2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology

A Text Similarity Meta-Search Engine Based on Document Fingerprints and Search Results Records

Felipe Bravo-Marquez, Gaston L'Huillier, Sebastián A. Ríos, Juan D. Velásquez
Web Intelligence Consortium Chile Research Centre
Department of Industrial Engineering
University of Chile
Santiago, Chile
fbravo@dcc.uchile.cl, {glhuilli,srios,jvelasqu}@dii.uchile.cl

***Abstract*—The retrieval of similar documents from the Web using documents as input instead of key-term queries is not currently supported by traditional Web search engines. One approach for solving the problem consists of fingerprint the document's content into a set of queries that are submitted to a list of Web search engines. Afterward, results are merged, their URLs are fetched and their content is compared with the given document using text comparison algorithms. However, the action of requesting results to multiple web servers could take a significant amount of time and effort. In this work, a similarity function between the given document and retrieved results is estimated. The function uses as variables features that come from information provided by search engine results records, like rankings, titles and snippets. Avoiding therefore, the bottleneck of requesting external Web Servers. We created a collection of around 10,000 search engine results by generating queries from 2,000 crawled Web documents. Then we fitted the similarity function using the cosine similarity between the input and results content as the target variable. The execution time between the exact and approximated solution was compared. Results obtained for our approximated solution showed a reduction of computational time of 86% at an acceptable level of precision with respect to the exact solution of the web document retrieval problem.**

***Keywords*-Meta-Search Engine, Similar Document Retrieval, Document Fingerprinting, Query Generation, Ranking Fusion**

I. Introduction

Web search engines have been developed with the aim of solving information requirements from users, allowing them to represent these requirements in elaborated query languages which support Boolean operators, wild-card queries, exact phrases, etc. However, common users employ few key terms to represent their information needs [18].

As proposed in [11], Web search queries can be grouped into three categories: Informational queries, which seek general information of a specific topic; navigational queries, which seek a single Web page that the user has in mind; and transactional queries which represent an intention from the user related with an action, like purchasing products or downloading a file. In [9], a different information requirement is described, that consists of retrieving the most similar documents from the Web using as input a given document instead of a query. In this work, this problem is called as the Web Document Similarity Retrieval Problem (WDSRP). Likewise, we define a Web Document Similarity Retrieval Tool (WDSRT) as a Web tool which receives a document as input and returns the most similar Web documents scored and ranked by a similarity measure.

Different contexts in which a WDSRT could be applied, are presented below:

1) *Plagiarism Detection*: In the educational context, the massification of the Web and search engines, has contributed to access large bibliographic contents, much larger than the generally needed for students' assignments. Likewise, the quality of reviewers' task for authenticity checking in delivered documents has been compromised [12]. The mechanism that most reviewers have adopted for authenticity checking, is to extract and submit suspicious text passages to Web search engines. Unfortunately, commercial search engines represent queries by HTTP request URIs which are limited by a maximum size, admitting therefore, a fixed number of characters in their queries. For which the suspicious text needs to be chopped into several parts. Then, checking all results, the original document manually searched in all retrieved documents. This process can take a significant amount of time, and the detection of malicious activities is not guaranteed. In this context, a WDSRT would clearly facilitate the reviewer's labor.
2) *Duplicate detection*: Near-Duplicate Detection, as described by [8], consists in identifying documents with slight textual or formatting differences within a document corpus. Thus, it is usually used by Web crawlers for avoiding indexing the same content many times. Given this, a WDSRT can be used to identify Web duplicates as high scored results.
3) *Impact analysis*: A strong relationship between how many times a text passage is present in the Web and the impact of its content could be assumed. Likewise, we could analyze the impact of a text passage founded

978-0-7695-4513-4/11 $26.00 © 2011 IEEE
DOI 10.1109/WI-IAT.2011.27

in Web sources like news, forums, or blog entries by submitting it to a WDSRT. Moreover, the number of high scored results retrieved could be used as an impact measure.

4) *Related ideas retrieval*: An idea, a patent, a poem, or a new theory could be used as input in a WDSRT in order to find similar ideas, the author would not need to define key terms related to the idea.

