TSUBAKI: An Open Search Engine Infrastructure for Developing New Information Access Methodology

Keiji Shinzato[†], **Tomohide Shibata**[†], **Daisuke Kawahara**[‡], **Chikara Hashimoto**[‡‡] and **Sadao Kurohashi**[†]

[†]Graduate School of Informatics, Kyoto University
[‡]National Institute of Information and Communications Technology
[‡‡]Department of Informatics, Yamagata University

{shinzato, shibata, kuro}@nlp.kuee.kyoto-u.ac.jp
dk@nict.go.jp ch@yz.yamagata-u.ac.jp

Abstract

As the amount of information created by human beings is explosively grown in the last decade, it is getting extremely harder to obtain necessary information by conventional information access methods. Hence, creation of drastically new technology is needed. For developing such new technology, search engine infrastructures are required. Although the existing search engine APIs can be regarded as such infrastructures, these APIs have several restrictions such as a limit on the number of API calls. To help the development of new technology, we are running an open search engine infrastructure, TSUBAKI, on a high-performance computing environment. In this paper, we describe TSUBAKI infrastructure.

1 Introduction

As the amount of information created by human beings is explosively grown in the last decade (University of California, 2003), it is getting extremely harder to obtain necessary information by conventional information access methods, i.e., Web search engines. This is obvious from the fact that knowledge workers now spend about 30% of their day on only searching for information (The Delphi Group White Paper, 2001). Hence, creation of drastically new technology is needed by integrating several disciplines such as natural language processing (NLP), information retrieval (IR) and others.

Conventional search engines such as Google and Yahoo! are insufficient to search necessary information from the current Web. The problems of the conventional search engines are summarized as follows:

Cannot accept queries by natural language sentences: Search engine users have to represent their needs by a list of words. This means that search engine users cannot obtain necessary information if they fail to represent their needs into a proper word list. This is a serious problem for users who do not utilize a search engine frequently.

Cannot provide organized search results: A search result is a simple list consisting of URLs, titles and snippets of web pages. This type of result presentation is obviously insufficient considering explosive growth and diversity of web pages.

Cannot handle synonymous expressions: Existing search engines ignore a synonymous expression problem. Especially, since Japanese uses three kinds of alphabets, Hiragana, Katakana and Kanji, this problem is more serious. For instance, although both Japanese words "こども" and "子供" mean *child*, the search engines provide quite different search results for each word.

We believe that new IR systems that overcome the above problems give us more flexible and comfortable information access and that development of such systems is an important and interesting research topic.

To develop such IR systems, a search engine infrastructure that plays *a low-level layer role* (i.e., retrieving web pages according to a user's query from a huge web page collection) is required. The Application Programming Interfaces (APIs) provided by

commercial search engines can be regarded as such search engine infrastructures. The APIs, however, have the following problems:

1. The number of API calls a day and the number of web pages included in a search result are limited.
2. The API users cannot know how the acquired web pages are ranked because the ranking measure of web pages has not been made public.
3. It is difficult to reproduce previously-obtained search results via the APIs because search engine's indices are updated frequently.

These problems are an obstacle to develop new IR systems using existing search engine APIs.

The research project "Cyber Infrastructure for the Information-explosion Era[1]" gives researchers several kinds of shared platforms and sophisticated tools, such as an open search engine infrastructure, considerable computational environment and a grid shell software (Kaneda et al., 2002), for creation of drastically new IR technology. In this paper, we describe an open search engine infrastructure **TSUBAKI**, which is one of the shared platforms developed in the Cyber Infrastructure for the Information-explosion Era project. The overview of TSUBAKI is depicted in Figure 1. TSUBAKI is built on a high-performance computing environment consisting of 128 CPU cores and 100 tera-byte storages, and it can provide users with search results retrieved from approximately 100 million Japanese web pages.

The mission of TSUBAKI is to help the development of new information access methodology which solves the problems of conventional information access methods. This is achieved by the following TSUBAKI's characteristics:

API without any restriction: TSUBAKI provides its API without any restrictions such as the limited number of API calls a day and the number of results returned from an API per query, which are the typical restrictions of the existing search engine APIs. Consequently, TSUBAKI API users can develop systems that handle a large number of web pages. This feature is important for dealing with the Web that has the long tail aspect.

