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
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
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  
Related Work  
Technologies and Projects used  
  The Clue Web 09 dataset  
  REST Services  
  Microsoft Web N-Gram Service (Public Beta) N-grams data  
  The Lemur Project  
  Indri Project  
  Indri Build Index  
  Query Clarity with retrieval  
  Retrieval User interface (RetUI)  
The Experiment  
  Technique  
  The Original Approach  
  Our Approach  
  Implementation  
  Benchmarking and results  
Conclusion  
Future Scope  
References  
Appendix A: Code Snippets
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 recreate 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.
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].

<table>
<thead>
<tr>
<th>Unique topics</th>
<th>217 topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic-docno pairs</td>
<td>19829</td>
</tr>
<tr>
<td>Unique topic-docno pairs</td>
<td>19636</td>
</tr>
<tr>
<td>Images present</td>
<td>Jpg: 18512</td>
</tr>
<tr>
<td>Images missing</td>
<td>Jpg: 1124</td>
</tr>
<tr>
<td>Pdf files present</td>
<td>17243</td>
</tr>
<tr>
<td>Pdf files missing</td>
<td>2393</td>
</tr>
<tr>
<td>Plain text files present</td>
<td>19636</td>
</tr>
<tr>
<td>Unique wget’d pages</td>
<td>19636</td>
</tr>
</tbody>
</table>

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:

```xml
<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
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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>index</strong></td>
<td>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</td>
</tr>
</tbody>
</table>
| **indexType**   | The type of the index you want to build.  
  * key for KeyfileIncIndex (.key)  
  * indri for IndriIndex (.ind) |
| **memory**      | Memory (in bytes) to pre-allocate (def = 96000000)                                                                                           |
| **Stopwords**   | Name of file containing the stopword list.                                                                                                    |
| **Acronyms**    | Name of file containing the acronym list, currently not supported by IndriIndex. These acronyms will still be indexed in lowercase by IndriIndex. |
| **countStopWords** | If true, count stopwords in document length.                                                          |
| **docFormat**   | - TREC for standard TREC formatted documents.  
  - web for web TREC formatted documents.  
  - Chinese for segmented Chinese text (TREC format, GB encoding).  
  - chinesechar for unsegmented Chinese text (TREC format, GB encoding).  
  - arabic for Arabic text (TREC format, Windows CP1256 encoding). |
| **Stemmer**     | - porter: Porter stemmer.  
  - Krovetz: Krovetz stemmer.  
  - Arabic: arabic stemmer, requires additional parameters.  
  - arabicStemDir: Path to directory of data files used by the Arabic stemmers.  
  - arabicStemFunc: Which stemming algorithm to apply, one of:  
    - arabic_stop : arabic_stop.  
    - arabic_norm2 : table normalization.  
    - arabic_norm2_stop : table normalization with stopping.  
    - arabic_light10 : light9 plus ll prefix.  
    - arabic_light10_stop : light10 and remove stop words. |
| **dataFiles**   | Name of file containing list of data files to index.                                                                                           |

*Table 6: IndriBuildIndex Parameters*
**Figure 1: Indri Index build setup**

![Indri Index Builder screenshot](image1.png)

**Figure 2: Indri indexing in progress**

![Indri Index Builder screenshot](image2.png)
Figure 3: Indices created by Indri for search and other retrieval-based applications.

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].