The Web search engine architecture proposed in [3], [11] based on building an inverted index over crawled Web documents, could be used for a WDSRT implementation. This data structure allows the retrieval of "relevant" documents from a query, looking up the query terms in the index, merging their post lists and returning top k relevant results. Hence, documents could be used as input in Web search engines and the WDSRP would be solved. However, the hardware resources required in order to build a search engine architecture supporting documents as queries are enormous. Indeed, some of the reasons why commercial search engines restrict length of queries, are that long queries take a huge computation time, are hard to cache, and are usually composed of many irrelevant terms.

Considering the high resources required for building a complete search engine with a high coverage of the Web, the idea of using external inverted index from commercial search engines makes a lot of sense. Furthermore, in order to have a better coverage from the Web, results from several search engines could be combined into a meta-search engine architecture[1]. Below we present two critical factors to consider for building a WDSRT using a meta-search engine approach:

1) **Input size restriction**: In order to deal with the query length restriction described above, the document must be chopped into several pieces of text called fingerprints to be used as query inputs. Therefore, a document query generation strategy needs to be developed.
2) **Absence of a similarity measure**: Commercial search engines don't return a similarity measures between the query and results. One possible strategy is to download the content from high ranked URLs and compute their similarities with the input using text comparison algorithms. However, requesting URLs to several Web servers could result in a bottleneck if we want to return results to the final user in few seconds. Hence, the similarity could be estimated directly using information provided by using search engines results, like rankings, titles, and snippets [10].

This work's contribution is a text similarity meta-search engine that given a document, represents its content with a set of generated queries and using several search engines retrieves from the Web similar documents from this source. Then, estimates the similarity between the input and results using a function which consider as features just the information provided by search engines result records, like rankings, titles and snippets (result summary) of retrieved results. Hence, results can be scored and ranked in few seconds, and issues like developing a complete search engine architecture from scratch, requesting results from several Web servers or computing text processing algorithms over large document collections are avoided.

The structure of this paper is defined as follows: In section II, previous work on document fingerprinting, meta search engines, and previous developments for WDSRP are reviewed. Then, in section III, two query generation techniques are introduced. In section IV, the meta-search engine query submission and the results gathering procedure are explained. In section V the extraction of features from gathered results is explained, where Zipf-like and snippet similarity features are presented. In section V-C we present a similarity function estimation based on the extracted features as predictors and the real cosine similarity as target variable. In section VI the experimental setup is explained, and results are discussed. Finally, in section VII the main conclusions of this paper and future work are detailed.

II. Related Work

In this section, different techniques for document query generation and fingerprinting are reviewed. Also, the latest approaches for meta-search engines and ranking fusion methods are discussed. Finally, previous solutions to the WDSRP are presented.

A. Document Fingerprinting and Query Generation

Fingerprinting techniques are document content representation approaches based on the extraction of moderately sized text chunks from its content [21]. They are commonly used in plagiarism detection [15] by comparing them rather than the whole content. Therefore, partial similarities within the documents can be identified. Instead, in the WDSRP context the fingerprints are used to represent a given document in a set of queries.

Fingerprint approaches proposed in [15], [9], [21] consider the selection of a fixed number of document sentences. These approaches are closely related to the selection of sequential words limited by a character length or the extraction of k-grams. Some of the criteria proposed for sentence selection are: sentences containing non lexical terms (like UFO or GNU), proportionally distributed sentences, or sentences containing terms that achieve the highest *tf-idf* score from the vector space model [3].

Furthermore, considering that queries are often treated as bag of words by search engines, compositions of term extractions can be used as document fingerprints [17]. Some deterministic approaches consider selecting terms with the

[1]A meta-search engine is a a tool, which aggregates results from multiple search engines in a single list.

highest frequency or *tf-idf*. And some probabilistic approaches consider a randomized extraction of terms using probabilities proportional with the frequency or *tf-idf* weights.

