIndex understands tags as well (like <head></head> in HTML documents) and hence can index by tags as well [18].

IndriRunQuery:

This application evaluates queries and returns a ranked list of documents [18]. These queries are evaluated against one or more Indri repositories [18]. For passage retrieval queries, Indri can be instructed to print the document text as well [18].

IndriDaemon:

This application is a repository server. It awaits connections from the IndriRunQuery instances and processes queries that come as network requests [18]. An instance of IndriRunQuery can connect to multiple IndriDaemon instances concurrently [18]. This makes retrieval using a cluster of machines possible [18].

Indri Build Index

This application builds the index for a collection of documents to be used by other applications.

<i>index</i>	Name of the index table-of-content file without the extension. Use full path information here to use index later from other directories. i.e. /lemur/indexes/myindex
<i>indexType</i>	The type of the index you want to build. <i>key</i> for KeyfileInclIndex (.key) <i>indri</i> for IndriIndex (.ind)
<i>memory</i>	Memory (in bytes) to pre-allocate (def = 96000000)
<i>Stopwords</i>	Name of file containing the stopword list.
<i>Acronyms</i>	Name of file containing the acronym list, currently not supported by IndriIndex. These acronyms will still be indexed in lowercase by IndriIndex.
<i>countStopWords</i>	If true, count stopwords in document length.
<i>docFormat</i>	<ul style="list-style-type: none"> • <i>TREC</i> for standard TREC formatted documents. • <i>web</i> for web TREC formatted documents. • <i>Chinese</i> for segmented Chinese text (TREC format, GB encoding) . • <i>chinesechar</i> for unsegmented Chinese text (TREC format, GB encoding) . • <i>arabic</i> for Arabic text (TREC format, Windows CP1256 encoding) .
<i>Stemmer</i>	<ul style="list-style-type: none"> • <i>porter</i>: Porter stemmer. • <i>Krovetz</i>: Krovetz stemmer. • <i>Arabic</i>: arabic stemmer, requires additional parameters. • <i>arabicStemDir</i>: Path to directory of data files used by the Arabic stemmers. • <i>arabicStemFunc</i>: Which stemming algorithm to apply, one of: <ul style="list-style-type: none"> • <i>arabic_stop</i> : arabic_stop. • <i>arabic_norm2</i> : table normalization. • <i>arabic_norm2_stop</i> : table normalization with stopping. • <i>arabic_light10</i> : light9 plus ll prefix. • <i>arabic_light10_stop</i> : light10 and remove stop words.
<i>dataFiles</i>	Name of file containing list of data files to index.

Table 6: IndriBuildIndex Parameters

Source: <http://www.lemurproject.org/lemur/indexing.php> [12]

