Governors State University
OPUS Open Portal to University Scholarship

All Capstone Projects   Student Capstone Projects

Fall 2015

A Survey to Fix the Threshold and Implementation for Detecting Duplicate Web Documents

Manojreddy Bhimireddy
Governors State University

Krishna Pavan Gandi
Governors State University

Reuven Hicks
Governors State University

Bhargav Roy Veeramachaneni
Governors State University

Table of Contents

1 Project Description

1.1 Project Abstract

The drastic development in the information accessible on the World Wide Web has made the employment of automated tools to locate the information resources of interest, and for tracking and analyzing the same a certainty. Web Mining is the branch of data mining that deals with the analysis of World Wide Web. The concepts from various areas such as Data Mining, Internet technology and World Wide Web, and recently, Semantic Web can be said as the origin of web mining. Web mining can be defined as the procedure of determining hidden yet potentially beneficial knowledge from the data accessible in the web. Web mining comprise the sub areas: web content mining, web structure mining, and web usage mining. Web content mining is the process of mining knowledge from the web pages besides other web objects. The process of mining knowledge about the link structure linking web pages and some other web objects is defined as Web structure mining. Web usage mining is defined as the process of mining the usage patterns created by the users accessing the web pages.

The search engine technology has led to the development of World Wide. The search engines are the chief gateways for access of information in the web. The ability to locate contents of particular interest amidst a huge heap has turned businesses beneficial and productive. The search engines respond to the queries by employing the process of web crawling that populates an indexed repository of web pages. The programs construct a confined repository of the segment of the web that they visit by navigating the web graph and retrieving pages.

There are two main types of crawling, namely, Generic and Focused crawling. Generic crawlers crawls documents and links of diverse topics. Focused crawlers limit the number of pages with the aid of some prior obtained specialized knowledge. The systems that index, mine, and otherwise analyze pages (such as, the search engines) are provided with inputs from the repositories of web pages built by the web crawlers. The drastic development of the Internet and the growing necessity to incorporate heterogeneous data is accompanied by the issue of the existence of near duplicate data. Even if the near duplicate data don’t exhibit bit wise identical nature they are remarkably similar. The duplicate and near duplicate web pages either increase the index storage space or slow down or increase the serving costs which annoy the users, thus causing huge problems for the web search engines. Hence it is inevitable to design algorithms to detect such pages.

1.2 Competitive Information

This work is the enhanced version of our previous work with experimental analysis [39]. The main objective of our research is the development of a novel and efficient approach for detection of near duplicates in web documents. Firstly, document parsing is employed to preprocess the crawled web pages, in which the HTML tags and java scripts present in the web documents are removed. Secondly, the common words or stop words present in the crawled pages are removed. Subsequently, the keywords are obtained by employing the stemming algorithm that filters the affixes present in the crawled documents. Lastly, the extracted keywords are utilized in the calculation of the similarity score between two documents. The documents are considered as near duplicates if its similarity scores are lesser than a predefined threshold value. We have verified that the proposed algorithm outperforms previous ones by conducting an extensive experimental study employing several real datasets.

Every engine relies on a crawler module to provide the grist for its operation (shown on the left in Figure 2). Crawlers are small programs that `browse' the Web on the search engine's behalf, similarly to how a human user would follow links to reach different pages. The programs are given a starting set of URLs, whose pages they retrieve from the Web. The crawlers extract URLs appearing in the retrieved pages, and give this information to the crawler control module. This module determines what links to visit next, and feeds the links to visit back to the crawlers. (Some of the functionality of the crawler control module may be implemented by the crawlers themselves.) The crawlers also pass the retrieved pages into a page repository.