B. Meta Search Engines

As described in [16], meta-search engines provide a single unified interface, where a user enters a specific query, the engine forwards it in parallel to a list of search engines, whose results are collated and ranked into a single list. Hence, data fusion strategies must be considered in order to achieve a global ranking function over several local ranked results [10]. Some ranking fusion techniques are presented below: Borda-count is a ranking fusion method discussed in [2]. The model is based on democratic voting, where Web documents are considered as candidates and search instances as voters. For each search instance a result is given $c-(r-1)$ points where r is the local ranking and c is the total number of points to be distributed between all retrieved results. For documents that are not retrieved by a search instance, the remaining points are divided evenly among them [10]. Once results from different search instances are gathered, points from documents retrieved in different search instances are added. Finally, documents are ranked in order of total points. Weighted Borda-fuse is a variation from Borda-count where search engines are not treated equally using a search engine confidence factor in the point assignment. Finally, Bayes-fuse uses a probabilistic naive Bayes approach for estimating the probability of a result to be relevant to a query.

C. Web Document Similarity Retrieval Tools

The WDSRP has been studied by different researchers, where most solutions of the problem have been developed in the context of plagiarism detection. The first approach are Text Similarity Search Engines (TSSEs), where all issues from a commercial search Engine like crawling and indexing are considered. These kinds of search engines allow the submission of large text passages providing information retrieval operations over document databases. *Turnitin*[2] and *eTBLAST*[3] are well known examples of this approach. In the second approach external commercial search engines are used. Fingerprinting techniques for document representation into a set of queries are used, and similar candidate documents from the search engines are retrieved. Pereira and Ziviani propose in [9] to retrieve the complete text from top ranked results, and compare them using text comparison strategies, like Patricia trees and Shingles method.

Furthermore, Zaka presented in [21] an approach where results snippets are compared with the generated query using the cosine similarity from the vector space model. In [5] a scoring function is proposed based on result ranks, but the model was only tested using short paragraphs as inputs instead of whole documents. *Duplichecker*[4] submits each sentence directly into a single search engine without merging results or similarity score. Another example, *Plagium*[5] delivers ranked and scored results using Yahoo! as external search engine. Finally, a third example *Copyscape*[6] allows submitting documents and retrieving them using Google search API.

III. Document Fingerprinting for Query Generation

In this section, a query generation procedure based on fingerprinting techniques is presented. The main goal is to retrieve the most similar results to the document D given as input. In order to have a good representation of the document's content, we combine two fingerprinting techniques. The first one, is a *Hypergeometric Language Model* [4], [19] which creates bag of words queries using a randomized term extraction without replacement. The second one is an *n-gram random sample* approach, which extracts proportionally distributed random n-grams. Both techniques are complementary, where the former prioritizes relevant terms without considering the order in which the words appeared within the document, and the latter aims to ensure the coverage of the documents content. It is important to consider that before the query generation procedure, the input text should be cleaned by removing stopwords and special characters.

A. Hypergeometric Language Model

As stated in [11], given a document D, from which a vocabulary V can be extracted, a language model M_D from D is a function that maps a probability measure over strings drawn from V. Language models are used as ranking functions in information retrieval, scoring documents with the probability of generating the query q given the document language model M_D. In our query generation task, the probabilistic distribution from the language model is used as a randomized term extraction procedure. The reason for using randomized term permutations, is that similar documents from D do not necessarily contain words in the same order. Furthermore, as a strong but realistic assumption, search engines, where queries will be submitted, treat user natural text queries following a bag of words property [3].

The Hypergeometric Language Model (HLM) proposed in [4] is an extension of language models inspired in the multivalued hypergeometric distribution [7], where terms are extracted one by one without replacement. The property contributes to the hypothesis that a new term gives more information to a search engine than a repeated term in the generated query. This concept has been used before to identify the most relevant terms in document corpuses [1].

[2]http://turnitin.com [online: accessed 03-03-2011]
[3]http://etest.vbi.vt.edu/etblast3/ [online: accessed 03-03-2011]
[4]http://www.duplichecker.com [online: accessed 03-03-2011]
[5]http://www.plagium.com/ [online: accessed 03-03-2011]
[6]http://www.copyscape.com/ [online: accessed 03-03-2011]

Hypergeometric Language Models are generalized in [19]. The model and the query generation algorithm are presented in the following.

Consider that the extracted vocabulary is determined by $V = \{t_1, \ldots, t_m\}$, where each term t_i in the document has an assigned positive value w_i stored in vector $\overrightarrow{w} = \{w_1, \ldots, w_m\}$. These values can be determined by several weighting approaches, like term frequency or *tf-idf* weights.