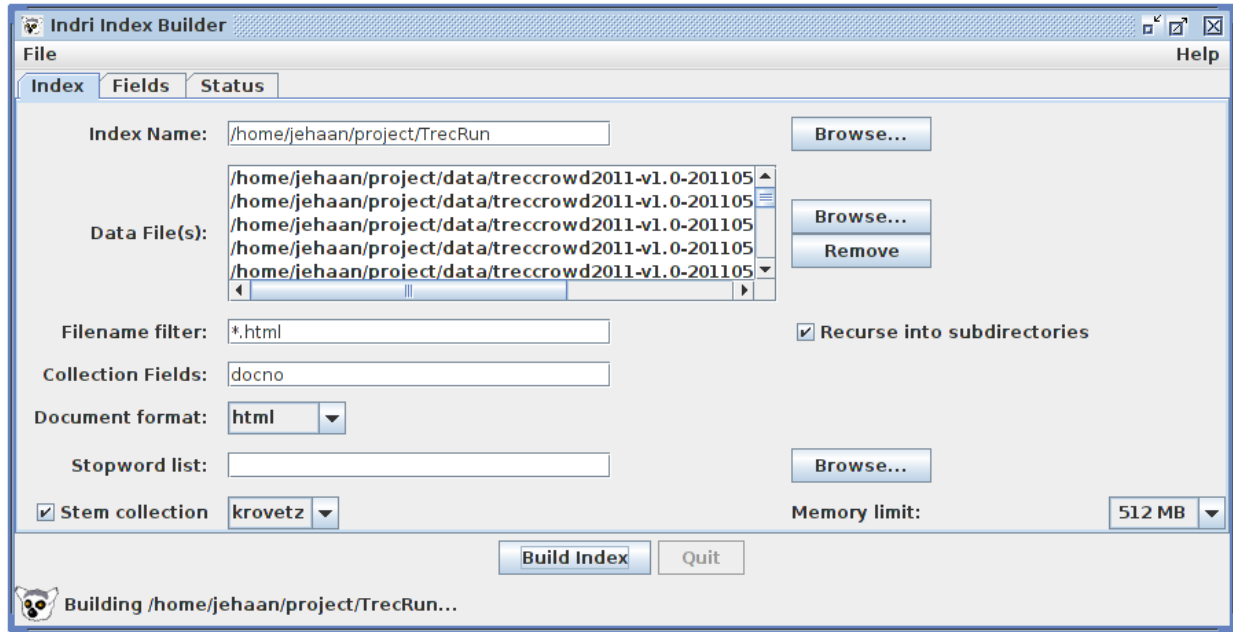


Figure 1: Indri Index build setup

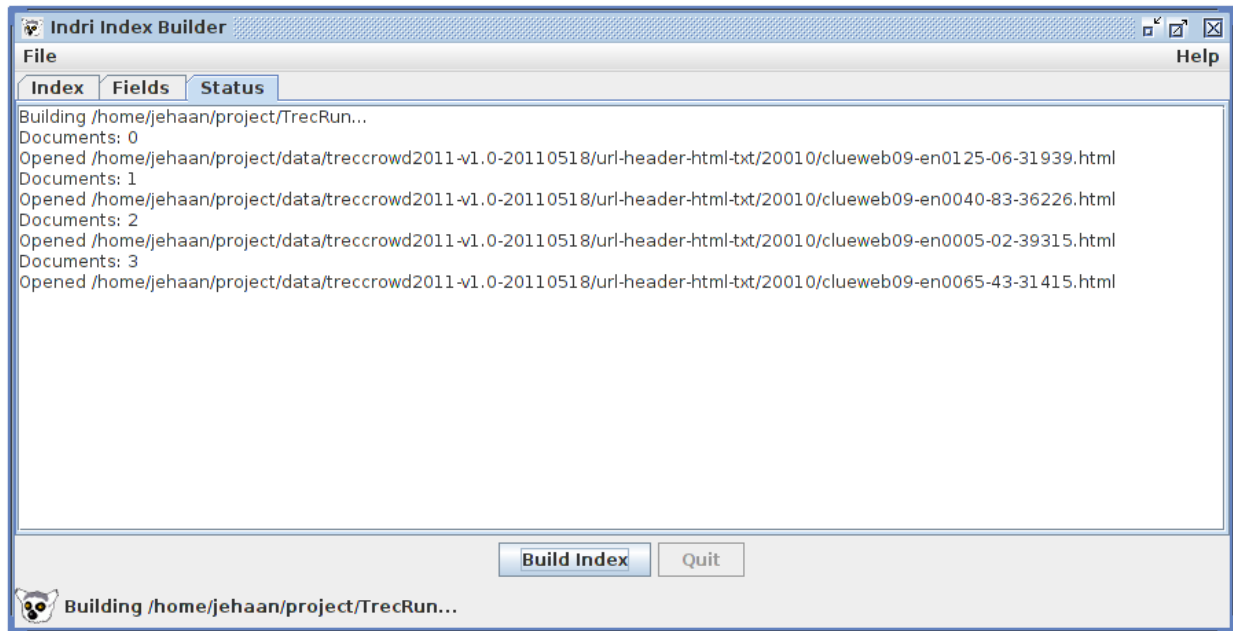


Figure 2: Indri indexing in progress

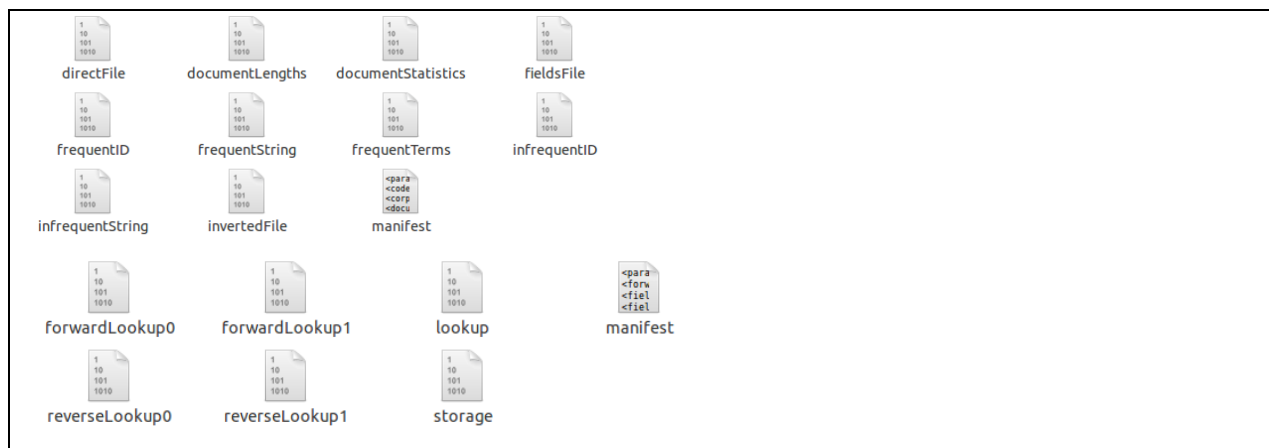


Figure 3: Indices created by Indri for search and other retrieval-based applications.

```

<parameters>
<memory>2560M</memory>
<storeDocs>>false</storeDocs>
<index>ClueWeb09_English_1</index>
  <corpus>
    <path>/home/jehaan/project/demo/ClueWeb09_English_Sample.warc</path>
    <class>warc</class>
  </corpus>
  <field><name>title</name></field>
  <field><name>heading</name></field>
  <stemmer><name>krovetz</name></stemmer>
</parameters>

```

Figure 4: Sample Indri indexing parameters

Query Clarity with retrieval

Clarity scores measure the ambiguity of a query with respect to the collection of documents and show that they correlate positively with average precision in a variety of TREC test sets [20]. Query Clarity with retrieval computes clarity scores for an expanded query model [6]. The calculation is based on pseudo-feedback documents [6]. Clarity scores are calculated for the entire query as well as each individual term within the query [6].

<i>Index</i>	The complete name of the index table-of-content file for the database index.
<i>smoothSupportFile</i>	The name of the smoothing support file (e.g., one generated by <code>GenerateSmoothSupport</code>).
<i>textQuery</i>	The original query text stream.
<i>expandedQuery</i>	The file to store the query clarity scores.
<i>feedbackDocCount</i>	The number of docs to use for pseudo-feedback. If not specified or 0, the value defaults to 500.
<i>queryUpdateMethod</i>	Feedback method, one of: <ul style="list-style-type: none"> • <code>mixture</code> or <code>mix</code> or 0 for mixture. • <code>divmin</code> or <code>div</code> or 1 for div min. • <code>markovchain</code> or <code>mc</code> or 2 for markov chain. • <code>relevancemodel1</code> or <code>rm1</code> or 3 for relevance model 1. • <code>relevancemodel2</code> or <code>rm2</code> or 4 for relevance model 2.
<i>For all interpolation-based approaches</i>	
<i>feedbackCoefficient</i>	The coefficient of the feedback model for interpolation. The value is in [0,1], with 0 meaning using only the original model (thus no updating/feedback) and 1 meaning using only the feedback model (thus ignoring the original model).
<i>feedbackTermCount</i>	Truncate the feedback model to no more than a given number of words/terms.
<i>feedbackProbThresh</i>	Truncate the feedback model to include only words with a probability higher than this threshold. Default value: 0.001.
<i>feedbackProbSumThresh</i>	Truncate the feedback model until the sum of the probability of the included words reaches this threshold. Default value: 1.
<i>feedbackMixtureNoise</i>	<ul style="list-style-type: none"> • For the collection mixture model method, <code>feedbackMixtureNoise</code> is the collection model selection probability in the mixture model. That is, with this probability, a word is picked according to the collection language model, when a feedback document is "generated". • For the divergence minimization method, <code>feedbackMixtureNoise</code> means the weight of the divergence from the collection language model. (The higher it is, the farther the estimated model is from the collection model). • For the Markov chain method, <code>feedbackMixtureNoise</code> is the

	probability of <i>not</i> stopping, i.e., $1 - \alpha$, where α is the stopping probability while walking through the chain.
emIterations	Maximum number of iterations the EM algorithm will run. Default: 50.

Table 7: Query Clarity with retrieval parameters

Source: <http://www.lemurproject.org/doxygen/lemur/html/RetQueryClarity.html> [6]

```

<parameters>
<index>ClueWeb09_English_1</index>
  <corpus>
    <path>/home/jehaan/project/demo/ClueWeb09_English_Sample.warc</path>
    <class>warc</class>
  </corpus>
<textQuery>face</textQuery>
<expandedQuery>/home/jehaan/project/demo/expandedQuery</expandedQuery>
<feedbackDocCount>500</feedbackDocCount>
<queryUpdateMethod>0</queryUpdateMethod>
<feedbackCoefficient>0</feedbackCoefficient>
<feedbackTermCount>30</feedbackTermCount>
<feedbackProbThresh>0.001</feedbackProbThresh>
<feedbackProbSumThresh>1</feedbackProbSumThresh>
<feedbackMixtureNoise>0.5</feedbackMixtureNoise>
</parameters>

```

Figure 5: Sample Clarity Parameters

Retrieval User interface (RetUI)

RetUI is a Graphical user interface based Indri retrieval application. Once a connection to the index or index server is established, a query can be entered in the system following which a search can be performed. The number of documents returned can be pre-set. The Database(s) list shows all open indexes and index servers. Indexes can be easily added or removed via the file menu.

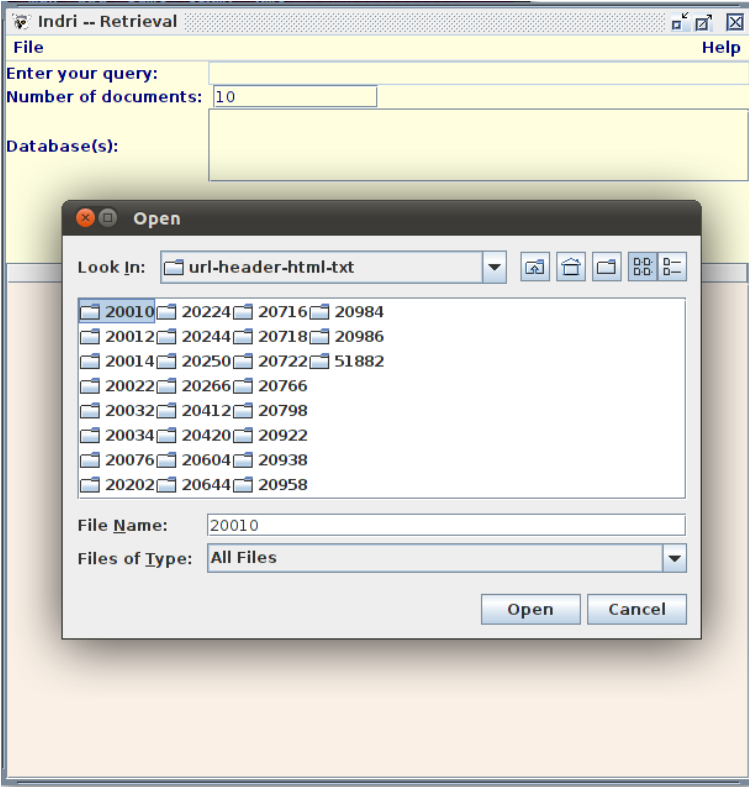


Figure 6: Indri Retrieval User Interface (index selection phase)

