SimSeerX: A Similar Document Search Engine

Kyle Williams[‡], Jian Wu[‡], C. Lee Giles[†‡]
[‡]Information Sciences and Technology, [†]Computer Science and Engineering
The Pennsylvania State University, University Park, PA 16802, USA
kwilliams@psu.edu, jxw394@ist.psu.edu, giles@ist.psu.edu

ABSTRACT

The need to find similar documents occurs in many settings, such as in plagiarism detection or research paper recommendation. Manually constructing queries to find similar documents may be overly complex, thus motivating the use of whole documents as queries. This paper introduces SimSeerX, a search engine for similar document retrieval that receives whole documents as queries and returns a ranked list of similar documents. Key to the design of SimSeerX is that is able to work with multiple similarity functions and document collections. We present the architecture and interface of SimSeerX, show its applicability with 3 different similarity functions and demonstrate its scalability on a collection of 3.5 million academic documents.

Categories and Subject Descriptors

H.3.3 [**Information Storage And Retrieval**]: Information Search and Retrieval; I.7.5 [**Document and Text Processing**]: Document Capture—*Document Analysis*

General Terms

Design, Experimentation

Keywords

Similarity search; document similarity; query by document

1. INTRODUCTION

Search engines have simplified the way in which information is discovered. By submitting queries that capture an information need, relevant information can efficiently be found on the Web and in document collections. In the majority of cases, these queries are constructed based on keywords that are related to a topic of interest; however, a difficulty often arises in constructing queries for complex information needs. To address this problem, search methodologies such as content-based information retrieval have been developed where the queries are based on the *content* of digital objects.

DocEng'14, September 16–19, 2014, Fort Collins, Colorado, USA.

ACM 978-1-4503-2949-1/14/09 ...$15.00.
http://dx.doi.org/10.1145/2644866.2644895.

Similar document search is a type of content-based information retrieval where the goal is to find documents that are *similar* to a query document. The definition of document similarity depends on the application. For instance, document similarity might be defined as documents that are about related topics or have overlapping text.

Traditionally, to find similar files, users construct queries that are submitted to an information retrieval system; however, as already mentioned, in many cases it may not be obvious to the user how they should construct queries from a document in order to retrieve the type of similar documents that they are looking for or, when the user does know how to construct the query, the complexity of actually constructing the query may be a limiting factor. A content-based search method known as *Query by Document (QBD)* attempts to overcome these problems by allowing users to submit whole documents as queries to an information retrieval system which then returns a ranked list of similar documents based on a pre-defined similarity function [12].

In this paper, we present SimSeerX[1], a similar document search engine framework. SimSeerX can be used to find similar files in a collection of documents and can support many different types of similarity scoring functions and document collections. SimSeerX incorporates a pseudo relevance feedback mechanism in the form of recursive search whereby the results of a search are used to formulate queries for additional searches. The recursive search may return additional results that were not retrieved by the original query and all results can be combined and ranked. SimSeerX has applications in a number of domains, such as plagiarism detection [4], near duplicate detection [13] and research paper recommendation. It can also be used to compare and evaluate new similarity functions that can be plugged into the system.

2. RELATED WORK

There have been many systems designed to retrieve similar documents with most focusing on specific use cases. An early system involved retrieving similar documents from the Web [10]. Signatures based on representative sentences of query documents are submitted to a search engine and the returned results are labelled as candidate documents. The documents are then compared to the query document using shingles and Patricia trees. Govindaraju et al. [5] extract key phrases from documents and submit them as queries to a search engine. The extracted key phrases are based on co-occurrences of words and the results that are returned are

[1]http://simseerx.ist.psu.edu

scored based on the Jaccard similarity of the keywords and key phrases of the query document and the returned results. Dasdan et al. [3] designed a similarity system based on querying a search engine interface. Queries are constructed based on the least frequent terms in the document and the similarity of the returned documents is calculated based on shingle similarity.