A generated query q can be modeled as a sequence of tokens defined by $q = s_1, \ldots s_n$, where each token $s_j \in q$ is an integer taking values in $\{1 \ldots m\} \in V$. By using the chain rule of probabilities, the probability of generating the query q from a language model M_D, can be defined as, $P(q|M_D) = P(s_1|M_D) \ldots P(s_n|s_1, \ldots, s_{n-1}, M_D)$. In HLM, the conditional distribution of extracting the token s_j given M_D is estimated by $\hat{P}(s_j|M_D) = \frac{w_{s_j}}{||\overrightarrow{w}||_1}$ where $||\overrightarrow{w}||_1 = \sum_{i=1}^{m} |w_i|$. Then, considering the non replacement extraction property from HLM, the conditional distribution of generating the token s_k given an accumulated generated query $q = s_1, \ldots s_n$ and the language model M_D, is determined by $\hat{P}(s_k|q, M_D) = \frac{w_{s_k}}{||\overrightarrow{w}||_1 - \sum_{j=1}^{n} w_{s_j}}$ taking a zero value if the token value was already generated.

The query generation algorithm extract sequence of terms using the probabilities described above. The number of terms in a query is assigned by a parameter $length$. The algorithm gives higher probabilities of occurrence to most relevant terms using the weighting approach as relevant criteria parameter (term frequency in our case). The extraction is modeled with a multinomial distribution parametrized by $(\frac{\overrightarrow{w}}{||\overrightarrow{w}||_1})$. The non replacement property is modeled by reconstructing the multinomial distribution after each extraction, where the dimensions of the $\overrightarrow{w}$ and V vectors are reduced removing the corresponding extracted terms. The process is presented in the following algorithm:

Algorithm 1: HLM-QueryGeneration

Data: $D, length, weightApproach$
Result: q
Initialize $q = \{\}$;
V $\leftarrow$ ExtractVocabulary(D);
$\overrightarrow{w} \leftarrow$ ExtractWeigthVector(D,weigthApproach);
Multinomial $m \leftarrow$ CreateMultinomial$\left(\frac{\overrightarrow{w}}{||\overrightarrow{w}||_1}\right)$;
$i \leftarrow 0$;
while $i < length$ ***and*** $\overrightarrow{w}$*.size()* > 0 **do**
  $s \leftarrow$ extractRandomElement(m) ;
  $q.add(s)$;
  $\overrightarrow{w}$.remove(s) ;
  V.remove(s) ;
  $m \leftarrow$ CreateMultinomial$\left(\frac{\overrightarrow{w}}{||\overrightarrow{w}||_1}\right)$;
  $i \leftarrow i + 1$;
return q;

In order to achieve a set of different queries, the algorithm can be used several times over the same document, where the random extractions will produce queries different from each other.

B. Random n-gram sample

An n-gram or n-shingle for a document is a sequence of n contiguous tokens from its content. The *Random n-gram sample* (RNS) fingerprint approach aims to extract a sample of proportionally distributed n-grams ensuring that the terms of a query belong to the same topic. Algorithm parameters are the length of n-grams n, the number of desired queries Q, and document D. It starts extracting an array of all words of the document called $unigrams$. Then, each n-gram is built by the extraction of words from $unigrams$ from position pos to pos+n, where the pointer makes a random jump proportional to the number of words divided by Q. The iteration continues until the word pointer is lower than the size of $unigrams$. Each jump is multiplied by a random number between $1 - \epsilon$ and $1 + \epsilon$ with $\epsilon \in [0, 1]$, ensuring the construction of different query sets by running the algorithm several times over the same document. The process is detailed in algorithm 2.

Algorithm 2: RNS-QueryGeneration

Data: D, n, Q, ϵ
Result: $sample$
Initialize $sample = \{\}$;
$unigrams \leftarrow ExtractWords(D)$;
$start \leftarrow n * random(1 - \epsilon, 1 + \epsilon)$;
while $start + n < unigrams.size()$ **do**
  $end \leftarrow start + n$;
  $ngram \leftarrow getNgram(unigrams, start, end)$;
  $sample.add(ngram)$;
  $start \leftarrow end + (\frac{unigrams.size}{Q}) * random(1-\epsilon, 1+\epsilon)$;
return $sample$;

IV. Query Set Submission and Merging Results

At this part of the process, document D is represented by a set of queries Q generated with the combination of both fingerprinting techniques previously described. Assuming that we have a set of S external search engines available, each query $q \in Q$ is submitted to a search engine $s \in S$. Each search engine of S should receive a considerable amount of queries in order to ensure the retrieval of similar documents from D for each search engine. As the coverage of the Web is potentiated by using several search engines, the document retrieval is based on the union of the indexed documents presented in each of the search engines. The hypothesis is that if the inverted indexes of S contains documents with a higher similarity to D, these documents should be responded in many search instances in the top ranks.

