SimSeerX:
A Similar Document Search Engine

Kyle Williams, Jian Wu and C. Lee Giles

Information Sciences and Technology
Computer Science and Engineering
The Pennsylvania State University

Introduction

- The need to find similar files occurs in many situations
  - Plagiarism detection
  - Research paper recommendation
  - Near duplicate detection
- Definition of “similar” depends on domain and problem
  - Text similarity
  - Semantic similarity
  - Data similarity
  - Citation similarity

Finding Similar Files

- Generally, to find similar files users construct queries and submit them to a search engine
  - Select a sentence as a query
  - Identify the main ideas and make them a query
- It may be difficult to do this for complex information needs
  - So allow the user to submit the whole document as a query
  - Automatically extracts and construct query from document
  - Query by Document (QBD)

SimSeerX

- A similar document search engine and architecture
- Supports multiple notions of document similarity
- Supports different types of collections
- Incorporates pseudo-relevance feedback
- Many applications
  - Plagiarism detection
  - Research paper recommendation
  - Near duplicate detection

User Interface

- Allows user to select similarity function, upload a file and select number of recursive hops

Results Page

SimSeerX: A Similar Document Search Engine, by Kyle Williams, Jian Wu and C. Lee Giles

Keyphrase Shingles Simhash Number of hops 2

Fields to search: Keyphrases

Ranking: Cosine Similarity

Conduct new search with new document: Choose File No file chosen Submit

---

1. **Andrew Turpin**, *ABSTRACT Fast Generation of Result Snippets in Web Search*
   Abstract. The presentation of query biased document snippets as part of results pages presented by search er algorithms and data structures required as part of a search engine to allow efficient generation of query biased reduces snippet generation time by 58 % over a baseline using the zlib compression library. These experiment generating snippets, and so caching documents in RAM is essential for a fast snippet generation process. Usin caches. Finally we propose and analyse document reordering and compaction, revealing a scheme that increas This scheme effectively doubles the number of documents that can fit in a fixed size cache.
   Similarity score = 0.9118895530700684
   View in CiteSeerX

2. **Fabrizio Silvestri**, *LSDS-IR'10 8 th Workshop on Large-Scale Distributed Systems for Information Retrieval*
   Abstract.
   Similarity score = 0.911067545413971
   View in CiteSeerX

Document Representation

- SimSeerX works with any similarity function where a document d can be represented by:

$$d = \mathbb{X} + m + \epsilon$$

X: document/similarity features

m: metadata

□: additional information

Representation Requirements

Feature set X should be structured in such a way that the following conditions are satisfied as best possible:

If documents A and B are similar then:

$$\left|\mathbb{X}_A \cap \mathbb{X}_B\right| \geq 1$$

Similarly, if A and B are not similar then:

$$\left|\mathbb{X}_A \cap \mathbb{X}_B\right| = \varnothing$$

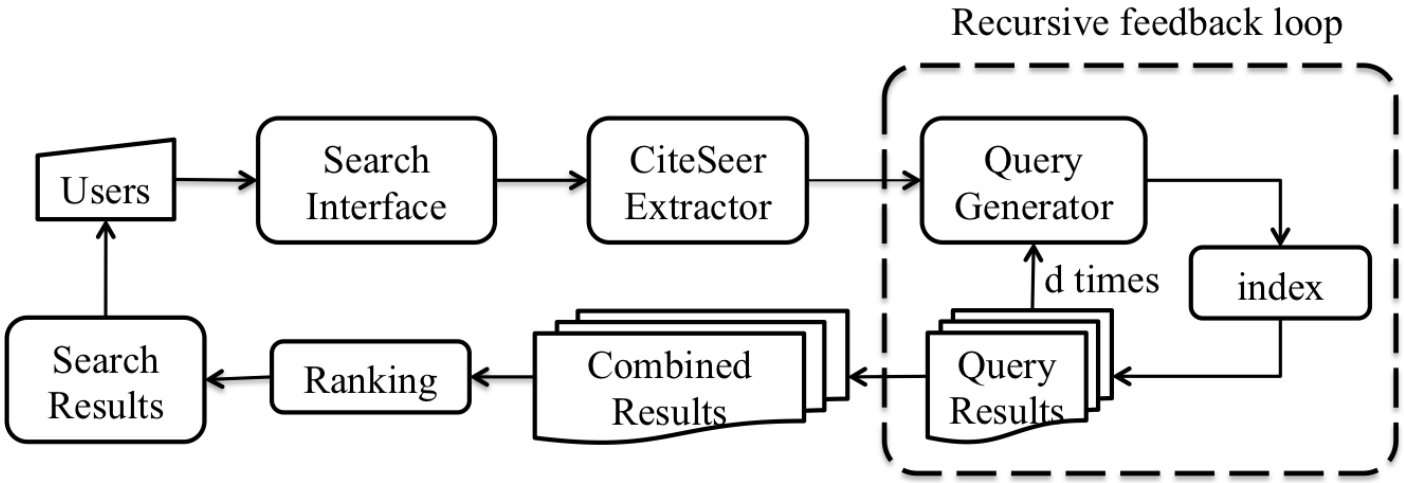
IR Operations

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.

Query Subsystem



- Users submit documents
- Metadata and queries are automatically extracted and generated
- Recursive pseudo-feedback search is used
- All results are combined and ranked

Pseudo-Relevance Feedback

- Sometimes a similar file may have no features in common with the query document
  - But it may have features in common with a result document $\left|\mathbb{X}_A \cap \mathbb{X}_B\right| = \varnothing$

- The results of a search are used to conduct new searches in a recursive manner
  - We refer to the number of times as the *depth*
  - The depth can be selected by the user at query time
  - A depth beyond 2 adds little value
  - Make sure not to search with the same document

Similarity in SimSeerX

- Key phrases
  - Documents and queries are represented by their keyphrases
- Shingles
  - Documents are represented by their sequences of ***w*** words (shingles). Queries are set of shingles in query document.
- Simhash
  - Documents are represented by their ***simhash***, a binary hash which is partitioned into sub-hashes indexing and for retrieval
  - Queries are sub-hashes

Ranking in SimSeerX

- Generally, each similarity function implements its own ranking
  - Lucene similarity for keyphrases
  - Jaccard similarity for shingles
  - Hamming distance for simhash
- SimSeerX can also incorporate generic ranking functions
  - Results can be re-scored after retrieval
  - Currently cosine similarity is implemented

Scalability

- Tested the scalability of SimSeerX on subsets of the CiteSeerX collection
  - 1.5M, 2.5M, 3.5M documents (M = Million)

- Measured time taken to search using key-phrases for 10 query documents
  - Cold start and cached search

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)

Current Activity/Future Development

- Currently, search is performed for each similarity function independently
  - Conduct searches simultaneously and combine/merge results
- Allow users to upload/create their own collections
- Incorporate new similarity functions
  - Semantic similarity, data similarity, etc.

Thanks

- Partial support by the National Science Foundation

SimSeerX

http://simseerx.ist.psu.edu