[1]http://i-explosion.ex.nii.ac.jp/i-explosion/ctr.php/m/IndexEng/a/Index/

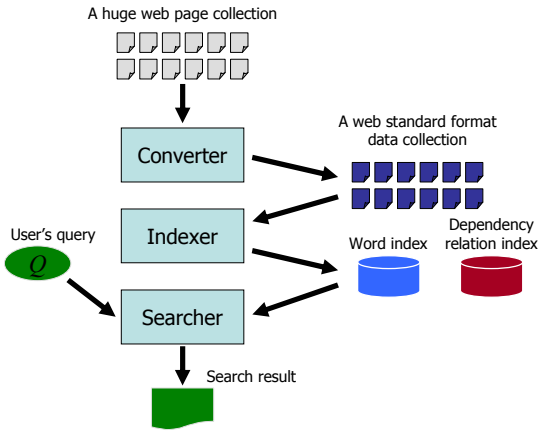

Figure 1: An overview of TSUBAKI.

Transparent and reproducible search results: TSUBAKI makes public not only its ranking measure but also its source codes, and also provides reproducible search results by fixing a crawled web page collection. Because of this, TSUBAKI keeps its architecture transparency, and systems using the API can always obtain previously-produced search results.

Web standard format for sharing pre-processed web pages: TSUBAKI converts a crawled web page into a web standard format data. The web standard format is a data format used in TSUBAKI for sharing pre-processed web pages. Section 2 presents the web standard format in detail.

Indices generated by deep NLP: TSUBAKI indexes all crawled web pages by not only words but also dependency relations for retrieving web pages according to the meaning of their contents. The index data in TSUBAKI are described in Section 3.

This paper is organized as follows. Section 2 describes web standard format, and Section 3 shows TSUBAKI's index data and its search algorithm. Section 4 presents TSUBAKI API and gives examples of how to use the API. Section 5 shows related work.

2 Sharing of Pre-processed Web Pages on a Large Scale

Web page processing on a large scale is a difficult task because the task generally requires a

high-performance computing environment (Kawahara and Kurohashi, 2006) and not everybody can use such environment. Sharing of large scale pre-processed web pages is necessary for eliminating the gap yielded by large data processing capabilities.

TSUBAKI makes it possible to share pre-processed large scale web pages through the API. TSUBAKI API provides not only cached original web pages (i.e., 100 million pages) but also pre-processed web pages. As pre-processed data of web pages, the results of commonly performed processing for web pages, including sentence boundary detection, morphological analysis and parsing, are provided. This allows API users to begin their own processing immediately without extracting sentences from web pages and analyzing them by themselves.

In the remainder of this section, we describe a web standard format used in TSUBAKI for sharing pre-processed web pages and construction of a large scale web standard format data collection.

2.1 Web Standard Format

The web standard format is a simple XML-styled data format in which meta-information and text-information of a web page can be annotated. The meta-information consists of a title, in-links and out-links of a web page and the text-information consists of sentences extracted from the web page and their analyzed results by existing NLP tools.

An example of a web standard format data is shown in Figure 2. Extracted sentences are enclosed by <RawString> tags, and the analyzed results of the sentences are enclosed by <Annotation> tags. Sentences in a web page and their analyzed results can be obtained by looking at these tags in the standard format data corresponding to the page.

2.2 Construction of Web Standard Format Data Collection

We have crawled 218 million web pages over three months, May - July in 2007, by using the Shim-Crawler,[2] and then converted these pages into web standard format data with results of a Japanese parser, KNP (Kurohashi and Nagao, 1994), through our conversion tools. Note that this web page collection consists of pages written not only in Japanese but also in other languages.

```
<?xml version="1.0" encoding="UTF-8"?>
<StandardFormat
Url="http://www.kantei.go.jp/jp/koizumiprofile/1_sinnen.html" OriginalEncoding="Shift_JIS" Time="2006-08-14 19:48:51"><Text Type="default">
<S Id="1" Length="70" Offset="525">
  <RawString>小泉総理の好きな格言のひとつに「無信不立（信無くば立たず）」があります。</RawString>
  <Annotation Scheme="KNP">
    <![CDATA[* 1D <文頭><サ変><人名><助詞><連体修飾><体言><係:ノ格><区切:0-4><RID:1056>
小泉 こいずみ 小泉 名詞 6 人名 5 * 0 * 0 NIL <文頭><漢字><かな漢字><名詞相当語><自立><タグ単位始><文節始><固有キー>
 ...
ます ます ます 接尾辞 14 動詞性接尾辞 7 動詞性接尾辞ます型 31 基本形 2 NIL <表現文末><かな漢字><ひらがな><活用語><付属><非独立無意味接尾辞>
。 。 。 特殊 1 句点 1 * 0 * 0 NIL <文末><記号><付属>
    EOS]]>
  </Annotation>
</S>
...
</Text>
</StandardFormat>
```