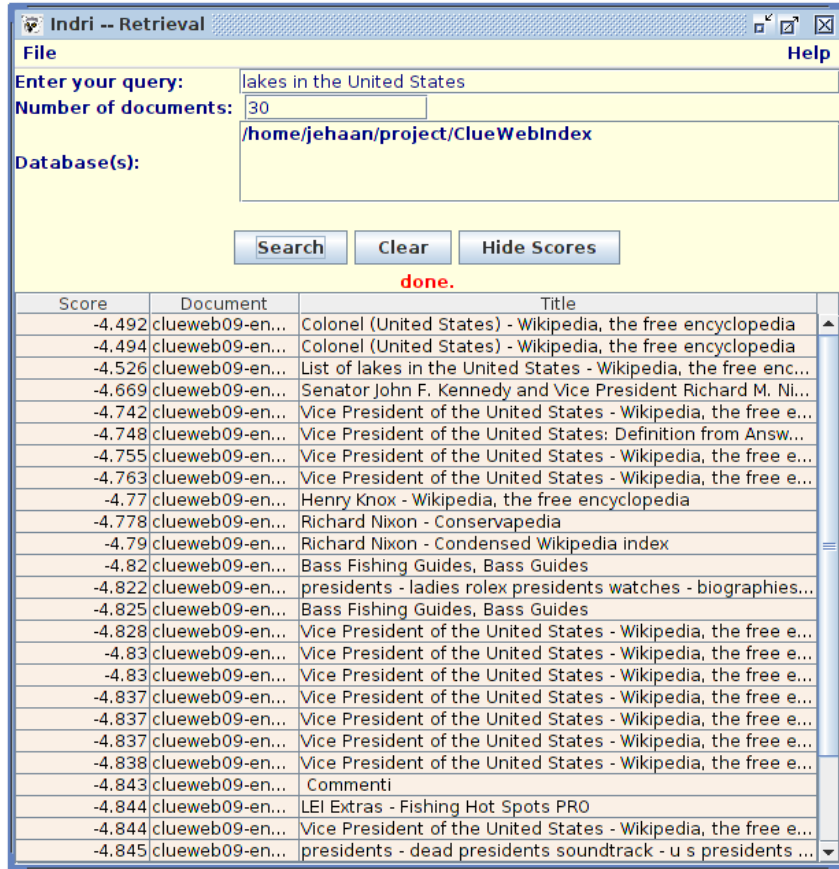


Figure 7: Example of retrieval using Indri's Retrieval User interface

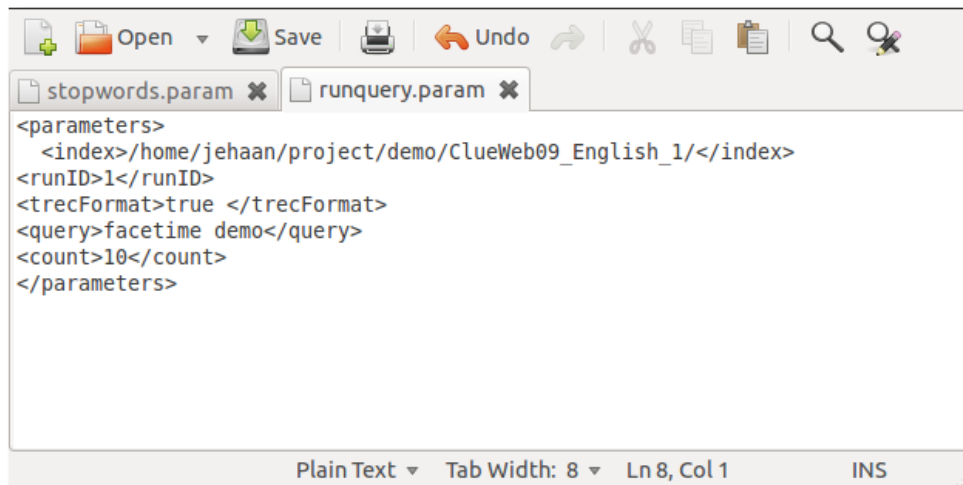


Figure 8: Sample parameters used for retrieval in non GUI mode.

The Experiment

Technique

Query reduction is one of the many approaches that can be used to optimize the search performance of a query. As established earlier, the search retrieval performance is inversely proportional to the length of the query. The longer the query the more specific it gets, and hence the number of results returned by the search engine is reduced.

Query reduction – the technique of automatically identifying and removing extraneous terms from the long queries- has proved to be an effective technique for improving performance on long queries [9].

The Original Approach

The authors Kumaran et al [2] approach reduction of long queries by dropping unnecessary terms and hence improving performance of ad-hoc retrieval on TREC collections.

They proposed three learning formulations that combine query performance predictors to perform automatic query reduction [2]. These formulations allow easy integration into the search engines architecture for rank-time query reduction [2]. Their approach yields an approximate improvement of more than 12% in NDCG@5 in the impacted set of queries and hence significantly outperforms the original query [2]. This method delivers consistent retrieval gains in original queries that perform poorly [2]. They approach reduction by dropping a single term at a time [2]. Their studies show that just dropping a single and correct term from the original long query can result in a 26% improvement in NDCG@5 [2].

They define the query reduction problem as:

Let $f: P \times D \rightarrow R$, denote a ranking function (**R**) that scores documents (**D**) with respect to a query (**P**), represented as a set of query terms. Let $T_f(P)$ denote a target measure of the effectiveness of the ranking produced by function **f** for the query **P** [2].

The problem is to find the reduced version of P^* such that the highest value for the target measure is achieved as $P^* = \arg \max T_f(P)$ where P is a subset of Q [2]. Since this cannot be completely inferred over all possible instances of sub queries, it is estimated [2]. Hence the task turns to maximizing the estimated target measure. Query performance predictors like Clarity [7] or Query Scope [19] can be used to estimate this target measure [2]. This would help select a near optimal reduced version P^* of the original query Q .

Efficiency is a key challenge for reduction of queries. This is due to the exponential number of possible sub queries to evaluate in order to yield the optimal sub set of query terms. This is critical especially for web engines where response times are as critical as the quality of results returned. To address this issue they present a simpler version of the problem. They consider reduced versions that only differ from the original query by one term. They selected n sub-queries of length $n-1$ [2]. In this way they limited their sample space and noticed improvements in search quality performance in some queries over the original query.

From their experiments they noticed the following:

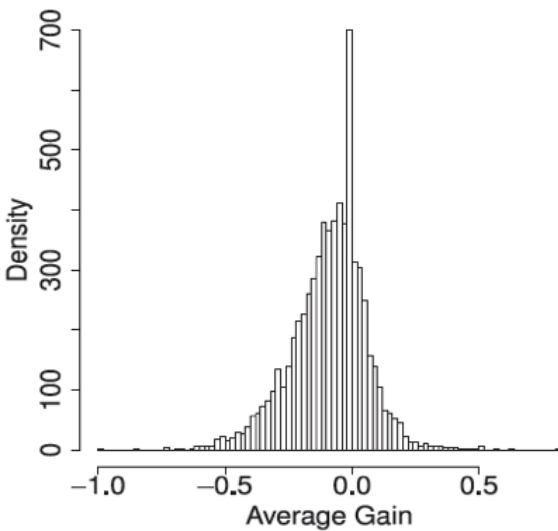


Figure 9 (a): Average Gain
Source : Kumaran et al [2]

Figure 9 (a) shows distribution of gain. It shows that on an average the reduced versions' effectiveness is worse than the original query's effectiveness [2]. In other words the original query outperforms the reduced versions on an average.