Similar document search can also be applied to different types of documents. For instance, Pera et al. developed a book recommender for K-12 users [9]. In addition to traditional content similarity, Pera et al. also considered the readability of books. Weng et al [12] formalize query by document as a document decomposition problem where the representation of a document is based on decomposing it into a feature vector as well as other information. Topics are used as features, which are supplemented with keywords that are used for the ranking. While SimSeerX does not use the same features as this approach, it does make use of the document representation for indexing and retrieval.

3. THE SIMSEERX SYSTEM

The SimSeerX system is made up of the user interface, the index subsystem and the query subsystem.

3.1 User Interface

As previously stated, SimSeerX supports multiple similarity functions and thus the first decision a user can make when using SimSeerX is which similarity function they wish to use when searching. Currently, SimSeerX supports three different similarity functions but it is possible to add more. When a similarity function has been selected, options specific to that similarity function appear. These options include search parameters as well as the ranking method that should be used. The user also has the option to select the recursive depth at which to search, which is referred to as the number of hops. Lastly, the user submits a file.

Figure 1 shows the results page that is displayed after the user conducts the search. The results show metadata that is automatically extracted from the query document, options to re-run the search with different options and a ranked list of results.

3.2 Document Representation

The ability of SimSeerX to work with multiple similarity signatures is based on the document representation. In SimSeerX, each document is decomposed into a series of components [12]. Each document d is represented by $d = \mathbb{X}+m+\epsilon$, where $\mathbb{X}$ is a set of document signatures, m is document metadata, and ϵ is other document information. X should be constructed in such a way that the following conditions are satisfied as best possible: if documents A and B are similar (a binary judgment), then (1) $|\mathbb{X}_A \cap \mathbb{X}_B| \geq 1$; and, by extension, if documents A and B are not similar then, ideally, (2) $|\mathbb{X}_A \cap \mathbb{X}_B| = \varnothing$. (1) serves to ensure that similar documents can be retrieved based on their signatures alone, whereas (2) serves to minimize the number of comparisons made between non-similar documents. SimSeerX can be used with any similarity function that allows for documents to be decomposed this way since the document signatures $\mathbb{X}$ can be used for indexing and retrieval.

Using this document representation, the indexing, querying, and ranking processes are then described by:

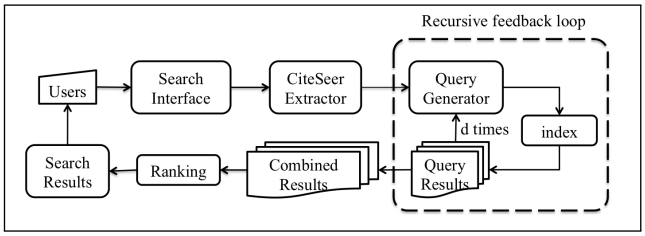

Figure 2: The SimSeerX workflow.

Indexing. Index every document in C in a standard information retrieval index I: $\forall d \in C, index(X, m, \epsilon)$ in I.

Querying. For a query document $d_q = X + m + \epsilon$, retrieve the set of candidate similar documents S from the index using the document signatures as queries: $S = query(X_{d_q}, I)$.

Ranking. Score each document d in S using a scoring function $sim(\cdot)$ that calculates the similarity between d and d_q: $\forall d \in S, sim(d, d_q)$, where $sim(\cdot)$ might take into consideration any of X, m, ϵ. Return S sorted by score in descending order.

3.3 Indexing Subsystem

The indexing subsystem indexes each document in order to allow for similar files to be retrieved based on document signatures. Each document to be indexed is preprocessed, which involves tokenization, punctuation removal, conversion to lower case and possibly stemming. Signatures are then constructed for each document and indexed in a Lucene index, along with metadata and other information that may be necessary to calculate the full similarity of the documents with a given query document. The only difference between the type of indexing that takes place in most search engines and the indexing that takes place here, is that here the focus is on indexing document signatures for retrieval.