Figure 2: An example of web standard format data with results of the Japanese parser KNP.

The web pages in the collection are converted into the standard format data according to the following four steps:

Step 1: Extract Japanese web pages from a given page collection.

Step 2: Detect Japanese sentence boundaries in the extracted web pages.

Step 3: Analyze the Japanese sentences by the NLP tools.

Step 4: Generate standard format data from the extracted sentences and their analyzed results.

We followed the procedure proposed in Kawahara and Kurohashi (2006) for Steps 1 and 2.

The web pages were processed by a grid computing environment that consists of 640 CPU cores and 640 GB main memory in total. It took two weeks to finish the conversion. As a result, 100 million web standard format data were obtained. In other words, the remaining 118 million web pages were regarded as non-Japanese pages by our tools.

The comparison between original web pages and the standard format data corresponding to these pages in terms of file size are shown in Table 1. We

[2] http://www.logos.t.u-tokyo.ac.jp/crawler/

Table 1: File size comparison between original web pages and standard format data (The number of web pages is 100 millions, and both the page sets are compressed by gzip.)

Document set	File size [TB]
Original web pages	0.6
Standard format styled data	3.1

can see that the file size of the web standard format data is over five times bigger than that of the original web pages.

3 Search Engine TSUBAKI

In this section, we describe the indices and search algorithm used in TSUBAKI.

3.1 Indices in TSUBAKI

TSUBAKI has indexed 100 million Japanese web pages described in Section 2.2. Inverted index data were created by both words and dependency relations. Note that the index data are constructed from parsing results in the standard format data.

3.1.1 Word Index

Handling of synonymous expressions is a crucial problem in IR. Especially, since Japanese uses three kinds of alphabets, Hiragana, Katakana and Kanji, spelling variation is a big obstacle. For example, the word "*child*" can be represented by at least three spellings "こども", "子ども" and "子供" in Japanese. Although these spellings mean *child*, existing search engines handle them in totally different manner. Handling of spelling variations is important for improving search engine performance.

To handle spelling variations properly, TSUBAKI exploits results of JUMAN (Kurohashi et al., 1994), a Japanese morphological analyzer. JUMAN segments a sentence into words, and gives representative forms of the words simultaneously. For example, JUMAN gives us "子供" as a representative form of the words "こども", "子ども" and "子供." TSUBAKI indexes web pages by word representative forms. This allows us to retrieve web pages that include different spellings of the queries.

TSUBAKI also indexes word positions for providing search methods such as an *exact phrase search*. A word position reflects the number of words appearing before the word in a web page. For example, if a page contains N words, the word appearing in the beginning of the page and the last word are assigned 0 and $N - 1$ as their positions respectively.

3.1.2 Dependency Relation Index

The understanding of web page contents is crucial for obtaining necessary information from the Web. The word frequency and link structure have been used as clues for conventional web page retrieval. These clues, however, are not sufficient to understand web page's contents. We believe that other clues such as parsing results of web page contents are needed for the understanding.

Let us consider the following two sentences:

S1: Japan exports automobiles to Germany.

S2: Germany exports automobiles to Japan.

Although the above sentences have different meanings, they consist of the same words. This means that a word index alone can never distinguish the semantic difference between these sentences.

On the other hand, syntactic parsers can produce different dependency relations for each sentence. Thus, the difference between these sentences can be grasped by looking at their dependency relations. We expect that dependency relations work as efficient clues for understanding web page contents.

As a first step toward web page retrieval considering the meaning of web page contents, TSUBAKI indexes web pages by not only words but also dependency relations. An index of the dependency relation between *A* and *B* is represented by the notation $A \rightarrow B$, which means *A modifies B*. For instance, the dependency relation indices *Japan* $\rightarrow$ *export*, *automobile* $\rightarrow$ *export*, *to* $\rightarrow$ *export*, and *Germany* $\rightarrow$ *to* are generated from the sentence **S1**.