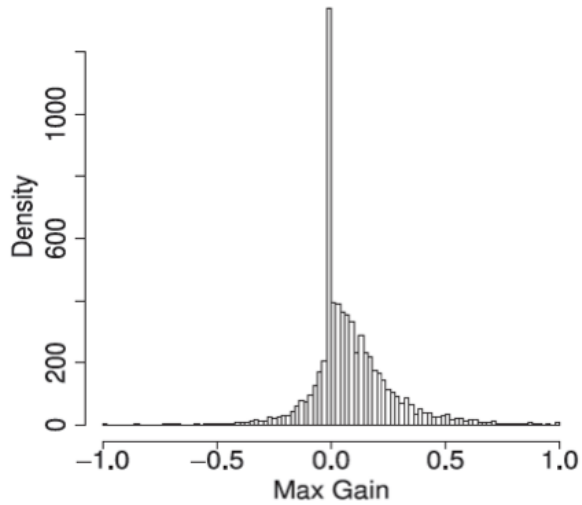


Figure 9 (b): Max Gain
Source: Kumaran et al [2]

Figure 9 (b): The Maximum gains that can be achieved if the best-reduced version is selected are mostly positive. Also for some queries the maximum gains are negative indicating that any reduction in the query will result in decreased performance.

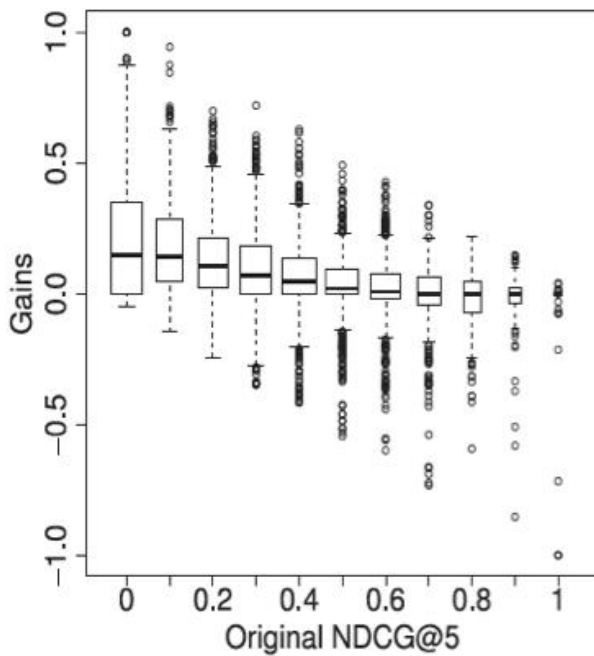


Figure 9 (c): Original versus Gains
Source: Kumaran et al [2]

Lastly they noticed that if the original query had poor performance the reduced versions are more likely to outperform the original [2]. Conversely it was hard to find a reduced version of a well performing original query that could provide substantial gains [2].

Based on these observations they developed learning formulations.

Independent Prediction:

The performance of the original long query and its reduced versions is predicted independently. The query with the highest performance is selected [2]. Thus the query selection problem is transformed into selecting the query with the highest predicted performance [2].

Difference Prediction:

Since independent prediction does not encode the relationship between the original query and its reduced versions, the difference in prediction between them needs to be considered to accurately predict the effectiveness of the individual queries [2]. Hence the difference in performance between the original long query and its reduced versions is predicted and the query with the highest performance is selected [2].

Ranking Queries:

In this formulation the original query and its reduced versions are ranked in order to select the top ranking query [2]. This is done by training on pair wise preferences between the queries [2].

Thresholding:

Thresholding limits the selection of a sub-query by specifying a certain minimal gain that has to be achieved in order to be shortlisted for final selection.

- In independent prediction, a reduced version is selected only if the reduced version outperforms the original query by a specified threshold [2].
- In difference prediction, the positive difference has to exceed a threshold in order for the reduced version to be selected [2].
- For Ranking, the predicted performance of the top ranking reduced version must exceed the original query's predicted performance by the threshold specified [2].

Our Approach

The approach as described by Kumaran et al has tradeoffs in terms of the number of queries affected versus the overall average gains achieved by query reduction. The naïve approximation to the full scale (exponential) query reduction problem substantially improves efficiency (exponential to linear), while still providing significant effectiveness gains [2]. In the improved average performance, they noticed high variance in the performance [2].

Hence we try to build upon their concepts, by increasing the pool of queries whose performance is to be predicted as well as keeping the number of queries to be evaluated linear. We understand that while the naïve approach would determine best results, is not feasible. But by considering more subsets of queries the performance of the above approach can be improved.

Hence our aim was to improve the performance of the above approach by building upon their model. Their baseline was the original query. Our baseline is their approach, and hence the improvements they achieved. This way we guarantee the minimum performance what they already achieved as well as improvements beyond, which in certain cases are very close to the ideal or best case.

We calculate the best case by considering all possible combinations of the given query and calculating the clarity scores for each one and ranking them by their scores. Then we take the weighted average of the top 5 queries from the ranked list.

To increase the sample space of subset of queries we broke the queries up into 3 parts. For the subset of queries with length one to five terms we used n-grams to evaluate and return only the top ranking queries from each length segment. Then we considered queries of length six to n-2, which we selected randomly. Lastly we selected all n possible sub queries of length n-1 and the original long query as described by Kumaran et al.

Then using this subset of queries we calculated the clarity scores for each query. This would serve as a score to understand to retrieval quality performance of the query. We then took the difference in clarity scores between these reduced versions and their original version. By ranking these scores we could compare the predicted performance of each query.

To obtain a metric for query performance, we considered the weighted average of the query clarity scores by multiplying each query's clarity score by the difference between its rank and the lowest ranked query and then took the sum of all these values. For this we only considered the top 5 ranked queries. Hence a single normalized metric was obtained to compare query document retrieval performance which takes the ranking of the queries into consideration.

Implementation

To start with the experiment we first loaded the data set. This was the TREC Crowdsourcing 2011 track. We used Indri search engines IndriBuildIndex Application to build the index. This could be done by either using the supplied GUI tool or using the command line. We used experimented with both approaches. Once the dataset was indexed we ran trial queries using the IndriRetUI GUI tool, to understand indexing performance and effect of the various parameters that can be set for indexing.

Once indexing was completed we ran Query Clarity on sample queries to understand how ambiguous and unambiguous queries performed. Clarity was used as a measure to compare and hence judge the performance of the queries generated. The original authors approach was replicated as accurately as possible.

After replication of the original method we tried to see the difference in performance by understanding the effect of n grams to select the optimal query. N-grams being indexed are quickly retrieved and hence the performance overhead should be near negligible and hence relatively computationally inexpensive.

Since the first five terms are selected using n-grams, the remaining sub queries are selected randomly from length 6 to n-2. Then using the authors approach all the queries of length n-1 and n are selected.

We calculated the clarity score for each of the chosen sub queries and then ranked these queries by their score. These tests were run about a 1000 times to understand the average performance of random selection of sub queries.

