DOCODE-Lite: A Meta-Search Engine for Document Similarity Retrieval

Felipe Bravo-Marquez[1], Gaston L'Huillier[1], Sebastián A. Ríos[1], Juan D. Velásquez[1], and Luis A. Guerrero[2]

[1] University of Chile, Department of Industrial Engineering, Santiago, Chile
{fbravo,glhuilli}@dcc.uchile.cl, {srios,jvelasqu}@dii.uchile.cl
[2] University of Chile, Department of Computer Science, Santiago, Chile
luguerre@dcc.uchile.cl

Abstract. The retrieval of similar documents from large scale datasets has been the one of the main concerns in knowledge management environments, such as plagiarism detection, news impact analysis, and the matching of ideas within sets of documents. In all of these applications, a light-weight architecture can be considered as fundamental for the large scale of information needed to be analyzed. Furthermore, the relevance score for documents retrieval can be significantly improved using several previously built search engines and taking into account the relevance feedback from users. In this work, we propose a web-services architecture for the retrieval of similar documents from the web. We focus on software engineering to support the manipulation of users' knowledge into the retrieval algorithm. An human evaluation for the relevance feedback of the system over a built set of documents is presented, showing that the proposed architecture can retrieve similar documents by using the main search engines. In particular, the document plagiarism detection task was evaluated, for which its main results are shown.

Keywords: Meta-Search architecture, Plagiarism Detection, Random Query Generation, Document Similarity Retrieval, SOA.

1 Introduction

The access to massive amount of information is considered as one of the main potentials of the Internet, in particular for educational institutions [9]. However, this access can lead to problems such as document plagiarism, that unfortunately has taken leadership in our today's society. Furthermore, 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 their assignments.

Likewise, reviewers' task for authenticity checking in delivered documents, has been compromised. This phenomenon is presented in both educational institutions as well as the academic environment, where scientific contributions are constantly being delivered. The mechanism that most reviewers has adopted for authenticity checking, is to search for the suspicious extract of the document in a search engine. Then, checking all results, the original document manually searched in all retrieved documents. This process can take large amounts of time, and the detection of malicious activities is not guaranteed.

A solution to this, is to develop a web crawler that indexes the complete document from a given seed of URLs. Then after a dataset of documents is indexed, search mechanisms could be used for document similarity evaluation from within this set of documents. Unfortunately, resources needed to the crawler and indexing steps are difficult to get. A second alternative is to submit the whole document in a search engine. Unfortunately, search engines admits a fixed number of characters in their queries, for which the document needs to be chopped into several parts, and then delivered in parts to the search engine, which leads to the same reviewer problem stated in the latter.

This work's contribution is an information system architecture that given a text passage source, using different search engines, retrieves from the Web similar documents from this source. Furthermore, the proposed knowledge management system is based on a web service architecture and developed using Object Oriented Programming (OOP), for which it is highly scalable and extensible to new information retrieval requirements.

This paper is structured as follows: In section 2, previous work on knowledge management information systems for plagiarism detection using meta-search engine architectures is presented. Then, in section 3, the main contribution of this work and proposed architecture is described. In section 4 the experimental setup and results are presented. Finally in section 5, the main conclusions of this paper and in section 6 future work is detailed.

2 Previous Work

As described in [8], meta search engines provides a single unified interface, where a user enters an specific query, the engine forwards it in parallel to a given list of search engines, and results are collated and ranked into a single list. Several of these approaches have been developed in [1,10,11].

In [6] the idea of exploiting directly the scores of each search engine is proposed, where the main information is the relative rank of each result. In [1], different ranking approaches are analyzed, for example Borda-fuse which is based on democratic voting, the Borda count, or the weighted borda-fuse, in which search engines are not treated equally. The document similarity retrieving problem has been studied by different researchers [4,12]. These approaches propose fingerprinting techniques for document representation into sets of relevant terms. Also, these approaches uses meta search engine architectures for retrieving an extended list of similar candidate documents. On the one hand, in [12] document snippets are retrieved from search engines and compared with the query document using cosine similarity from their Vector Space Model [7]. The previous process is deployed in a collaborative Service Oriented Architecture (SOA), where multiple services for search result analysis are developed in a modular and flexible architecture. On the other hand, Pereira and Ziviani [4] proposes to retrieve the complete text from each web document, comparing them with the query document using text comparison strategies, like Patricia trees and k-grams or shingles method. Non of these techniques' approach have used the user knowledge to enhance results.