3.1.3 Construction of Index data

We have constructed word and dependency relation indices from a web standard format data collection described in Section 2.2. The file size of the constructed indices are shown in Table 2. We can see that the file size of the word index is larger than that of dependency relation. This is because that the word index includes all position for all word index expression.

Table 2: File sizes of the word and dependency relation indices constructed from 100 million web pages.

Index type	File size [TB]
Word	1.17
Dependency relation	0.89

Table 3: Comparison with index data of TSUBAKI and the Apache Lucene in terms of index data size (The number of web pages is a million.)

Search engine	File size [GB]
TSUBAKI (words)	12.0
TSUBAKI (dependency relations)	9.1
Apache Lucene	4.7

Moreover, we have compared index data constructed by TSUBAKI and the Apache Lucene,[3] an open source information retrieval library, in terms of the file size. We first selected a million web pages from among 100 million pages, and then indexed them by using the indexer of TSUBAKI and that of the Lucene.[4] While TSUBAKI's indexer indexed web pages by the both words and dependency relations, the Lucene's indexer indexed pages by only words. The comparison result is listed in Table 3. We can see that the word index data constructed by TSUBAKI indexer is larger than that of the Lucene. But, the file size of the TSUBAKI's index data can be made smaller because the TSUBAKI indexer does not optimize the constructed index data.

3.2 Search Algorithm

TSUBAKI is run on a load balance server, four master servers and 27 search servers. Word and dependency relation indices generated from 100 million web pages are divided into 100 pieces respectively, and each piece is allocated to the search servers. In short, each search server has the word and dependency relation indices generated from at most four million pages.

The procedure for retrieving web pages is shown in Figure 3. Each search server calculates relevance scores between a user's query and each document that matches the query. We used the sum of OKAPI BM25 (Robertson et al., 1992) scores over index expressions in the query as the relevance score. The relevance score $score_{rel}$ is defined as:

$$score_{rel}(Q, d) = \sum_{q \in Q} BM25(q, d)$$

$$BM25(q, d) = w \times \frac{(k_1 + 1)fq}{K + fq} \times \frac{(k_3 + 1)qfq}{k_3 + qfq}$$

$$w = \log \frac{N - n + 0.5}{n + 0.5}, K = k_1((1 - b) + b\frac{l}{l_{ave}})$$

where q is an index expression extracted from the query Q, fq is the frequency of the expression q in a document d, qfq is the frequency of q in Q, and N is the total number of crawled web pages. TSUBAKI used 1.0×10^8 as N. n is the document frequency of q in 100 million pages, l is the document length of d (we used the number of words in the document d), and l_{ave} is the average document length over all the pages. In addition to them, the parameters of OKAPI BM25, k_1,k_3 and b were set to 2, 0 and 0.75, respectively.

Consider the expression "*global warming's effect*" as a user's query Q. The extracted index expressions from Q are shown in Figure 4. Each search server calculates a BM25 score for each index expression (i.e., *effect*, *global*, ... , *global* $\rightarrow$ *warm*), and sums up the calculated scores.

Note that BM25 scores of dependency relations are larger than those of single words because the

Step 1: The load balance server forwards user's query Q to the most unoccupied master server.

Step 2: The master server extracts the set of index expressions q from the given query Q, and transmits the set of q and search conditions such as a logical operator (i.e., AND/OR) between words in Q to 27 search servers.

Step 3: The search server retrieves web pages according to the set of q and search conditions by using word and dependency relation indices.

Step 4: The search server calculates a relevance score for each retrieved document, and then returns the documents with their scores to the master server.

Step 5: The master server sorts the returned documents according to their calculated scores.

Step 6: The top M documents are presented to the user as a search result.

Figure 3: The search procedure of TSUBAKI. (Steps 3 and 4 are performed in parallel.)

[3] http://lucene.apache.org/java/docs/index.html

[4] We used the Lucene 2.0 for Japanese which is available from https://sen.dev.java.net/servlets/ProjectDocumentList?folderID=755&ex pandFolder=755&folderID=0

Word index: *effect*, *global*, *warm*,
Dependency relation index: *global* → *warm*, *warm* → *effect*

Figure 4: The index expressions extracted from the query "*global warming's effect.*"

document frequencies of dependency relations are relatively smaller than those of single words. Consequently, TSUBAKI naturally gives high score values to web pages that include the same dependency relations as the one included in the given query.