<table>
<thead>
<tr>
<th>Index</th>
<th>The complete name of the index table-of-content file for the database index.</th>
</tr>
</thead>
<tbody>
<tr>
<td>smoothSupportFile</td>
<td>The name of the smoothing support file (e.g., one generated by GenerateSmoothSupport).</td>
</tr>
<tr>
<td>textQuery</td>
<td>The original query text stream.</td>
</tr>
<tr>
<td>expandedQuery</td>
<td>The file to store the query clarity scores.</td>
</tr>
<tr>
<td>feedbackDocCount</td>
<td>The number of docs to use for pseudo-feedback. If not specified or 0, the value defaults to 500.</td>
</tr>
<tr>
<td>queryUpdateMethod</td>
<td>Feedback method, one of:</td>
</tr>
<tr>
<td></td>
<td>• mixture or mix or 0 for mixture.</td>
</tr>
<tr>
<td></td>
<td>• divmin or div or 1 for div min.</td>
</tr>
<tr>
<td></td>
<td>• markovchain or mc or 2 for markov chain.</td>
</tr>
<tr>
<td></td>
<td>• relevancemdl1 or rm1 or 3 for relevance model 1.</td>
</tr>
<tr>
<td></td>
<td>• relevancemdl2 or rm2 or 4 for relevance model 2.</td>
</tr>
</tbody>
</table>

For all interpolation-based approaches

| feedbackCoefficient | 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). |
| feedbackTermCount | Truncate the feedback model to no more than a given number of words/terms. |
| feedbackProbThresh | Truncate the feedback model to include only words with a probability higher than this threshold. Default value: 0.001. |
| feedbackProbSumThresh | Truncate the feedback model until the sum of the probability of the included words reaches this threshold. Default value: 1. |
| feedbackMixtureNoise | For the collection mixture model method, feedbackMixtureNoise 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, feedbackMixtureNoise 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, feedbackMixtureNoise is the
probability of not stopping, i.e., $1 - \alpha$, where $\alpha$ 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

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 that 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 $Q$. 

26
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:

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Query</th>
<th>Authors Approach</th>
<th>Our Approach</th>
<th>Best Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Professional web Hosting Service Provider HSP and Corporate IT professionals</td>
<td>6.07143</td>
<td>21.51782</td>
<td>33.73132</td>
</tr>
<tr>
<td>2</td>
<td>find tons of cheap international travel airlines and they can be found all over the place</td>
<td>6.05855</td>
<td>6.43822</td>
<td>31.3861</td>
</tr>
<tr>
<td>3</td>
<td>airframe that became The Red Baron</td>
<td>5.56061</td>
<td>9.42546</td>
<td>19.59863</td>
</tr>
<tr>
<td>4</td>
<td>searchable in a variety of ways from price to product type</td>
<td>4.635595</td>
<td>8.485841</td>
<td>35.71394</td>
</tr>
<tr>
<td>5</td>
<td>The Internet Definitive Buyers Services Guide</td>
<td>7.61017</td>
<td>7.61017</td>
<td>35.40384</td>
</tr>
<tr>
<td>6</td>
<td>The best choice of cheap downloadable OEM software is offered</td>
<td>11.56594</td>
<td>15.53296</td>
<td>40.94454</td>
</tr>
<tr>
<td>7</td>
<td>Finding the Best T1 Service Provider in Your Area</td>
<td>2.27936</td>
<td>2.93455</td>
<td>16.78798</td>
</tr>
<tr>
<td>8</td>
<td>wedding entertainment professionals who have entertained thousands of couples</td>
<td>-0.25852</td>
<td>-0.10314</td>
<td>22.7893</td>
</tr>
<tr>
<td>9</td>
<td>DJ Spinelli Assoc is a professional Disc Jockey</td>
<td>2.41148</td>
<td>2.42157</td>
<td>7.51011</td>