3 DOCODE-Lite

In the following, the proposed architecture for document similarity retrieval from web documents is presented. Firstly, all the components and processes are described with their respective interactions. Secondly, the query generating process and the metasearch scoring function are described. Finally, all software engineering components are detailed in order to show how the architecture can be constructed and deployed.

3.1 Web Service Architecture

In order to allow a flexible communication with other applications, the system was developed as a Web Service. We used in its implementation the `Axis2 1.5` Web Services/WSDL/SOAP engine. The DOCODE-lite architecture (Fig. 1) is determined by different components. Each one is responsible to handle different tasks of document similarity retrieval problem, whose interaction is described as follows:

The process starts with an input paragraph and finishes with a ranked list of web documents output. These documents are candidates from input paragraph and scored by their similarity.

The input is converted in a set of queries, which are generated by a randomized process, that gives higher probabilities of occurrence to the more relevant terms. This queries are sent in parallel to a customizable list of search engines.

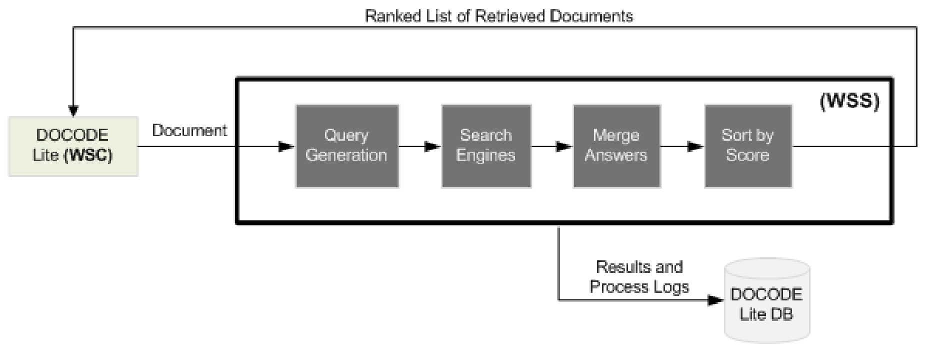

Fig. 1. DOCODE-lite SOA client and server for similar document retrieval

The search instances results are collated and scored using their local ranks, the number of appearances, and search engines reliability parameters. Finally, retrieved documents are ranked by this score, stored in a data base model, and returned to the Web Service Client. The user interface also allows the user to evaluate the relevance of results. This evaluations are stored in the database for a posterior effectiveness evaluation.

The reason of developing the system as a Web Service, is to inter-operate with different systems, where the needed service is independent to our user interface. Therefore, batch document processing tasks can be executed directly by using the proposed architecture.

3.2 Document Fingerprinting Model and Meta Search Scoring Function

The query generation process from a paragraph is based on a randomized fingerprinting model. The process starts with the extraction of the document vocabulary and the assignment of term extraction probabilities, calculated using customizable weighting approaches like *tf*, *tf-idf*, among others. These weights are normalized in order to satisfy the probability normalization constraint. Afterwards, we proceed to construct our queries by the concatenation of successive randomized term extractions. Terms are extracted without replacement in order to increment the occurrence probabilities of the others. The length of queries is also customizable, understanding that short queries achieved very fast retrieval times in search engines as stated in [3].

This process aims to represent relevant information from the given document in a set of queries. All queries are sent to the list of search engines, whose results associated to the same URLs are gathered. Each single result ω is defined a a tuple (s, q, r) where s is the used search engine, q the requested query, and r the rank in which the result has appeared. A collated result set Ω is defined as a set of single results which point to the same URL. The scoring function for these results is defined as follows:

$$score(\Omega) = \frac{1}{N} \sum_{\omega \in \Omega} \frac{c_{s(\omega)}}{(r_{\omega})^{\beta_{s(\omega)}}} \tag{1}$$

Where N is the number of requested queries and c, β represents confidence factors in the search engine s. The $c \in [0, 1]$ is a constant which represents the average relevance of the best response of a Web search engine s, and β represents the decay factor of the results' relevance while increasing the amount of results retrieved. This score measure, aims to estimate the relevance of the collated result using its local ranks and the Web search engine reliabilities.

This strategy is inspired in the idea that, if an URL appears many times and in higher ranks, the probability that the pointed document is related to the given document from which the query was generated should by high.