For every query, an asynchronous request into the assigned search engine is executed, therefore avoiding falling into a busy-wait state. Then, for all search instances, each

search engine will respond the top k results retrieved. Then we use the information contained in these results to create a set of *queryAnswer* objects. A *queryAnswer* object ω is defined as a tuple (s, q, r) , where r represents the ranking assigned by search engine s for query q. Furthermore, the URL, the title and the snippet from the result are stored as variables in the object.

After all asynchronous requests have been responded to, we proceed to merge the *queryAnswer* objects retrieved from multiple search instances into new sets of objects called *metaAnswer*. A *metaAnswer* object Ω ($|\Omega| \geq 1$) is a set of *queryAnswer* objects ω, where all *queryAnswer* $\omega \in \Omega$ share the same URL value. We will have therefore, one *metaAnswer* object for every distinct Web document retrieved, and the cardinality of a *metaAnswer*, will represent the number of hits of a Web document for all our search instances.

Once we have merged all *queryAnswers* in *metaAnswers*, we proceed to extract features from the *metaAnswers* in order to estimate the similarities between D and each different Web document retrieved.

V. Feature extraction and Estimated Similarity Function

The set of retrieved *metaAnswer* objects contains a list of similar document candidates from D. Nevertheless, a similarity measure between these documents and D still needs to be calculated. Considering that we do not want to fetch these URLs from several Web servers and compute the similarities using text comparison algorithms, the similarities must be estimated using information contained in the *metaAnswer* set. In this section, we present two different features, that will be used as variables in the estimated similarity function.

A. Zipf-Like Feature

The Zipf-like Feature (ZLF) based on the Zipf-like scoring function [4] aims to concentrate in one single value the cardinality of a *metaAnswer* and the rankings of their *queryAnwers* in a similar manner that Borda-Fuse does. The Zipf law [22] has been used in the natural language community for the analysis of term frequencies in documents. As stated by [13], if f denotes the popularity of an item and r denotes its relative rank, then f and r are related as $f = \frac{c}{r^{\beta}}$, where c is a constant and $\beta > 0$. If $\beta = 1$, then f follows exactly the Zipf law, otherwise, is it said to be Zipf-like.

In [13], the Web popularity is modeled as the Zipf law, where the relative frequency with which Web pages are requested for the r^{th} most popular Web page is proportional to $1/r$. Furthermore, in this work we propose to model the relevance of a search engine result with a Zipf-like distribution, considering that the relevance of results presented in a Web search engine are inversely related to their rankings.

In this work, all queries are generated from the same document and have a common underlying search intention. If a specific URL has been founded by many search instances and top ranked, the probability of being similar to D should be high. Thus, the value of the Zipf-like feature for a *metaAnswer* Ω is expressed by

$$ZLF(\Omega) = \frac{1}{|Q|} \sum_{\omega \in \Omega} \frac{c_s}{r^{\beta_s}} \quad (1)$$

The value is normalized by the number of queries requested, in order to represent $ZLF(\Omega) \in [0, 1]$ and making it independent from the number of queries. The parameters c and β are confidence factors in the quality of search engines' results, where $c_s \in [0, 1]$ represents the average relevance of the best response of s, and β_s represents the decay factor of the results' relevance while the amount of retrieved results increases. These values make the feature flexible for treating each different search engine in a proper manner. Nevertheless, finding the optimal values of these parameters is beyond the scope of this work, so we set for each search engine both parameter values to 1. Thus, an exact Zipf law is being assumed.