</tr>
<tr>
<td>10</td>
<td>planning your wedding is fun and easy</td>
<td>3.200196</td>
<td>20.45951</td>
<td>32.99578</td>
</tr>
<tr>
<td>11</td>
<td>The MinuteMan site has been online since 2002</td>
<td>2.50426</td>
<td>4.03769</td>
<td>55.63561</td>
</tr>
<tr>
<td>12</td>
<td>the NJ Environmental Digital Library Census Bureau online mapping</td>
<td>1.40685</td>
<td>4.80456</td>
<td>27.37782</td>
</tr>
<tr>
<td>13</td>
<td>Major League Baseball selects the Adobe Flash Platform</td>
<td>5.93864</td>
<td>26.4392</td>
<td>46.98035</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Scores</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>save an incredible amount of time and effort</td>
<td>0.198724 12.571999 30.72971</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Consolidate data from two or more data sources into a data warehouse</td>
<td>0.17384 2.37894 27.158895</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Flash Player bug and issue management system is now available for use by external users</td>
<td>2.11436 24.28304 48.87496</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>protects you from hackers phishing and other online fraud</td>
<td>0.00122 0.00243 1.2941</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>do not have the correct Flash Player installed</td>
<td>3.95951 14.88498 27.35322</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>If you use the Internet Explorer browser or AOL you need</td>
<td>4.278315 22.032245 48.464915</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>OEMs to differentiate their handsets and devices</td>
<td>2.74477 22.7155 29.72684</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>runtime lets developers use proven web technologies</td>
<td>-0.43894 12.25017 25.60872</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>only for purposes of achieving the distribution described</td>
<td>1.125018 4.632321 11.57964</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Inventions links of learned franklin philosopher American</td>
<td>2.88241 8.91441 22.61704</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>barber shop carson daly ben harper benchmarking ben jerry</td>
<td>0.46696 0.91797 3.55108</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>gained the recognition of scientists and intellectuals across Europe</td>
<td>0.035952 0.695969 17.22355</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>worried about all the moving arrangements</td>
<td>1.99168 2.368 10.8209</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Select from 165 Ben Franklin items available to buy</td>
<td>1.77397 20.24482 30.62086</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Ben Franklins Wit and Wisdom</td>
<td>2.48495 5.62292 11.83825</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Highway 6 at the Lake Murray Dam in Irmo</td>
<td>1.88247 5.34145 30.65717</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Glass containers are not allowed in the park</td>
<td>1.641058 13.104961 34.973295</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>I sell real estate in the Columbia area</td>
<td>0.30193 2.19788 32.84217</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>----------------------------------------------------------------------------------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>32</td>
<td>Looking for the perfect gift to spark the interest</td>
<td>0.504753</td>
<td>1.003977</td>
<td>21.20402</td>
</tr>
<tr>
<td>33</td>
<td>chairman of the Falmouth School Board</td>