Benchmarking and results

We randomly selected 100 queries from the dataset that was indexed to benchmark the different approaches. The authors approach scores at best a significant improvement over the original query and worse case the same as the original query [2]. Our approach uses the authors approach as the baseline and has a few scores closer to the ideal case. The ideal case scores as mentioned earlier are calculated by ranking all the possible reduced versions in order to select the top 5 sub-queries for which the weighted average of the Clarity scores would be calculated. The results of our benchmarking tests are :

Sr. No.	Query	Authors Approach	Our Approach	Best Case
1	Professional web Hosting Service Provider HSP and Corporate IT professionals	6.07143	21.51782	33.73132
2	find tons of cheap international travel airlines and they can be found all over the place	6.05855	6.43822	31.3861
3	airframe that became The Red Baron	5.56061	9.42546	19.59863
4	searchable in a variety of ways from price to product type	4.635595	8.485841	35.71394
5	The Internet Definitive Buyers Services Guide	7.61017	7.61017	35.40384
6	The best choice of cheap downloadable OEM software is offered	11.56594	15.53296	40.94454
7	Finding the Best T1 Service Provider in Your Area	2.27936	2.93455	16.78798
8	wedding entertainment professionals who have entertained thousands of couples	-0.25852	-0.10314	22.7893
9	DJ Spinelli Assoc is a professional Disc Jockey	2.41148	2.42157	7.51011
10	planning your wedding is fun and easy	3.200196	20.45951	32.99578
11	The MinuteMan site has been online since 2002	2.50426	4.03769	55.63561
12	the NJ Environmental Digital Library Census Bureau online mapping	1.40685	4.80456	27.37782
13	Major League Baseball selects the Adobe Flash Platform	5.93864	26.4392	46.98035

14	save an incredible amount of time and effort	0.198724	12.571999	30.72971
15	Consolidate data from two or more data sources into a data warehouse	0.17384	2.37894	27.158895
16	Flash Player bug and issue management system is now available for use by external users	2.11436	24.28304	48.87496
17	protects you from hackers phishing and other online fraud	0.00122	0.00243	1.2941
18	do not have the correct Flash Player installed	3.95951	14.88498	27.35322
19	If you use the Internet Explorer browser or AOL you need	4.278315	22.032245	48.464915
20	OEMs to differentiate their handsets and devices	2.74477	22.7155	29.72684
21	runtime lets developers use proven web technologies	-0.43894	12.25017	25.60872
22	only for purposes of achieving the distribution described	1.125018	4.632321	11.57964
23	Inventions links of learned franklin philosopher American	2.88241	8.91441	22.61704
24	barber shop carson daly ben harper benchmarking ben jerry	0.46696	0.91797	3.55108
25	gained the recognition of scientists and intellectuals across Europe	0.035952	0.695969	17.22355
26	worried about all the moving arrangements	1.99168	2.368	10.8209
27	Select from 165 Ben Franklin items available to buy	1.77397	20.24482	30.62086
28	Ben Franklins Wit and Wisdom	2.48495	5.62292	11.83825
29	Highway 6 at the Lake Murray Dam in Irmo	1.88247	5.34145	30.65717
30	Glass containers are not allowed in the park	1.641058	13.104961	34.973295
31	I sell real estate in the Columbia area	0.30193	2.19788	32.84217

32	Looking for the perfect gift to spark the interest	0.504753	1.003977	21.20402
33	chairman of the Falmouth School Board	0.142199	7.719973	21.002735
34	Trout fishing is somewhat sporadic however and actually	1.01262	9.87368	41.90773
35	Build a mini fire extinguisher and float a soap	0.387849	12.799405	22.312555
36	suggest the rhythm played at the time rivaled the tempo	0.712627	9.004548	20.722311
37	Professor Probenius is your chemistry professor for CHEM	1.67967	1.67967	11.77977
38	current operating schedules and announcements visit the COSD Water Dept	4.38881	7.477932	22.598615
39	I encourage all believers to give up the shackles of faith	1.387086	7.032334	53.85263
40	The smallest particle of light is a photon	1.224286	6.118562	18.550834
41	The association uses donations to support arts	5.394981	11.217877	33.238945
42	women who are in love can recognize their partner	-0.003938	14.590807	39.999595
43	The latest release of the Virtual Earth	0.07193	0.07717	30.08633
44	The Daily Mail is encouraging its readers to buy the traditional non	0.79785	3.471969	50.45134
45	customers to search for more types of mapping information	1.865478	17.80624	38.39073
46	mashups with an intuitive JavaScript programming model	0.38099	3.67857	38.50879
47	imagery enhances our currently available data by seeing	0.269676	2.956008	4.200583
48	UK government have signed up to an EU decision	5.987756	46.01875	53.87314
49	natively be a premium content layer	2.567362	6.509855	31.614445

50	MSDN technical article posted online showing users how to authenticate	-0.33698	24.83715	33.42506
51	see all the damage that has been done	0.310678	3.310625	8.501512
52	the only weather application that offers looping radar	4.204547	7.521331	22.973245
53	the drug is intended to help people with a rare hereditary	2.097099	5.127585	44.851695
54	Balance Board to talk to the program after decoding the Bluetooth	2.19096	11.455997	45.61618
55	sexy applications that push the limits of geospatial and Virtual Earth	0.67226	0.67226	62.62344
56	New Orleans area to show your insurance adjuster	7.05535	22.06113	30.12247
57	for its athletic programs as well as its band department	1.017211	11.951999	47.181005
58	East Ridge has gone to State Competition for Concert Band	1.8574	17.7914	38.06548
59	online mapping service that enables users to search	1.657023	15.710907	40.44724
60	Student enrolment at East Ridge High School is currently	0.419911	12.063247	23.629185
61	Microsoft provides a staging environment to test your application	3.03708	21.08689	31.19509
62	known for its athletic programs as well as its band department	-0.040917	8.094521	46.00286
63	A new director has be hired to oversee the percussion section	1.29421	4.524457	48.35589
64	Lowest prices cheap prescription diet pills	6.03234	8.37805	21.83186
65	not meant to substitute for the advice provided by your own physician	0.057815	1.965707	32.409165
66	Posted in Prescription online phentermine no prescription	2.93088	5.21027	14.57121

67	things running for fans around the country	0.485916	0.531482	7.392562
68	A statue to her memory stands in Slater Park	2.63427	15.85733	30.11823
69	derrick car at the Clinchfield Railroad yard	1.14479	3.17069	17.24397
70	initial startup never had anything to do with the military	1.332563	4.333282	25.910011
71	real estate virtual tour software service	1.46169	2.36188	16.55054
72	interfering with the absorption of certain nutrients in consumed food	0.675537	0.684862	27.51389
73	includes a list of ships with the same or similar names	0.646984	14.626586	45.89544
74	us presidents born in Massachusetts	0.54408	5.02678	6.93027
75	magic the gathering alpha black vice	0.46453	2.91722	18.65372
76	la times vice president public affairs	2.22008	5.1986	11.16651
77	evaluating a university vice presidential candidate	0.96957	1.19297	16.11441
78	English speaking nations largely followed either	7.60005	15.89927	25.50411
79	The fact that my two bikes are still going strong	0.081327	0.814137	6.787198
80	hybrid electric vehicle manufactured by Honda	2.84995	3.71631	31.71426
81	The raw data for Manhattan is aggregated from the NYC	9.951521	12.728404	36.779025
82	Nixon was sick on the first televised debate	0.854173	5.859417	27.20089
83	transmission up and down arrows suggest when to shift gears	4.294902	34.424209	48.634045
84	Diet pills aim to help overweight people to curb their hunger	6.08813	6.33453	26.56056
85	report to Employment and Immigration Minister Hector Goudreau	2.101119	20.6542	25.93064

86	significant deceleration when used in regenerative mode for braking	1.307598	17.22306	23.66421
87	vice president of arizona employers council	3.02186	4.0142	8.72438
88	lightweight aluminum structure to maximize fuel efficiency and minimize	1.55047	11.67285	18.59682
89	history of president franklin roosevelt	6.56802	7.03426	14.77119
90	The story goes that the military version could go	2.122865	4.312422	21.571245
91	has more than doubled in the last five years	1.346813	9.241467	23.44662
92	original factory new or used parts and manufacture parts	0.3509	9.58925	16.51057
93	he benefit may be modest and the side effects intolerable	1.897887	13.892916	28.397165
94	left so they sold them all to COMB liquidation	0.145526	15.624767	26.404715
95	superb Naomi Campbell figure is all lined	1.493825	13.976042	24.044755
96	appointment with your doctor will serve this purpose	3.584	4.918245	26.21529
97	Cathine is found in shrub Catha edulis	7.45465	19.17349	29.10425
98	may not reflect the actual production season	1.7464	2.19081	6.06296
99	five closing themes in the Japanese episodes	3.959086	11.247737	40.01236
100	Certain pills now under research and development	0.26553	0.26553	21.76097
	Totals	233.048222	956.674474	2780.776101

Table 8: Results

From the above results table we see that on an average our method scores about 4 times better than the original approach. Also worst-case performance is the same as the Author’s approach [2]. In many instances we can see that our approach’s score is closer to the best case than our baseline [2]. This is because we consider a greater sample space when compared to just the n-1 approach [2].