B. Title-Snippet similarity Feature

The title and the summary (a.k.a snippet) of a search engine result record provide partial information of the pointed document's content. It's important to consider that snippets from commercial search engines are query-dependent and usually contain several query terms [11]. Furthermore, in a particular *metaAnswer* object, we have several search instances where the same Web document was retrieved and different terms from D were used in the queries. Thus, we can assume that snippets from a same *metaAnswer* are not equal to each other. We proceed therefore, to combine these snippets together with the document title into a vector space model aiming to build an approximated representation of the pointed document's content.

The vector space model [14] is a vectorial representation of documents, where each term of the vocabulary is a dimension. We propose using term frequency weights for the given document D and Boolean weights for the *metaAnswer* Ω in order to avoid that repeated passages in snippets bias the representation. The *Title-Snippet similarity Feature* (TSF) is the cosine similarity between vector space models from D and Ω presented in the following formula:

$$TSF(\Omega) = \frac{\sum_{i=1}^{|V|} (w(t_i, D) \times w(t_i, \Omega))}{\sqrt{\sum_{i=1}^{|V|} w(t_i, D)^2} \times \sqrt{\sum_{i=1}^{|V|} w(t_i, \Omega)^2}} \quad (2)$$

C. Estimated Similarity Function

Once the proposed features are obtained, we proceed to estimate the similarity between document D and each document retrieved. We assume that features ZLF and TSF are strongly related with the similarity between D and the Web document pointed by a *metaAnswer* Ω. Thus, we

propose modeling the similarity sim using the following hypothesis function:

$$sim(D, \Omega) \approx h(ZLF(\Omega), TSF(\Omega); \theta) \quad (3)$$

The function h is used to predict the target variable sim and parameters θ can be fitted using methods such as Artificial Neural Networks (ANNs) [20], among other regression techniques.

VI. Experiments and Analysis

According to the previously described procedures, a prototype was implemented[7]. It was developed in the Java programming language (`JDK 1.6.0`) and as external search engines, Google and Yahoo! were used. As described in [4], for each search engine 2 HLM queries and 3 RNS queries were generated, where the length for both query generation techniques was settled to 6.

In this section the dataset construction and the hypothesis function training procedure is presented, together with the evaluation of the precision and the performance of the proposed model.

A. Dataset Construction

In order to fit the function, a dataset of training examples Z must be constructed.

$$Z = \{(ZLF(\Omega), TSF(\Omega), (sim(D, \Omega))^*\}$$

Each training example $z \in Z$ is a triple composed by the values of ZLF and TSF from a *metaAnswer* Ω and the similarity between the given document D and the content of the Web document pointed by Ω.

The dataset must be created by submitting documents to the meta-search engine and computing the features from the retrieved *metaAnswers*. Then the value of sim is obtained by fetching the URLs from results, extract their content and compute the similarity with the document used as input. We propose using the cosine similarity from the vector space model as similarity value.

Once a large dataset has been collected, the function is trained using several regression approaches like linear regressions, neural networks and support vector regressions among others. Finally the function h is used to score and rank the retrieved *metaAnswers* and results are presented to the final user.

B. Validating the Features and Fitting the hypothesis

In order to create the dataset Z, the content from several crawled Web pages was extracted and submitted to the prototype. A total of 12,019 results were retrieved, of which the features were calculated. Afterwards the URLs from these results were fetched, their content was extracted, and the cosine similarity between them and the documents given as inputs was computed.

In order to validate our pair of features, the correlation between them and the similarity variable was calculated. We obtained a correlation of 0.42 and 0.36 for ZLF and STF respectively. Therefore, there is enough evidence to validate the assumption that our features are related to the target variable. Another interesting insight was that the correlation between proposed features was low (0.13). We can conclude that each feature provides different information to describe the target variable.

Using cross-validation we trained a linear regression and a single hidden layer ANN over the data-set, where the correlation between the predicted and the observed variable was measured together with the root mean squared error (RMSE). We obtained a correlation of 0.52 for the linear regression and of 0.55 for the ANN, and a RMSE of 0.25 and 0.27 respectively. Considering that the regression achieved the lowest value of RMSE and acceptable level of correlation, we stayed with it as our hypothesis function h with the following coefficients θ:

$$h(ZLF, TSF; \theta) = 0.1 + 0.9 \times ZLF + 0.7 \times TSF \quad (4)$$

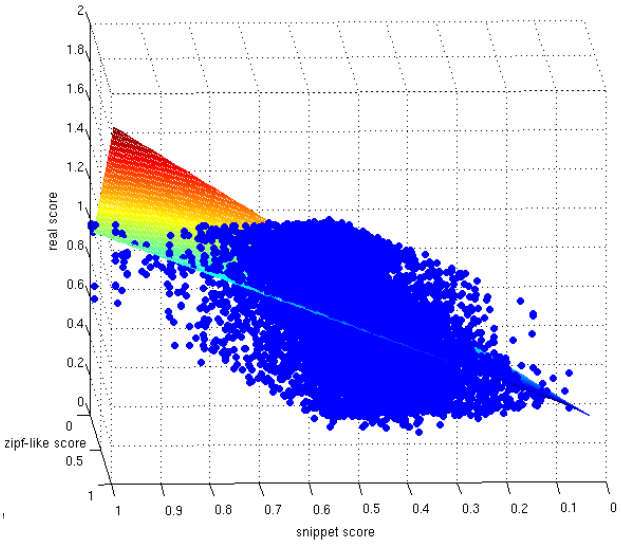

Figure 1. Obtained hyperplane and dataset mapped in TSF, ZLF, and target feature.

As presented in Figure 1, all data points are distributed in the TSF, ZLF, and the target value sim according to a hyperplane determined by the proposed linear regression.

C. Retrieval Rate, Precision and Performance

The goal of this experiment is to measure the effectiveness of the model at satisfying user information needs. In this case, those needs are related to the WDSRP. We created

[7]http://146.83.5.15:8080/Docode/ [online: accessed 03-03-2011]

α level	hits	retrieval rate
0.50	1794	0.895
0.60	1680	0.838
0.70	1528	0.762
0.80	1383	0.690
0.90	1118	0.558
0.95	1076	0.537
0.99	895	0.447

Table I
RETRIEVAL RATE

a dataset of $2,004$ documents crawled from the Web. The content of these documents was submitted into the prototype, where the content of the top 5 results scored with the hypothesis function were stored, together with the cosine similarities with the documents given as inputs. Considering that each document used as input in the experiment was extracted from the Web, we know that there is at least one Web document similar to it. Therefore, we consider as a *hit* in our experiment, in case of retrieving at least one similar document within the top 5 results.

The number of hits at a similarity level α was defined as the number of cases when at least one of the top 5 results has a similarity greater or equal than α. Likewise, we define the retrieval rate as the number of hits divided into the number of attempts. Table I shows the retrieval rate at different values of α.

The precision at k is an information retrieval evaluation measure [11], which measures the fraction of relevant results retrieved within the top k results.

$$\text{precision at } (k) = \frac{\text{relevant results retrieved within top } k}{\text{total top } k \text{ results retrieved}} \tag{5}$$

In this experiment, results which the cosine similarity with the document given as input was greater than or equal an $\alpha \in [0.5, 0.9]$ similarity cutoff value, were considered as relevant. In Figure 2 precision curves for different α values are presented.

Results presented above indicate that, even by increasing the cutoff similarity for considering a result as relevant, similar documents are often retrieved and ranked on top.

We also compared the performance between the estimated similarity function and the process of computing the cosine similarity by fetching URLs from results. We obtained an average execution time of 2.7 seconds for the former and of 18.82 for the latter. Hence, our approximated solution achieves a 86% reduction of execution time.

VII. Conclusions and Future Work

To the best of our knowledge, there are no methods for the Web document similarity retrieval problem (WDSRP) based on a text similarity meta-search engine architecture which uses mainly information provided by the search engines' results records. This work's experimental results

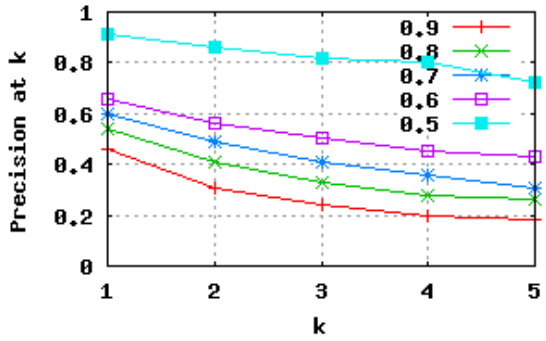

Figure 2. Precision at k at different relevant cutoff

showed that our approximated solution is able to satisfy the requirements of a WDSRT with acceptable levels of precision and providing a high reduction of execution time.