3.4 Query Subsystem

The query subsystem encapsulates the SimSeerX workflow, which is shown in Figure 2. The subsystem receives a document as a query and returns a ranked list of similar documents. Querying involves document submission, preprocessing, query signature construction, candidate retrieval, and re-ranking. A document undergoes automatic information extraction when it is submitted, which might involve text extraction if the document is a PDF as well as metadata extraction. SimSeerX makes use of CiteSeerExtractor [14] to perform text and metadata extraction. CiteSeerExtractor provides an API that performs various types of extraction, such as text, header, and citation extraction.

Once the text and metadata have been extracted from the document, queries are automatically constructed and used to query the index along with any query parameters that the user might specify. The Solr instance that SimSeerX is based on has been modified to support custom ranking functions for different types of similarity queries (discussed in Section 4). Thus, documents are retrieved based on their signatures and ranked using an appropriate ranking function. SimSeerX also includes a general ranking function whereby each result can be ranked by its cosine similarity with the original query document. Finally, the ranked results are returned to the user.

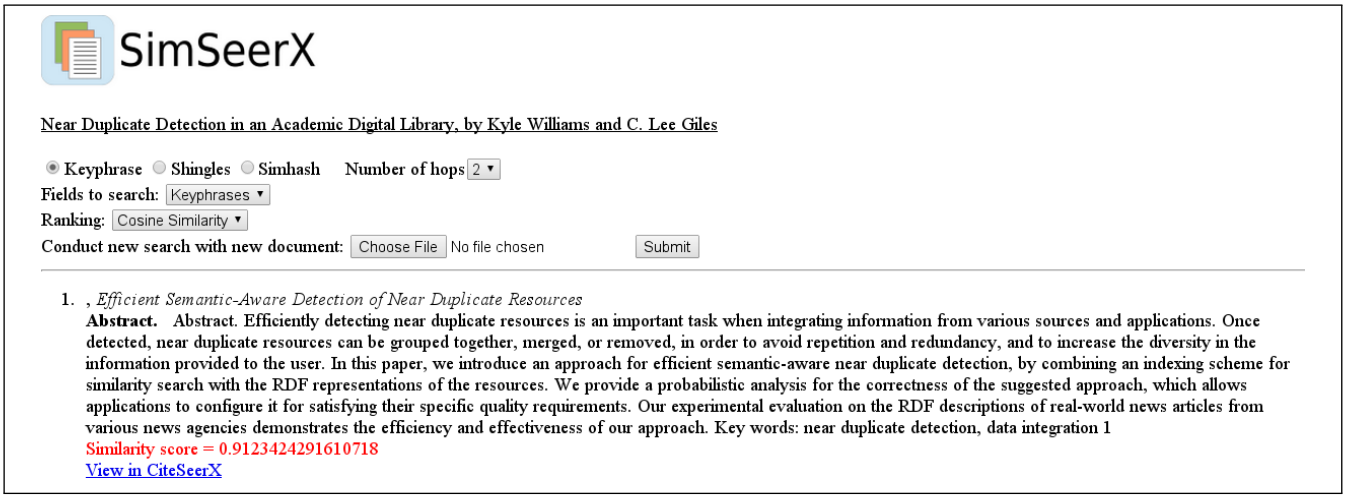

Figure 1: The SimSeerX results page.

4. SIMILARITY IN SIMSEERX

SimSeerX currently implements three similarity functions that can be used to find similar documents: key phrase-based similarity, shingle similarity and simhash similarity.

4.1 Key Phrase Similarity

Key phrases provide high level descriptions of documents and can be used for efficient document retrieval and similarity measures [11]. To generate key phrases, the Maui tool [8] is used and trained on the SemEval 2010 dataset. For each document that is indexed by SimSeerX, the top 10 key phrases are extracted and indexed alongside the document. At search time, the top 10 key phrases are extracted from the query document in the same way. These key phrases can be used to query either the key phrases of the indexed documents or the full text of the indexed documents. In both cases, the query is a phrase and the resulting documents are ranked using the standard Lucene ranking function.