<td>0.142199</td>
<td>7.719973</td>
<td>21.002735</td>
</tr>
<tr>
<td>34</td>
<td>Trout fishing is somewhat sporadic however and actually</td>
<td>1.01262</td>
<td>9.87368</td>
<td>41.90773</td>
</tr>
<tr>
<td>35</td>
<td>Build a mini fire extinguisher and float a soap</td>
<td>0.387849</td>
<td>12.799405</td>
<td>22.312555</td>
</tr>
<tr>
<td>36</td>
<td>suggest the rhythm played at the time rivaled the tempo</td>
<td>0.712627</td>
<td>9.004548</td>
<td>20.722311</td>
</tr>
<tr>
<td>37</td>
<td>Professor Probenius is your chemistry professor for CHEM</td>
<td>1.67967</td>
<td>1.67967</td>
<td>11.77977</td>
</tr>
<tr>
<td>38</td>
<td>current operating schedules and announcements visit the COSD Water Dept</td>
<td>4.38881</td>
<td>7.477932</td>
<td>22.598615</td>
</tr>
<tr>
<td>39</td>
<td>I encourage all believers to give up the shackles of faith</td>
<td>1.387086</td>
<td>7.032334</td>
<td>53.85263</td>
</tr>
<tr>
<td>40</td>
<td>The smallest particle of light is a photon</td>
<td>1.224286</td>
<td>6.118562</td>
<td>18.550834</td>
</tr>
<tr>
<td>41</td>
<td>The association uses donations to support arts</td>
<td>5.394981</td>
<td>11.217877</td>
<td>33.238945</td>
</tr>
<tr>
<td>42</td>
<td>women who are in love can recognize their partner</td>
<td>-0.003938</td>
<td>14.590807</td>
<td>39.999595</td>
</tr>
<tr>
<td>43</td>
<td>The latest release of the Virtual Earth</td>
<td>0.07193</td>
<td>0.07717</td>
<td>30.08633</td>
</tr>
<tr>
<td>44</td>
<td>The Daily Mail is encouraging its readers to buy the traditional non</td>
<td>0.79785</td>
<td>3.471969</td>
<td>50.45134</td>
</tr>
<tr>
<td>45</td>
<td>customers to search for more types of mapping information</td>
<td>1.865478</td>
<td>17.80624</td>
<td>38.39073</td>
</tr>
<tr>
<td>46</td>
<td>mashups with an intuitive JavaScript programming model</td>
<td>0.38099</td>
<td>3.67857</td>
<td>38.50879</td>
</tr>
<tr>
<td>47</td>
<td>imagery enhances our currently available data by seeing</td>
<td>0.269676</td>
<td>2.956008</td>
<td>4.200583</td>
</tr>
<tr>
<td>48</td>
<td>UK government have signed up to an EU decision</td>
<td>5.987756</td>
<td>46.01875</td>
<td>53.87314</td>
</tr>
<tr>
<td>49</td>
<td>natively be a premium content layer</td>
<td>2.567362</td>
<td>6.509855</td>
<td>31.614445</td>
</tr>
<tr>
<td></td>
<td>MSDN technical article posted online showing users how to authenticate</td>
<td>-0.33698</td>
<td>24.83715</td>
<td>33.42506</td>
</tr>
<tr>
<td>---</td>
<td>---------------------------------------------------------------------</td>
<td>-----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>51</td>
<td>see all the damage that has been done</td>
<td>0.310678</td>
<td>3.310625</td>
<td>8.501512</td>
</tr>
<tr>
<td>52</td>
<td>the only weather application that offers looping radar</td>
<td>4.204547</td>
<td>7.521331</td>
<td>22.973245</td>
</tr>
<tr>
<td>53</td>
<td>the drug is intended to help people with a rare hereditary</td>
<td>2.097099</td>
<td>5.127585</td>
<td>44.851695</td>
</tr>
<tr>
<td>54</td>
<td>Balance Board to talk to the program after decoding the Bluetooth</td>
<td>2.19096</td>
<td>11.455997</td>
<td>45.61618</td>
</tr>
<tr>
<td>55</td>
<td>sexy applications that push the limits of geospatial and Virtual Earth</td>
<td>0.67226</td>
<td>0.67226</td>
<td>62.62344</td>
</tr>
<tr>
<td>56</td>
<td>New Orleans area to show your insurance adjuster</td>
<td>7.05535</td>
<td>22.06113</td>
<td>30.12247</td>
</tr>
<tr>
<td>57</td>
<td>for its athletic programs as well as its band department</td>