Conclusion

In conclusion we would like to state that there is a vast scope for improvement in performance. Until evaluations of all possible combinations are a feasible option, using predictors to do the same is currently a good approach. This way, without extensive computation, the performance of a query can be predicted. The prediction is only as good as its sample space. Hence keeping the sample space linear is a trade off that dictates query performance (quality) vs. efficiency. Variations in query performance indicate that we still lack predictors that can give consistent improvements in search results. Besides that due to the closed nature of commercial search engines any sort of integration is built on an abstract layer and is loosely coupled which reduces the optimizations possible with tighter integration.

Using n-grams to find out the optimal performing sub queries is still feasible as it is limited to queries of length 5. Since n-grams are stored using directory structures their pre-computed joint and combined probabilities could be referred in sub-linear to linear time.

Introducing Random selection to select subset of queries from length 6 to $n-2$ is an inexpensive way to increase the sample space of sub-queries while leaving the possible options linear. Over time it also averages out to an approximately constant end result while still leaving scope for improvement. This is done without replacing the query in the query pool.

We used clarity score to understand the performance of the various methods. Clarity scores measure the ambiguity of a query with respect to the collection of documents and show that they correlate positively with average precision in a variety of TREC test sets [20]. In other words clarity scores can assist could be used to identify the performance of a query without relevance information [20].

Hence we conclude that while we have found evidence of improved performance over the baseline (original author's approach [2]), better prediction methods could yield further improvements as well as consistency in the results obtained.

Future Scope

There is a significant potential for further improvement in the field of query optimization/ reduction. Further enhancements could include utilizing n-grams to evaluate more than just a set of five terms at a time. This could be done by merging two or more sub-queries with overlapping terms.

Utilizing the Apache shingle [21] with n-grams could further yield improvements in query analysis. By utilizing better performing independent predictors more versions of the queries could be evaluated concurrently thus yielding better search results with minimal impact on query performance (speed). We could compare the performance (quality) of the retrieved results when the queries were collected using even as well as uneven sampling.

Delving further into the applications of random selection of query subsets could also yield a favorable improvement in query performance. But mostly consistency in the performance of the query needs further analysis. The maximum gains are sometimes very close to those returned by the ideal set of sub-queries, and yet at other times at par with our baseline, the original author's approach [2].

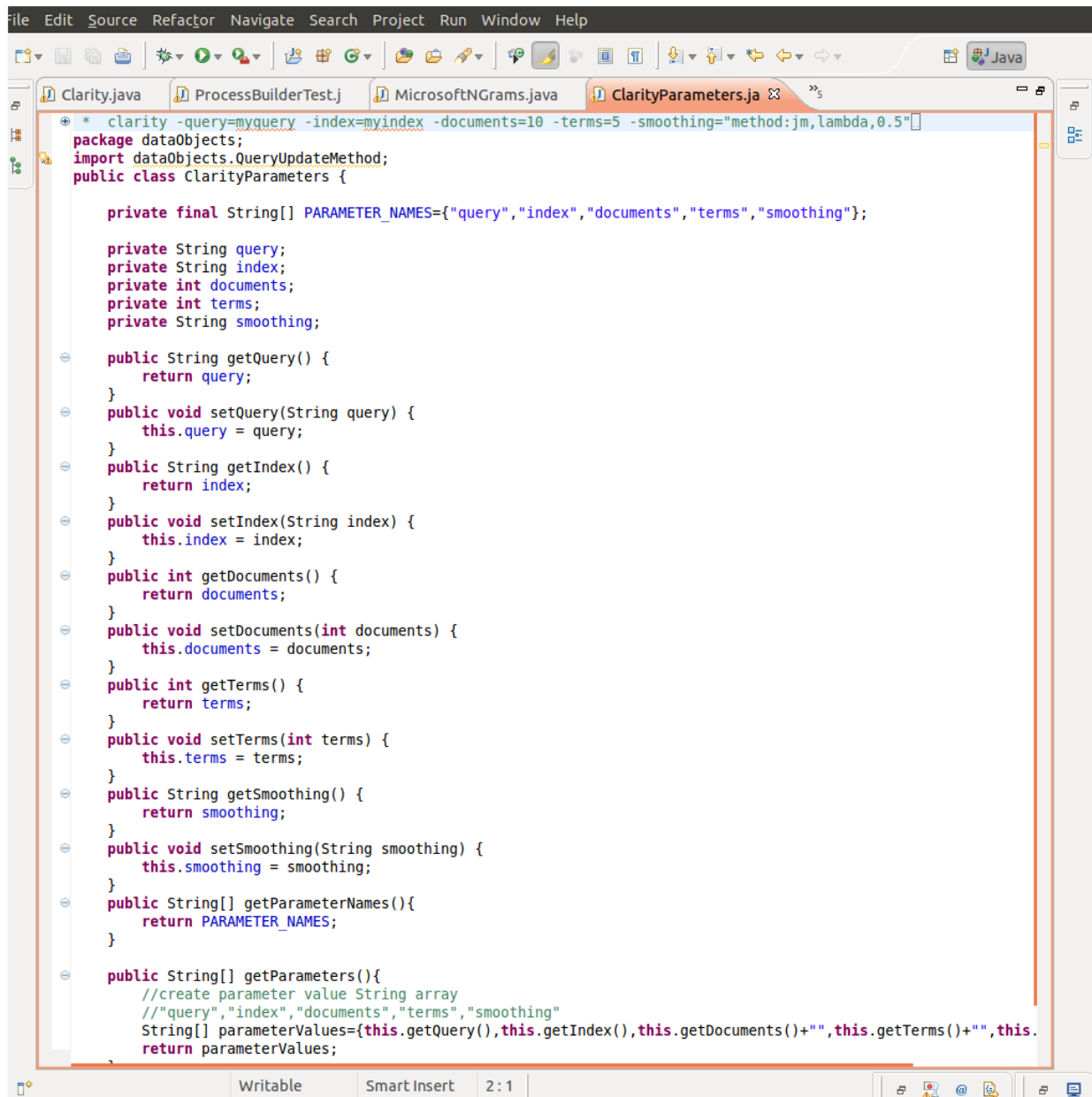
The right set of performance predictors could improve the performance of our approach. Predictors, which have low overhead and high accuracy, could lead to increased performance of ad-hoc query retrieval.

References

- [1] Chen, Yan; Zhang, Yan-Qing; "A Query Substitution-Search Result Refinement Approach for Long Query Web Searches," Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT '09. IEEE/WIC/ACM International Joint Conferences on, vol.1, no., pp.245-251, 15-18 Sept. 2009
doi: 10.1109/WI-IAT.2009.42
URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5286069&isnumber=5284878>
- [2] Niranjana Balasubramanian, Giridhar Kumaran, and Vitor R. Carvalho. 2010. Exploring reductions for long web queries. In Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval (SIGIR '10). ACM, New York, NY, USA, 571-578.
DOI=10.1145/1835449.1835545 <http://doi.acm.org/10.1145/1835449.1835545>
- [3] The ClueWeb09 dataset information. Release Version: 1.0 (May 18, 2011) Authors: Mark D. Smucker and Chandra Prakash Jethani Date: May 2011 URL: <http://lemurproject.org/clueweb09.php/>
- [4] Indri is a search engine that provides state-of-the-art text search and a rich structured query language for text collections of up to 50 million documents (single machine) or 500 million documents (distributed search). Available for Linux, Solaris, Windows and Mac OSX. URL:
<http://www.lemurproject.org/indri.php>
- [5] The Lemur Toolkit is designed to facilitate research in language modeling and information retrieval (IR), where IR is broadly interpreted to include such technologies as ad hoc and distributed retrieval with structured queries, cross-language IR, summarization, filtering, and categorization. The system's underlying architecture was built to support the technologies above. URL:
<http://www.lemurproject.org/lemur.php>
- [6] Query Clarity with Retrieval. This application computes clarity scores for an expanded query model based on pseudo-feedback documents. Performs the retrieval of those documents using the relevant SimpleKLRetMethod parameters. URL:
<http://www.lemurproject.org/doxygen/lemur/html/RetQueryClarity.html>
- [7] C. Hauff, V. Murdock, and R. Baeza-Yates. Improved query difficulty prediction for the web. In CIKM, Pages 439-448, 2008.
- [8] Searches getting longer: A weblog by Alan Long, hit wise intelligence. URL:
http://weblogs.hitwise.com/alan-long/2009/11/searches_getting_longer.html.
- [9] G. Kumaran and V. Carvalho. Reducing long queries using query quality predictors. In SIGIR, pages 564-571, 2009.
- [10] B. He and I. Ounis. Inferring query performance using pre-retrieval predictors. In SPIRE, pages 43-54, 2004.