4.2 Shingle Similarity

Shingles are sequences of tokens (words) in a document and were first introduced as a means for calculating the similarity of documents [1]. For a shingle length w, i.e. a sequence of w words, which is also known as a w-shingle, the similarity between two documents can be calculated in terms of the number of shingles that they have in common. Given the w-shingles of two documents, d_1 and d_2, the resemblance R of the two documents is given by:

$$R(d_1, d_2) = \frac{S(d_1) \cap S(d_2)}{S(d_1) \cup S(d_2)}, \quad (1)$$

where $S(d)$ is the set of shingles in document d.

The actual similarity measure is not calculated based on the shingles themselves, but rather on a hash of the shingles. Because it is computationally expensive to calculate the similarity of two documents based on all of their shingles, a fixed number of shingles, which is known as a sketch, are selected from each document and the similarity of the two documents is estimated based on the Jaccard similarity of the shingles in their sketch. The calculation of the sketch of a document is done using a technique known as *minhash*. For a length w, the sequences of tokens whose length is w represent the shingles. Then, each shingle, which is represented by the value of its 64-bit Rabin fingerprint is hashed with h hash functions and track is kept of the minimum hash value found for each hash function. The sketch of each document is then represented by its set of h minimum hash values and the resemblance of two documents can be estimated based on the extent to which their sketches overlap [6].

To select shingles we use 84 hash functions in the form of $h(x) = (Ax + B) \mod p$, where x is the Rabin fingerprint of the shingle, p is a large prime, which we set to $2^{64} - 59$, and A and B are random integers in the range $[1, p]$.

During indexing, the sketch of each document is calculated and the shingles of that sketch are indexed. During retrieval, the sketch of the query document is calculated and used to retrieve documents that have a shingle in common with the query document with the ranking of each document calculated by the Jaccard similarity of their sketches.

4.3 Simhash Similarity

Simhash [2] is a state of the art near duplicate detection algorithm. Near duplicate detection is a natural application of SimSeerX, where the goal is to retrieve near duplicates to a query document. For each document, the simhash is calculated as follows. A fingerprint V of size f represents each document. Each token (word) t that appears in a document is hashed using the 64-bit Jenkins hash function. Then, if the i-th bit of the hash F_t of token t is 1, then the i-th bit of V is increased by 1. Conversely, if the i-th bit of F_t is 0, then V is decreased by 1. Once all tokens have been processed, V contains both positive and negative numbers at each of its f locations that are the result of the sums of the weights of all of the tokens. Each of the f locations in V is thresholded to either 0 or 1 to create a final bit-hash for V.

In simhash two documents are considered as being near duplicates if their simhash Hamming distances is at most k. To retrieve documents that have a Hamming distance of at most k, we apply an efficient algorithm [7] that has been modified to work with an inverted index as follows. Once the simhash for each document has been calculated, it is partitioned into $k + 1$ sub-hashes and these sub-hashes are

Table 1: Average time taken (in seconds) using 10 query documents.

Data Size	Time (cold/cached)
∼3.5 million	4.74 (0.52)/1.70 (0.27)
2.5 million	4.26 (0.49)/1.88 (0.25)
1.5 million	4.23 (0.61)/1.89 (0.32)

indexed. At query time, the simhash of the query document is also partitioned into $k + 1$ sub-hashes that are used to query the index and retrieve documents that have at least one sub-hash in common with the query document. This method guarantees that all documents whose simhashes differ from the query simhash by k bits will be found since, in the worst case, the differing bits can only occur in k of the sub-hashes and the final similarity between two documents can then be calculated based on the Hamming distance of their full simhashes.

We have briefly introduced the 3 similarity functions that are currently supported by SimSeerX. It is relatively simple to implement additional similarity functions as long as a document can be decomposed into a set of signatures.

5. SCALABILITY

To evaluate the scalability of SimSeerX, a snapshot of the CiteSeer$^{\chi}$ dataset containing 3,577,543 documents was used. Evaluation is performed on subsets of this collection of size S with $S = 1.5M, 2.5M, \sim 3.5M$ (M = million). The time taken to search without recursion was measured using key phrases to search over key phrases. Two results are reported: the search time for a cold start whereby the memory buffers are flushed and the Solr instance restarted before every search run; and the search time for a cached search whereby the search is repeated after it completes the first time. The time reported is the wall time to perform search excluding document upload, extraction and result rendering.