As future work, the predictive ability of the model could be improved by identifying new features from result records. Here, approaches from fields like computational linguistics and natural language processing could be adopted. Furthermore, these techniques could also be used in the query generation task. For example, the sentence selection approach used for automatic text summarization in [6] could be applied in order to identify sentences providing relevant information like names, places or dates.

Acknowledgment

Authors would like to thank "Instituto Sistemas Complejos de Ingeniería" (ICM: P-05-004- F, CONICYT: FBO16; www.isci.cl) for their continuous support; FONDEF project (DO8I-1015) entitled, DOCODE: Document Copy Detection (www.docode.cl); and the Web Intelligence Research Group (wi.dii.uchile.cl).

References

[1] Giambattista Amati. Information theoretic approach to information extraction. In *Flexible Query Answering Systems*, volume 4027 of *Lecture Notes in Computer Science*, pages 519–529. Springer Berlin / Heidelberg, 2006.

[2] Javed A. Aslam and Mark Montague. Models for metasearch. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 276–284, New York, NY, USA, 2001. ACM.

[3] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. 1999.

[4] Felipe Bravo-Marquez, Gastón LHuillier, Sebastián Ríos, and Juan Velásquez. Hypergeometric language model and zipf-like scoring function for web document similarity retrieval. In *String Processing and Information Retrieval*, volume 6393 of *Lecture Notes in Computer Science*, pages 303–308. Springer-Verlag, 2010.

[5] Felipe Bravo-Marquez, Gastón LHuillier, Sebastián Ríos, Juan Velásquez, and Luis Guerrero. Docode-lite: A meta-search engine for document similarity retrieval. In *Knowledge-Based and Intelligent Information and Engineering Systems*, volume 6277 of *Lecture Notes in Computer Science*, pages 93–102. Springer-Verlag, 2010.

[6] Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, and Jaime Carbonell. Summarizing text documents: sentence selection and evaluation metrics. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 121–128, New York, NY, USA, 1999. ACM.

[7] William L. Harkness. Properties of the extended hypergeometric distribution. *Ann. Math. Statist.*, 36(3):938–945, 1965.

[8] Monika Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 284–291, New York, NY, USA, 2006. ACM.

[9] Álvaro R. Pereira Jr. and Nivio Ziviani. Retrieving similar documents from the web. *J. Web Eng.*, 2(4):247–261, 2004.

[10] Yiyao Lu, Weiyi Meng, Liangcai Shu, Clement T. Yu, and King-Lup Liu. Evaluation of result merging strategies for metasearch engines. In *WISE*, volume 3806 of *Lecture Notes in Computer Science*, pages 53–66. Springer, 2005.

[11] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[12] H. Maurer, F. Kappe, and B. Zaka. Plagiarism – a survey. *Journal of Universal Computer Science*, 12(8):1050–1084, 2006.

[13] S. V. Nagaraj. *Web Caching And Its Applications*. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[14] Gerard Salton, Chung-Shu Yang, and Anita Wong. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, 1975.

[15] Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. Winnowing: local algorithms for document fingerprinting. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 76–85, New York, NY, USA, 2003. ACM.

[16] Erik Selberg and Oren Etzioni. The metacrawler architecture for resource aggregation on the web. *IEEE Expert*, 12:11–14, 1997.

[17] Gabriel L. Somlo and Adele E. Howe. Using web helper agent profiles in query generation. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 812–818, New York, NY, USA, 2003. ACM.

[18] Amanda Spink, Dietmar Wolfram, B.J. Jansen, and Tefko Saracevic. Searching the web: the public and their queries. *J. Am. Soc. Inf. Sci. Technol.*, 52:226–234, 2001.

[19] Manos Tsagkias, Maarten de Rijke, and Wouter Weerkamp. Hypergeometric language models for republished article finding. In *Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR' 11, New York, NY, USA, 2011. ACM.

[20] Paul John Werbos. *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*. Wiley-Interscience, New York, NY, USA, 1994.

[21] Bilal Zaka. Empowering plagiarism detection with a web services enabled collaborative network. *J. Inf. Sci. Eng.*, 25(5):1391–1403, 2009.

[22] George K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.