- [11] Microsoft web n-gram services general information. URL: <http://research.microsoft.com/en-us/collaboration/focus/cs/web-ngram.aspx>
- [12] Lemur Project indexing information. URL: <http://www.lemurproject.org/lemur/indexing.php>
- [13] Lemur Project Information. URL: <http://lemurproject.org/lemur.php>
- [14] Clue Web 09 crowd sourcing file read me file
- [15] Representational state transfer. URL: http://en.wikipedia.org/wiki/Representational_State_Transfer
- [16] Information on REST Services. URL: <http://rest.elkstein.org/>
- [17] Lemur Project Information. URL: <http://www.lemurproject.org/>
- [18] Information on Indri Search Engine. URL: <http://sourceforge.net/apps/trac/lemur/wiki/Indri>
- [19] B. He and I. Ounis. Inferring query performance using pre-retrieval predictors. In SPIRE, pages 43-54, 2004.
- [20] Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. 2002. Predicting query performance. In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '02). ACM, New York, NY, USA, 299-306. DOI=10.1145/564376.564429 <http://doi.acm.org/10.1145/564376.564429>
- [21] Apache Shingle Information. URL: http://lucene.apache.org/java/3_0_3/api/contrib-analyzers/org/apache/lucene/analysis/shingle/package-summary.html
- [22] G. Kumaran and J. Allan. A case for shorter queries, and helping users create them. In HLT/NAACL, pages 220-227, 2007.
- [23] Hitwise search query analysis. URL: http://www.readwriteweb.com/archives/hitwise_search_queries_are_getting_longer.php
- [24] J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, 379-386, 2008.
- [25] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. Proceedings of the 15th international conference on World Wide Web, 387-396, 2006.
- [26] B. Tan and F. Peng. Unsupervised Query Segmentation Using Generative Language Models and Wikipedia. Proceeding of the 17th international conference on World Wide Web, 347-356, 2008.
- [27] N-grams Wikipedia information. URL: <http://en.wikipedia.org/wiki/N-gram>

Appendix A: Code Snippets



```
File Edit Source Refactor Navigate Search Project Run Window Help
Clarity.java ProcessBuilderTest.j MicrosoftNGrams.java ClarityParameters.java
* clarity -query=myquery -index=myindex -documents=10 -terms=5 -smoothing="method:jm,lambda,0.5"
package dataObjects;
import dataObjects.QueryUpdateMethod;
public class ClarityParameters {

    private final String[] PARAMETER_NAMES={"query","index","documents","terms","smoothing"};

    private String query;
    private String index;
    private int documents;
    private int terms;
    private String smoothing;

    public String getQuery() {
        return query;
    }
    public void setQuery(String query) {
        this.query = query;
    }
    public String getIndex() {
        return index;
    }
    public void setIndex(String index) {
        this.index = index;
    }
    public int getDocuments() {
        return documents;
    }
    public void setDocuments(int documents) {
        this.documents = documents;
    }
    public int getTerms() {
        return terms;
    }
    public void setTerms(int terms) {
        this.terms = terms;
    }
    public String getSmoothing() {
        return smoothing;
    }
    public void setSmoothing(String smoothing) {
        this.smoothing = smoothing;
    }
    public String[] getParameterNames(){
        return PARAMETER_NAMES;
    }

    public String[] getParameters(){
        //create parameter value String array
        //"query","index","documents","terms","smoothing"
        String[] parameterValues={this.getQuery(),this.getIndex(),this.getDocuments()+"",this.getTerms()+"",this.
        return parameterValues;
    }
}
```

Figure 10: Sample of Data Structures used

```
File Edit Source Refactor Navigate Search Project Run Window Help
AutorsApproachTest MyApproach.java ProcessBuilderTest.j *MicrosoftNgrams.jav >>4
import java.io.BufferedReader;
/**
 * This class connects to Microsoft's Rest Service and sends the requested information
 * @author jehaan
 */
public class MicrosoftNgrams {
    private static String baseUrl= "http://web-ngram.research.microsoft.com/rest/lookup.svc/bing-body/apr10/";
    private static String userToken= "u=xxxxxxxxxxxxxxxxxxxxxxxx";
    private String query=null;
    private String cp="/cp?";
    private String jp="/jp?";

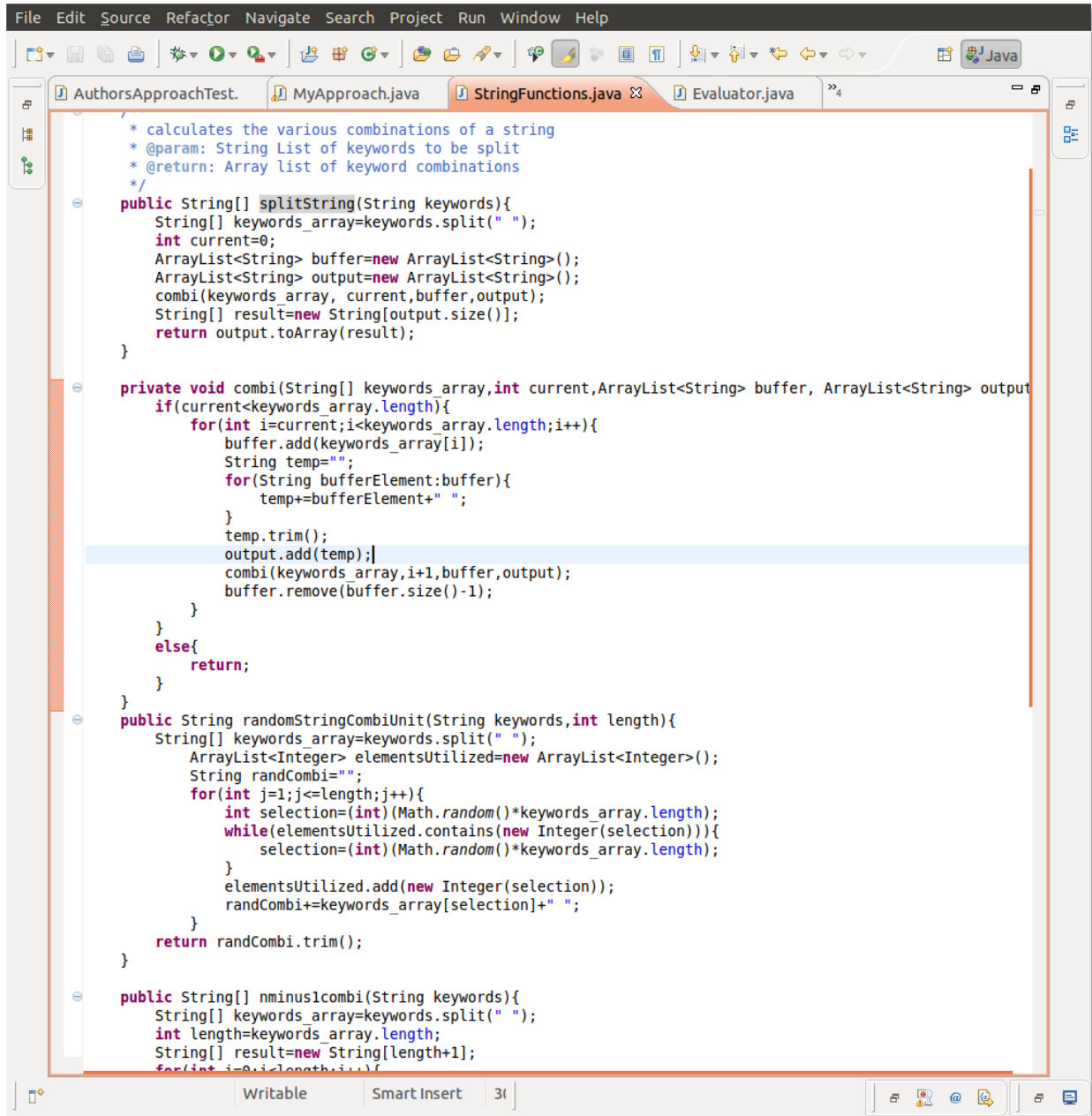
    public double getProbability(String query,String type){
        String url=null;
        double result=0;
        int counter=0;
        int length=getQueryLength(query);
        if(length!=0){
            try {
                if(type.compareToIgnoreCase("cp")==0){
                    type=cp;
                }
                else{
                    type=jp;
                }
                url=baseUrl+length+type+userToken+"&p="+URLEncoder.encode(query, "UTF-8")+"&format=json";
                while(result==0){
                    result=httpGet(url);
                }
            } catch (UnsupportedEncodingException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

        return result;
    }

    private int getQueryLength(String query){
        String[] queryArray=query.split(" ");
        int length=queryArray.length;
        return (length>5||length<1)?0:length;
    }

    private double httpGet(String urlStr) throws IOException {
        URL url = new URL(urlStr);
        HttpURLConnection conn =
```