Table 1 shows the mean time and standard deviation for both cold start and cached start search runs, where each search run involves searching with 10 papers from the CiteSeer$^{\chi}$ collection. As can be seen from the table, the time taken to conduct the search is relatively consistent regardless of the size of the indexed collection thus suggesting that the system can scale well.

6. CONCLUSIONS

We present SimSeerX, a query by document search engine for finding similar documents. SimSeerX was designed so as to allow users to submit full documents as queries in order to find which documents in a collection are most similar according to a predefined similarity function. The overall design of SimSeerX is a modular architecture with various pluggable similarity functions. The key difference between SimSeerX and existing query by document systems is that, while other work has tended to focus on specific query and ranking methods, SimSeerX as a framework provides a generic architecture for query by document for any similarity scoring function. Currently, SimSeerX requires users to submit queries for each similarity function separately. Thus, a future feature could involve displaying the results from different similarity functions side by side or combining them into a single ranked list.

Acknowledgments

We gratefully acknowledge partial support by the National Science Foundation under Grant No. 1143921.

7. REFERENCES

[1] A. Broder, S. Glassman, M. Manasse, and G. Zweig. Syntactic clustering of the Web. *Computer Networks and ISDN Systems*, 29(8-13):1157–1166, 1997.

[2] M. Charikar. Similarity estimation techniques from rounding algorithms. In *ACM symposium on Theory of computing*, pages 380–388, 2002.

[3] A. Dasdan, P. D'Alberto, S. Kolay, and C. Drome. Automatic retrieval of similar content using search engine query interface. In *Proceeding of the 18th ACM conference on Information and knowledge management*, pages 701–710, 2009.

[4] B. Gipp and N. Meuschke. Citation pattern matching algorithms for citation-based plagiarism detection. In *Proceedings of the 11th ACM symposium on Document engineering*, pages 249–258, 2011.

[5] V. Govindaraju and K. Ramanathan. Similar Document Search and Recommendation. *Journal of Emerging Technologies in Web Intelligence*, 4(1):84–93, 2012.

[6] M. Henzinger. Finding near-duplicate web pages. In *Proceedings of the 29th international ACM SIGIR conference on research and development in information retrieval*, pages 284–291, 2006.

[7] G. Manku, A. Jain, and A. D. Sarma. Detecting near-duplicates for web crawling. *Proceedings of the 16th international conference on World Wide Web*, pages 141–149, 2007.

[8] O. Medelyan, V. Perrone, and I. H. Witten. Subject Metadata Support Powered by Maui. In *Proceeding of the 10th annual international ACM/IEEE joint conference on Digital libraries*, pages 407–408, 2010.

[9] M. Pera and Y. Ng. Brek12: A book recommender for k-12 users. In *Proceedings of the 35th international ACM SIGIR conference on research and development in information retrieval*, pages 1037–1038, 2012.

[10] A. Pereira and N. Ziviani. Retrieving similar documents from the web. *Journal of Web Engineering*, 2(4):247–261, 2004.

[11] R. Shams and R. E. Mercer. Investigating Keyphrase Indexing with Text Denoising. In *Proceeding of the 12th annual international ACM/IEEE joint conference on Digital libraries*, pages 263–266, 2012.

[12] L. Weng, Z. Li, R. Cai, Y. Zhang, Y. Zhou, L. T. Yang, and L. Zhang. Query by document via a decomposition-based two-level retrieval approach. In *Proceedings of the 34th international ACM SIGIR conference on research and development in information retrieval*, pages 505–514, 2011.

[13] K. Williams and C. L. Giles. Near duplicate detection in an academic digital library. In *Proceedings of the 2013 ACM symposium on Document engineering*, pages 91–94, 2013.

[14] K. Williams, L. Li, M. Khabsa, J. Wu, P. C. Shih, and C. L. Giles. A Web Service for Scholarly Big Data Information Extraction. In *Proceedings of the IEEE International Conference on Web Services*, 2014.