3.3 Software Design and Architecture Engineering

As presented in the class diagram[1], Fig. 2, previously described components are modeled with an object oriented programming approach. **DocodeLiteService** class receives as constructor input the document as string, the Search engines list, and the number of queries to generate. Queries are generated in a **QueryGenerator** object instance and used to build a **MetaSearch** object. This object creates instances of **Search** objects with their respective list of search engines, and the generated queries as their constructor signature. Each Search Engine must be implemented according to abstract class **Search** inheritance. The **MetaSearch** object creates **Search** instances using the meta-programming reflection paradigm, which allows to create a class instance by only having their class name and constructor parameters. Each **Search** instance must request the assigned query to the Search Engine and create the **QueryAnswer** instance for each result. The **QueryAnswer** object (ω) has as instance variables the pointed web document url, its title, and its local rank of occurrence. The **MetaSearch** object collates and aggregates **QueryAnswers** when they point to the same url. The aggregation process is supported with a **Hashmap** data structure, mapping each url to a single **MetaAnswer** (Ω), whose score (Eq. 1) is determined by the **QueryAnswer** instances variables. Finally, the **DocodeLiteService** picks all **MetaAnswers** and returns them to the Web Service client.

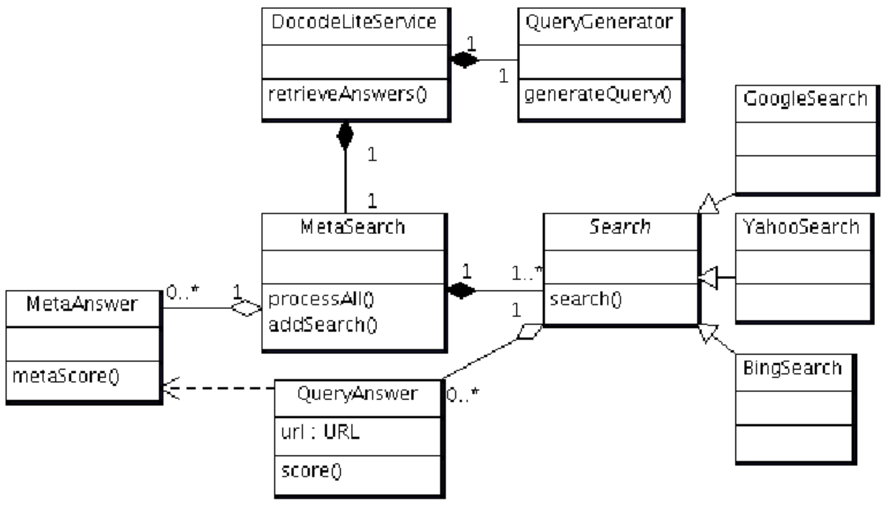

Fig. 2. Class diagram for DOCODE-lite software architecture

[1] Unified Modeling Language: `http://www.uml.org/`

4 Experiments and Results

In present section, all parameters used in this research are introduced, for which different information retrieval options are discussed. Finally, the evaluation mechanism from which the architecture can be evaluated is presented.

According to the previously described architecture, a prototype was implemented. It was developed in the Java programming language (`JDK 1.6.0`), providing a JavaServer Pages (JSP) user interface, mounted in `Apache Tomcat 6.0` Servlet Container. The JSP user interface includes a text area for the input text, and a relevance FeedBack button for each result responded, which allows the user to evaluate the results' relevance. The query generation algorithm was implemented with the `Apache Lucene (2.9.0)` Framework using a term frequency weighting approach and a stopwords' list.

The input area allows a user to insert a text without length constraints. However, If the whole document is considered as input, this could increase the number of non relevant pages retrieved, because of the randomized query generation process. Therefore, we recommend to use a single paragraph instead of a whole document. Since it is a self-contained independent information unit.

A database model was designed to store relevant information for process analysis (Parameters, Input Paragraph, Search Engines, MetaAnswers, and QueryAnswers) and its results (Ranking, Scores, Relevance Feedback, and URLs), presented by the usage of the system and deployed in `MySQL 5.1.37`[2].

4.1 Experimental Setup

A hand-crafted set of paragraphs were extracted from a sample of web documents. For this, a total number of 160 paragraphs were selected from different Spanish written web sites. The respective URL was stored and classified into three different document types: bibliographic documents or school essays (type 1), blog entries or personal web pages (type 2), and news (type 3).

After this paragraphs were sent to the system as input. The top 15 answers from each paragraph search were manually reviewed and classified as relevant or non relevant results. The criteria to label an answer as relevant was defined that the retrieved document must contain the given paragraph exactly[3].

The table 1 shows the numbers of paragraphs, generated queries and retrieved documents per label.

The selected search engines for the experiments were Google, Yahoo!, and Bing. The parameters used for each search engine for the query generation and the ranking procedures, as well as their formal estimation have been intentionally omitted.