<td>1.017211</td>
<td>11.951999</td>
<td>47.181005</td>
</tr>
<tr>
<td>58</td>
<td>East Ridge has gone to State Competition for Concert Band</td>
<td>1.8574</td>
<td>17.7914</td>
<td>38.06548</td>
</tr>
<tr>
<td>59</td>
<td>online mapping service that enables users to search</td>
<td>1.657023</td>
<td>15.710907</td>
<td>40.44724</td>
</tr>
<tr>
<td>60</td>
<td>Student enrolment at East Ridge High School is currently</td>
<td>0.419911</td>
<td>12.063247</td>
<td>23.629185</td>
</tr>
<tr>
<td>61</td>
<td>Microsoft provides a staging environment to test your application</td>
<td>3.03708</td>
<td>21.08689</td>
<td>31.19509</td>
</tr>
<tr>
<td>62</td>
<td>known for its athletic programs as well as its band department</td>
<td>-0.040917</td>
<td>8.094521</td>
<td>46.00286</td>
</tr>
<tr>
<td>63</td>
<td>A new director has been hired to oversee the percussion section</td>
<td>1.29421</td>
<td>4.524457</td>
<td>48.35589</td>
</tr>
<tr>
<td>64</td>
<td>Lowest prices cheap prescription diet pills</td>
<td>6.03234</td>
<td>8.37805</td>
<td>21.83186</td>
</tr>
<tr>
<td>65</td>
<td>not meant to substitute for the advice provided by your own physician</td>
<td>0.057815</td>
<td>1.965707</td>
<td>32.409165</td>
</tr>
<tr>
<td>66</td>
<td>Posted in Prescription online phentermine no prescription</td>
<td>2.93088</td>
<td>5.21027</td>
<td>14.57121</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>things running for fans around the country</td>
<td>0.485916</td>
<td>0.531482</td>
<td>7.392562</td>
</tr>
<tr>
<td>68</td>
<td>A statue to her memory stands in Slater Park</td>
<td>2.63427</td>
<td>15.85733</td>
<td>30.11823</td>
</tr>
<tr>
<td>69</td>
<td>derrick car at the Clinchfield Railroad yard</td>
<td>1.14479</td>
<td>3.17069</td>
<td>17.24397</td>
</tr>
<tr>
<td>70</td>
<td>initial startup never had anything to do with the military</td>
<td>1.332563</td>
<td>4.333282</td>
<td>25.910011</td>
</tr>
<tr>
<td>71</td>
<td>real estate virtual tour software service</td>
<td>1.46169</td>
<td>2.36188</td>
<td>16.55054</td>
</tr>
<tr>
<td>72</td>
<td>interfering with the absorption of certain nutrients in consumed food</td>
<td>0.675537</td>
<td>0.684862</td>
<td>27.51389</td>
</tr>
<tr>
<td>73</td>
<td>includes a list of ships with the same or similar names</td>
<td>0.646984</td>
<td>14.626586</td>
<td>45.89544</td>
</tr>
<tr>
<td>74</td>
<td>us presidents born in Massachusetts</td>
<td>0.54408</td>
<td>5.02678</td>
<td>6.93027</td>
</tr>
<tr>
<td>75</td>
<td>magic the gathering alpha black vice</td>
<td>0.46453</td>
<td>2.91722</td>
<td>18.65372</td>
</tr>
<tr>
<td>76</td>
<td>la times vice president public affairs</td>
<td>2.22008</td>
<td>5.1986</td>
<td>11.16651</td>
</tr>
<tr>
<td>77</td>
<td>evaluating a university vice presidential candidate</td>
<td>0.96957</td>
<td>1.19297</td>
<td>16.11441</td>
</tr>
<tr>
<td>78</td>
<td>English speaking nations largely followed either</td>
<td>7.60005</td>
<td>15.89927</td>
<td>25.50411</td>
</tr>
<tr>
<td>79</td>
<td>The fact that my two bikes are still going strong</td>
<td>0.081327</td>
<td>0.814137</td>
<td>6.787198</td>
</tr>
<tr>
<td>80</td>
<td>hybrid electric vehicle manufactured by Honda</td>
<td>2.84995</td>
<td>3.71631</td>
<td>31.71426</td>
</tr>
<tr>
<td>81</td>
<td>The raw data for Manhattan is aggregated from the NYC</td>
<td>9.951521</td>
<td>12.728404</td>
<td>36.779025</td>
</tr>
<tr>
<td>82</td>
<td>Nixon was sick on the first televised debate</td>
<td>0.854173</td>
<td>5.859417</td>
<td>27.20089</td>
</tr>
<tr>
<td>83</td>