Figure 11: Microsoft's n-gram web service connection class

The image shows a screenshot of an IDE window with several tabs: 'AuthorsApproachTest', 'MyApproach.java', 'StringFunctions.java', and 'Evaluator.java'. The 'StringFunctions.java' tab is active, displaying the following Java code:

```
File Edit Source Refactor Navigate Search Project Run Window Help
StringFunctions.java
* calculates the various combinations of a string
* @param: String List of keywords to be split
* @return: Array list of keyword combinations
*/
public String[] splitString(String keywords){
    String[] keywords_array=keywords.split(" ");
    int current=0;
    ArrayList<String> buffer=new ArrayList<String>();
    ArrayList<String> output=new ArrayList<String>();
    combi(keywords_array, current,buffer,output);
    String[] result=new String[output.size()];
    return output.toArray(result);
}

private void combi(String[] keywords_array,int current,ArrayList<String> buffer, ArrayList<String> output
    if(current<keywords_array.length){
        for(int i=current;i<keywords_array.length;i++){
            buffer.add(keywords_array[i]);
            String temp="";
            for(String bufferElement:buffer){
                temp+=bufferElement+" ";
            }
            temp.trim();
            output.add(temp);
            combi(keywords_array,i+1,buffer,output);
            buffer.remove(buffer.size()-1);
        }
    }
    else{
        return;
    }
}

public String randomStringCombiUnit(String keywords,int length){
    String[] keywords_array=keywords.split(" ");
    ArrayList<Integer> elementsUtilized=new ArrayList<Integer>();
    String randCombi="";
    for(int j=1;j<=length;j++){
        int selection=(int)(Math.random()*keywords_array.length);
        while(elementsUtilized.contains(new Integer(selection))){
            selection=(int)(Math.random()*keywords_array.length);
        }
        elementsUtilized.add(new Integer(selection));
        randCombi+=keywords_array[selection]+" ";
    }
    return randCombi.trim();
}

public String[] nminus1combi(String keywords){
    String[] keywords_array=keywords.split(" ");
    int length=keywords_array.length;
    String[] result=new String[length+1];
    for(int i=0;i<length;i++){
```

Figure 12: String Functions used

```
File Edit Source Refactor Navigate Search Project Run Window Help
* stores the best score for a run and then calculates the best of the average scores.
*/
public class AuthorsApproachTest {
    private StringFunctions stringFunctions;

    private String query;
    private ClarityParameters parameters;
    private Clarity clarity;
    private String INDEX="ClueWebIndex";
    private int DOCUMENTS=10;
    private int TERMS=10;
    private String SMOOTHING="method:jm,lambda,0.5";

    public AuthorsApproachTest(String inputString){
        //find all combinations of the query
        this.query=inputString;
        stringFunctions=new StringFunctions();
        parameters=new ClarityParameters();
        parameters.setIndex(INDEX);
        parameters.setDocuments(DOCUMENTS);
        parameters.setTerms(TERMS);
        parameters.setSmoothing(SMOOTHING);
    }

    private double executeScoring(){
        String[] combinations=stringFunctions.nminus1combi(query);
        int count=0;
        ClOutput[] results=new ClOutput[combinations.length];
        for(String option: combinations){
            //run each option
            parameters.setQuery(option);
            clarity=new Clarity(parameters);
            //save each queries score
            results[count++]=clarity.getClarityScore();
            System.out.println(results[count-1]);
        }
        Predictor predict=new Predictor(results);
        ClOutput[] diff=predict.getDifferencPrediction();

        int lengthOriginal=(query.split(" ").length);
        double tempScore=Evaluator.overallScore(diff,lengthOriginal);
        return tempScore;
    }

    public static void main(String args[]){
        AuthorsApproachTest test= new AuthorsApproachTest("New Zealand speed bike racer pinkish purple flowers");
        double output=test.executeScoring();
        System.out.println("The Score is :"+ output);
    }
}
```

Figure 13: Code snippet showing the original concepts implementation

```

File Edit Source Refactor Navigate Search Project Run Window Help
AuthorsApproachTest. MyApproach.java Clarity.java Evaluator.java ClarityTest.java
private double executeNGS(){
    System.out.println("Finding all combinations of the input string");
    String[] combinations=stringFunctions.splitString(query);
    System.out.println("Retrieving N-Grams for terms of length 1-5");
    for(String combination:combinations){
        //check if the nGrams data exists
        if(!nGScores.contains(combination)){
            double tempScore=ngramsService.getProbability(combination,"cp");
            nGScores.add(new ngramsScore(combination,tempScore));
        }
    }
    nGScores.sort();
    int queryLength= (query.split(" ").length);
    String[] queries=new String[2*queryLength];
    for(int j=0;j<queryLength;j++){
        if(j<5){
            queries[j]=nGScores.getTopnGram(j+1).getQuery();
        }
        else{
            queries[j]=stringFunctions.randomStringCombiUnit(query,j+1);
        }
    }
    String[] combinations2=stringFunctions.nminus1combi(query);
    for(int j=0;j<queryLength;j++){
        queries[j+queryLength]=combinations2[j];
    }
    //calculate query clarity for all new queries
    CLOutput[] results=new CLOutput[2*queryLength];
    int count=0;
    for(String option: queries){
        //run each option
        parameters.setQuery(option);
        clarity=new Clarity(parameters);
        //save each queries score
        results[count++]=clarity.getClarityScore();
    }
    Predictor predict=new Predictor(results);
    CLOutput[] diff=predict.getDifferencePrediction();
    double tempScore=Evaluator.overallScore(diff,queryLength);
    return (tempScore);
}

public static void main(String args[]){
    MyApproach experiment=new MyApproach("New Zealand speed bike racer pinkish purple flowers");
    double sum=0;
    for(int i=0;i<1000;i++){
        double tempScore=experiment.executeNGS();
        System.out.println("The overall score for round "+(i+1)+"is "+tempScore);
        sum+=tempScore;
    }
    System.out.println("sum: "+sum);
}

```

Figure 14: Code snippet showing our implementation