4 TSUBAKI API

As mentioned before, TSUBAKI provides the API without any restriction. The API can be queried by "REST (Fielding, 2000)-Like" operators in the same way of Yahoo! API. TSUBAKI API users can obtain search results through HTTP requests with URL-encoded parameters. Examples of the available request parameters are listed in Table 4. The sample request using the parameters is below:

Case 1: Get the search result ranked at top 20 with snippets for the search query "京都 (*Kyoto*)".
http://tsubaki.ixnlp.nii.ac.jp/api.cgi?query=%E4%BA%AC%E9%83%BD&starts=1&results=20

TSUBAKI API returns an XML document in Figure 5 for the above request. The result includes a given query, a hitcount, the IDs of web pages that match the given query, the calculated scores and others. The page IDs in the result enable API users to obtain cached web pages and web standard format data. An example request for obtaining the web standard format data with document ID 01234567 is below.

Case 2: Get web standard format data with the document ID 01234567.
http://tsubaki.ixnlp.nii.ac.jp/api.cgi?id=01234567&format=xml

The hitcounts of words are frequently exploited in NLP tasks. For example, Turney (Turney, 2001) proposed a method that calculates semantic similarities between two words according to their hitcounts obtained from an existing search engine. Although TSUBAKI API users can obtain a query's hitcount from a search result shown in Figure 5, TSUBAKI API provides an access method for directly obtaining query's hitcount. The API users can obtain only a hitcount according to the following HTTP request.

Table 4: The request parameters of TSUBAKI API.

Parameter	Value	Description
query	*string*	The query to search for (UTF-8 encoded). The query parameter is required for obtaining search results.
start	*integer*: default 1	The starting result position to return.
results	*integer*: default 20	The number of results to return.
logical_operator	AND/OR: default AND	The logical operation to search for.
dpnd	0/1: default 1	Specifies whether to use dependency relations as clues for document retrieving. Set to 1 to use dependency relations.
only_hitcounts	0/1: default 0	Set to 1 to obtain a query's hitcount only.
snippets	0/1: default 0	Set to 1 to obtain snippets.
id	*string*	The document ID to obtain a cached web page or standard format data corresponding to the ID. This parameter is required for obtaining web pages or standard format data.
format	html/xml	The document type to return. This parameter is required if the parameter id is set.

Case 3: Get the hitcount of the query "京都 (*Kyoto*)"
http://tsubaki.ixnlp.nii.ac.jp/api.cgi?query=%E4%BA%AC%E9%83%BD&only_hitcounts=1

In this case, the response of the API is a plain-text data indicating the query's hitcount.

5 Related Work

As mentioned before, existing search engine APIs such as Google API are insufficient for infrastructures to help the development of new IR methodology, since they have some restrictions such as a limited number of API calls a day. The differences between TSUBAKI API and existing search engine APIs are summarized in Table 5. Other than access restrictions, the serious problem of these APIs is that they cannot always reproduce previously-provided

```
<ResultSet time="2007-10-15 14:27:01" query="京都" totalResultsAvailable="4721570" totalResultsReturned="20"
          firstResultPosition="1" logicalOperator="AND" forceDpnd="0" dpnd="1" filterSimpages="1">
   <Result Rank="1" Id="017307147" Score="8.87700">
      <Title>ＪＴＢ　ｅ－Ｈｏｔｅｌの京都府のホテル・旅館一覧</Title>
      <Url>http://www.docch.net/blog/jtb-e/kyouto.shtml</Url>
      <Snippet/>
      <Cache>
         <Url>http://tsubaki.ixnlp.nii.ac.jp/index.cgi?URL=INDEX_DIR/h017/h01730/017307147.html&KEYS=%E4%BA%
              AC%E9%83%BD</Url>
         <Size>2900</Size>
      </Cache>
   </Result>
 ...
</ResultSet>
```

Figure 5: An example of a search result returned from TSUBAKI API.

search results because their indices are updated frequently. Because of this, it is difficult to precisely compare between systems using search results obtained on different days. Moreover, private search algorithms are also the problem since API users cannot know what goes on in searching web pages. Therefore, it is difficult to precisely assess the contribution of the user's proposed method as long as the method uses the existing APIs.