[2] This database is freely available to download in `http://dcc.uchile.cl/~fbravo/docode/dbcorpus.sql`

[3] This corpus can be downloaded from `http://dcc.uchile.cl/~fbravo/docode/corpus.xml`

Table 1. Number of paragraphs, queries, and retrieved documents for each type, used in the evaluation procedure

-	Type 1	Type 2	Type 3	All Types
Paragraphs	77	53	30	160
Generated Queries	539	371	210	1120
Retrieved Documents	1155	795	450	2400

4.2 Evaluation Criteria

The goal of this experiment is to measure the effectiveness of the architecture to satisfy user information needs. In this case, those needs are related with the document similarity retrieval problem. The results' relevance measuring process was realized by inspection.

The performance measure used in this work is *the precision at k*, whose composition is presented in equation 2:

$$\textit{precision at } (k) = \frac{\text{relevant retrieved in the top } k \text{ results}}{\text{documents retrieved in the top } k \text{ results}} \tag{2}$$

4.3 Results and Discussion

The table 2 shows the *precision at k* for results retrieved by the whole set of paragraphs associated to their document types. Is easy to see, that the *precision at k* differs with each type of documents.

Table 2. *Precision at k* for each type of documents for the first 10 ranks

k	Precision All	Precision Type 1	Precision Type 2	Precision Type 3
1	0,869	0,922	0,849	0,767
2	0,709	0,727	0,689	0,700
3	0,606	0,602	0,591	0,644
4	0,530	0,526	0,509	0,575
5	0,469	0,465	0,457	0,500
6	0,423	0,416	0,415	0,456
7	0,385	0,375	0,375	0,429
8	0,358	0,344	0,356	0,396
9	0,331	0,317	0,335	0,359
10	0,309	0,295	0,313	0,340

Firstly, type 1 has higher precision at the first values of k than the other types. This is because bibliographic documents are often founded in popular collections like Wikipedia, which are indexed by most of web search engines and usually ranked on top. In this case, a web document will appear as result for most of generated queries. Secondly, type 2 documents are not as popular as type 1. In this case, blog entries or personal pages are hardly indexed by all web search engines. Finally, we can observe a lower precision for the first ranked results of type 3 documents. However, a slow decayment of the *precision at k* is presented. That is because news are repeated in many different web pages, like RSS feeds aggregators, so is possible to find a high number of relevant results lower ranked documents.

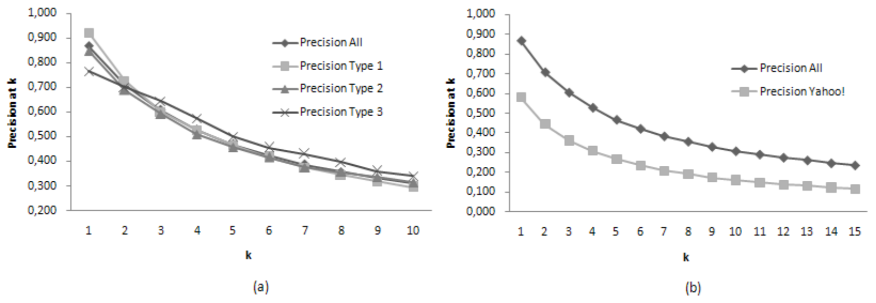

Fig. 3. In (a) the *precision at k* for different types of documents for the first 10 ranks is presented. In (b), the *precision at k* for all types evaluated with all search engines is compared against a single search engine.

As shown in Fig. 3 (b), the *precision at k* for all search engines outperforms a single one for the retrieval of all types of documents. This fact could be explained because the union of search engines gives a better coverage of the web, given that their intersection can be considered as low [2]. In this case, the overall evaluation of a search engine was compared against results retrieved by a single search engine, showing that the meta-search process enhances the number of relevant retrieved documents in comparison with the single search engine.

5 Conclusions

The effectiveness of the proposed SOA distribution of the meta-search engine by using an object oriented architecture was successfully tested. Experimental results showed that proposed architecture (DOCODE-lite) is able to satisfy the document similarity retrieval problem. Likewise, results shows that the developed meta-search model improved significantly the retrieval capacity than a single search engine, contributing to the reception of this work.

In this work, as well as previous proposed methods for Web document similarity retrieval, such as [4,12], are based in randomized query generation and meta-search algorithms. However, in this case the scoring function uses the local ranks and search engine reliability instead of downloading the retrieved documents [4], or text processing algorithms over the results snippets [12], aiming towards the minimization of the processing time a lite-weighted architecture.