<td>transmission up and down arrows suggest when to shift gears</td>
<td>4.294902</td>
<td>34.424209</td>
<td>48.634045</td>
</tr>
<tr>
<td>84</td>
<td>Diet pills aim to help overweight people to curb their hunger</td>
<td>6.08813</td>
<td>6.33453</td>
<td>26.56056</td>
</tr>
<tr>
<td>85</td>
<td>report to Employment and Immigration Minister Hector Goudreau</td>
<td>2.101119</td>
<td>20.6542</td>
<td>25.93064</td>
</tr>
<tr>
<td></td>
<td>significant deceleration when used in regenerative mode for braking</td>
<td>1.307598</td>
<td>17.22306</td>
<td>23.66421</td>
</tr>
<tr>
<td>---</td>
<td>-------------------------------------------------------------------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>87</td>
<td>vice president of arizona employers council</td>
<td>3.02186</td>
<td>4.0142</td>
<td>8.72438</td>
</tr>
<tr>
<td>88</td>
<td>lightweight aluminum structure to maximize fuel efficiency and minimize</td>
<td>1.55047</td>
<td>11.67285</td>
<td>18.59682</td>
</tr>
<tr>
<td>89</td>
<td>history of president franklin roosevelt</td>
<td>6.56802</td>
<td>7.03426</td>
<td>14.77119</td>
</tr>
<tr>
<td>90</td>
<td>The story goes that the military version could go</td>
<td>2.122865</td>
<td>4.312422</td>
<td>21.571245</td>
</tr>
<tr>
<td>91</td>
<td>has more than doubled in the last five years</td>
<td>1.346813</td>
<td>9.241467</td>
<td>23.44662</td>
</tr>
<tr>
<td>92</td>
<td>original factory new or used parts and manufacture parts</td>
<td>0.3509</td>
<td>9.58925</td>
<td>16.51057</td>
</tr>
<tr>
<td>93</td>
<td>he benefit may be modest and the side effects intolerable</td>
<td>1.897887</td>
<td>13.892916</td>
<td>28.397165</td>
</tr>
<tr>
<td>94</td>
<td>left so they sold them all to COMB liquidation</td>
<td>0.145526</td>
<td>15.624767</td>
<td>26.404715</td>
</tr>
<tr>
<td>95</td>
<td>superb Naomi Campbell figure is all lined</td>
<td>1.493825</td>
<td>13.976042</td>
<td>24.044755</td>
</tr>
<tr>
<td>96</td>
<td>appointment with your doctor will serve this purpose</td>
<td>3.584</td>
<td>4.918245</td>
<td>26.21529</td>
</tr>
<tr>
<td>97</td>
<td>Cathine is found in shrub Catha edulis</td>
<td>7.45465</td>
<td>19.17349</td>
<td>29.10425</td>
</tr>
<tr>
<td>98</td>
<td>may not reflect the actual production season</td>
<td>1.7464</td>
<td>2.19081</td>
<td>6.06296</td>
</tr>
<tr>
<td>99</td>
<td>five closing themes in the Japanese episodes</td>
<td>3.959086</td>
<td>11.247737</td>
<td>40.01236</td>
</tr>
<tr>
<td>100</td>
<td>Certain pills now under research and development</td>
<td>0.26553</td>
<td>0.26553</td>
<td>21.76097</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td><strong>233.048222</strong></td>
<td><strong>956.674474</strong></td>
<td><strong>2780.776101</strong></td>
<td></td>
</tr>
</tbody>
</table>

**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


[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


[14] Clue Web 09 crowd sourcing file read me file


[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


Appendix A: Code Snippets

Figure 10: Sample of Data Structures used
Figure 11: Microsoft's n-gram web service connection class
Figure 12: String Functions used

```java
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[] minusculeCombi(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++){
        result[i]=keywords_array[i].toLowerCase();
    }
    result[length]=keywords_array[length].toLowerCase();
    return result;
}
```
Figure 13: Code snippet showing the original concepts implementation
Figure 14: Code snippet showing our implementation