Open source projects with respect to search engines such as the Apache Lucene and the Rast[5] can be also regarded as related work. Although these projects develop an open search engine module, they do not operate web search engines. This is different from our study. The comparison between TSUBAKI and open source projects with respect to indexing and ranking measure are listed in Table 6.

The Search Wikia project[6] has the similar goal to one of our goals. The goal of this project is to create an open search engine enabling us to know how the system and the algorithm operate. However, the algorithm of the search engine in this project is not made public at this time.

The Web Laboratory project (Arms et al., 2006) also has the similar goal to ours. This project aims at developing an infrastructure to access the snapshots of the Web taken by the Internet Archive.[7] Currently the pilot version of the infrastructure is released. The released infrastructure, however, allows users to access only the web pages in the Amazon.com Web site. Therefore, TSUBAKI is different from the infrastructure of the Web Laboratory project in terms of the scale of a used web page collection.

[5] http://projects.netlab.jp/rast/

[6] http://search.wikia.com/wiki/Search_Wikia

[7] http://www.archive.org/index.php

Table 5: The differences between TSUBAKI API and existing search engine APIs.

Features	Google	Yahoo!	TSUBAKI
# of API calls a day	1,000	50,000	unlimited
# of URLs in a search result	1,000	1,000	unlimited
Providing cached pages	Yes	Yes	Yes
Providing processed pages	No	No	Yes
Updating indices	Yes	Yes	No

Table 6: Comparison with indexing and ranking measure.

Search Engine	Indexing	Ranking Measure
TSUBAKI	word, dependency relation	OKAPI BM25
Apache Lucene	character bi-gram, word	TF·IDF
RAST	character bi-gram, word	TF·IDF

6 Conclusion

We have described TSUBAKI, an open search engine infrastructure for developing new information access methodology. Its major characteristics are:

- the API without any restriction,
- transparent and reproducible search results,
- Web standard format for sharing pre-processed web pages and
- indices generated by deep NLP.

TSUBAKI provides not only web pages retrieved from 100 million Japanese pages according to a user's query but also pre-processed large scale web

pages produced by using a high-performance computing environment.

On the TSUBAKI infrastructure, we are developing a new information access method that organizes retrieved web pages in a search result into clusters of pages that have relevance to each other. We believe that this method gives us more flexible information access than existing search methods.

Furthermore, we are building on the TSUBAKI infrastructure a common evaluation environment to evolve IR methodology. Such an environment is necessary to easily evaluate novel IR methodology, such as a new ranking measure, on a huge-scale web collection.

Our future work is to handle synonymous expressions such as "*car*" and "*automobile*." Handling synonymous expressions is important for improving the performance of search engines. The evaluation of TSUBAKI's performance is necessary, which is also our future work.

References

Wiiliam Y. Arms, Selcuk Aya, Pavel Dmitriev, Blazej J. Kot, Ruth Mitchell, and Lucia Walle. 2006. Building a research library for the history of the web. In *Proceedings of the Joint Conference on Digital Libraries, June 2006*, pages 95–102.

Roy Thomas Fielding. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. Ph.D. thesis, University of California, Irvine.

Kenji Kaneda, Kenjiro Taura, and Akinori Yonezawa. 2002. Virtual private grid: A command shell for utilizing hundreds of machines efficiently. In *In 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2002)*.

Daisuke Kawahara and Sadao Kurohashi. 2006. Case frame compilation from the web using high-performance computing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC2006)*, pages 1344–1347.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, (4):507–534.

Sadao Kurohashi, Toshihisa Nakamura, Yuji Matsumoto, and Makoto Nagao. 1994. Improvements of japanese morphological analyzer juman. In *The International Workshop on Sharable Natural Language*, pages 22 – 28.

Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. 1992. Okapi at TREC. In *Text REtrieval Conference*, pages 21–30.

The Delphi Group White Paper. 2001. Connecting to your knowledge nuggets. http://www.delphiweb.com/knowledgebase/documents/upload/pdf/1802.pdf.

Peter Turney. 2001. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-2001)*, pages 491–502.

University of California. 2003. How much information? 2003. http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/.