Presented architecture can be applied for plagiarism detection, document impact analyzer, related ideas retrieval, among other document retrieval problems. For example, in plagiarism detection, the proposed model will retrieve a similar document to the suspicious one, allowing the user to determine its authenticity. Furthermore, plagiarized documents uses the same words than the original source, usually changing the order of terms, for which proposed model will convert suspicious document into a set queries that could retrieve original source document. Likewise, for document impact analysis, the number of retrieved documents could be increased in order to review the frequency of appearance of a quote paragraph.

Many plagiarism detection tools like *eTBLAST*[4] [5] compare the suspicious documents over specific topic databases. Taking use of DOCODE-lite in the search task, the search scope could be expanded, considering large document collections. However, plagiarism detection tools like *duplichecker*[5] allows to search using commercial search engines, but don't combine them in one single search. Furthermore the fingerprinting process is deterministic, converting each sentence directly into a query. Our probabilistic approach allow us to detect more sophisticated plagiarism techniques. Finally *Plagium*[6] seems to be using a randomized query generation process, but only uses the Yahoo! search engine.

6 Future Work

As future work, the proposed architecture could be extended into a large scale processing environment. This fact leads to complex distributed system architecture and parallel processing problems that needs to be taken into consideration. Also, further development in terms of the relevance feedback from users could be considered in an active learning schema over the scoring parameters, in order to enhance the effectiveness of the system.

It is important to consider, that DOCODE-Lite is a first approach for a document plagiarism detection system. The proposed model is focused on solving the document similarity retrieval problem for a given paragraph. A complete plagiarism detector system should also allow working with large collection of documents. The system should be able to identify potential plagiarized paragraphs in documents using for example text mining algorithms. Those paragraphs could consume the DOCODE-lite Web Service in order to retrieve a collection of source candidates documents. Afterwards more sophisticated plagiarism detection strategies must be used. Finally a plagiarism detection system that consumes the DOCODE-lite service will have a great advantage against other similar tools. Because using DOCODE-lite Web Service gives the best coverage from the Web at combing the effectiveness of commercial Web search engines.

[4] `http://etest.vbi.vt.edu/etblast3/`

[5] `http://www.duplichecker.com`

[6] `http://www.plagium.com/`

Acknowledgment

Authors would like to thank Marcos Himmer for his useful comments in the system's architecture and interface development. Also, we would like to thank continuous support of "Instituto Sistemas Complejos de Ingeniería" (ICM: P-05-004- F, CONICYT: FBO16; `www.sistemasdeingenieria.cl`); FONDEF project (DO8I-1015) entitled, DOCODE: Document Copy Detection (`www.docode.cl`); and the Web Intelligence Research Group (`wi.dii.uchile.cl`).

References

1. Aslam, J.A., Montague, M.: Models for metasearch. In: SIGIR 2001: Proceedings of the 24th Aannual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 276–284. ACM, New York (2001)
2. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston (1999)
3. Cormack, G.V., Palmer, C.R., Van Biesbrouck, M., Clarke, C.L.A.: Deriving very short queries for high precision and recall (multitext experiments for trec-7) (1998)
4. Pereira Jr., Á.R., Ziviani, N.: Retrieving similar documents from the web. J. Web Eng. 2(4), 247–261 (2004)
5. Lewis, J., Ossowski, S., Hicks, J., Errami, M., Garner, H.R.: Text similarity: an alternative way to search medline. Bioinformatics 22(18), 2298–2304 (2006)
6. Rasolofo, Y., Abbaci, F., Savoy, J.: Approaches to collection selection and results merging for distributed information retrieval. In: CIKM 2001: Proceedings of the Tenth International Conference on Information and Knowledge Management, pp. 191–198. ACM, New York (2001)
7. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. ACM Commun. 18(11), 613–620 (1975)
8. Selberg, E., Etzioni, O.: The metacrawler architecture for resource aggregation on the web. IEEE Expert, 11–14 (January-February 1997)
9. Velasquez, J.D., Palade, V.: Adaptive Web Sites: A Knowledge Extraction from Web Data Approach. IOS Press, Amsterdam (2008)
10. Wu, Z., Meng, W., Yu, C., Li, Z.: Towards a highly-scalable and effective metasearch engine. In: WWW 2001: Proceedings of the 10th International Conference on World Wide Web, pp. 386–395. ACM, New York (2001)
11. Wu, Z., Raghavan, V., Qian, H., Rama, V., Meng, W., He, H., Yu, C.: Towards automatic incorporation of search engines into a large-scale metasearch engine. In: WI 2003: Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence, Washington, DC, USA, p. 658. IEEE Computer Society, Los Alamitos (2003)
12. Zaka, B.: Empowering plagiarism detection with a web services enabled collaborative network. Journal of Information Science and Engineering 25(5), 1391–1403 (2009)