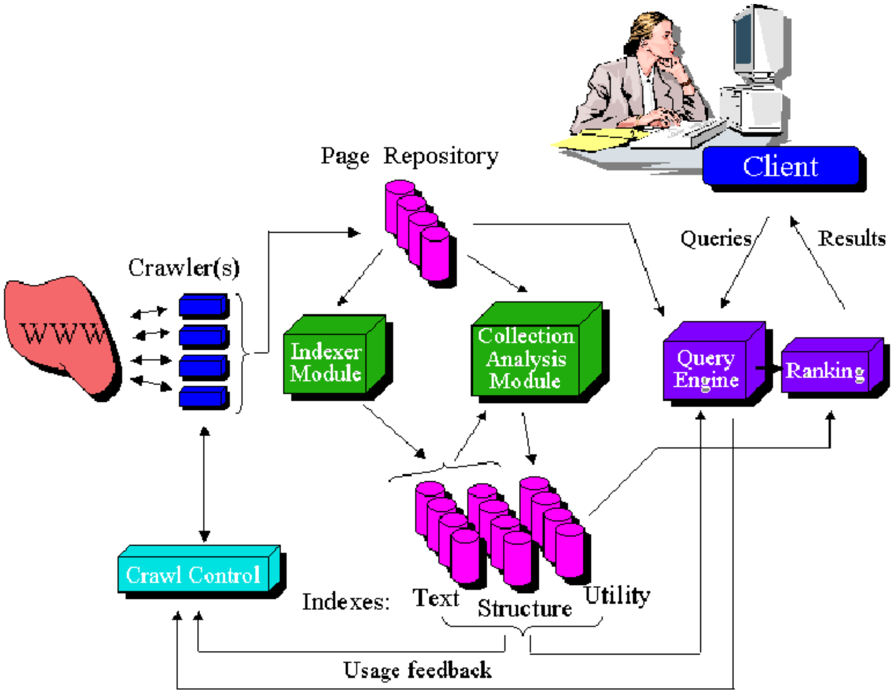


Figure 2

Crawlers continue visiting the Web, until local resources, such as storage, are exhausted. This basic algorithm is modified in many variations that give search engines different levels of coverage or topic bias. For example, crawlers in one engine might be biased to visit as many sites as possible, leaving out the pages that are buried deeply within each site. The crawlers in other engines might specialize on sites in one specific domain, such as governmental pages. The crawl control module is responsible for directing the crawling operation. Once the search engine has been through at least one complete crawling cycle, the crawl control module may be informed by several indexes that were created during the earlier crawl(s). The crawl control module may, for example, use a previous crawl's link graph (the structure index in Figure 2) to decide which links the crawlers should explore, and which links they should ignore. Crawl control may also use feedback from usage patterns to guide the crawling process (connection between the query engine and the crawl control module in Figure 1). Section 2 will examine crawling operations in more detail.

1.3 Relationship to Other Applications/Projects

Our work has been inspired by a number of previous works on duplicate and near duplicate document and web page detection.

Sergey Brin et al. have proposed a system for registering documents and then detecting copies, either complete copies or partial copies. They described algorithms for detection, and metrics required for evaluating detection mechanisms covering accuracy, efficiency and security. They also described a prototype implementation of the service, COPS, and presented experimental results that suggest the service can indeed detect violations of interest.

Andrei Z. Broder et al. have developed an efficient way to determine the syntactic similarity of files and have applied it to every document on the World Wide Web. Using their mechanism, they have built a clustering of all the documents that are syntactically similar. Possible applications include a "Lost and Found" service, filtering the results of Web searches, updating widely distributed web-pages, and identifying violations of intellectual property rights.

Jack G. Conrad et al. have determined the extent and the types of duplication existing in large textual collections. Their research is divided into three parts. Initially they started with a study of the distribution of duplicate types in two broad-ranging news collections consisting of approximately 50 million documents. Then they examined the utility of document signatures in addressing identical or nearly identical duplicate documents and their sensitivity to collection updates. Finally, they have investigated a flexible method of characterizing and comparing documents in order to permit the identification of non-identical duplicates. Their method has produced promising results following an extensive evaluation using a production-based test collection created by domain experts.

1.4 Assumptions and Dependencies

Broadly speaking, duplicate-detection systems have been developed for four types of document collections:

a) Web Documents: Near-duplicate systems have been developed for finding related-pages, for extracting structured data, and for identifying web mirrors.

b) Files in a file system: Manber develops algorithms for near-duplicate detection to reduce storage for files. The Venti file system and the Low-bandwidth file system have similar motivations.

c) E-mails: Kolcz et al identify near-duplicates for spam detection.

d) Domain-specific corpora: Various groups have developed near-duplicate detection systems for legal documents (see Conrad and Schriber), TREC benchmarks, Reuters news articles, and Citeseer data. Our work falls into the first category (Web Documents). We experimented with 8B pages { this is way larger than collection sizes tackled by previous studies: web-clustering by Broder et al (30M URLs in 1997), \related pages" by Dean and Henzinger (180M URLs in 1998), web clustering by Haveliwala et al (35M URLs in 2000).

Each document is viewed as a sequence of tokens. The letters, or words, or lines are taken as tokens. It is assumed that there is a parser program that takes an arbitrary document and reduces it to a canonical sequence of tokens. (Here “canonical” means that any two documents that differ only in formatting or other information that we chose to ignore, for instance punctuation, formatting commands, capitalization, and so on, will be reduced to the same sequence.)

1.5 Future Enhancements

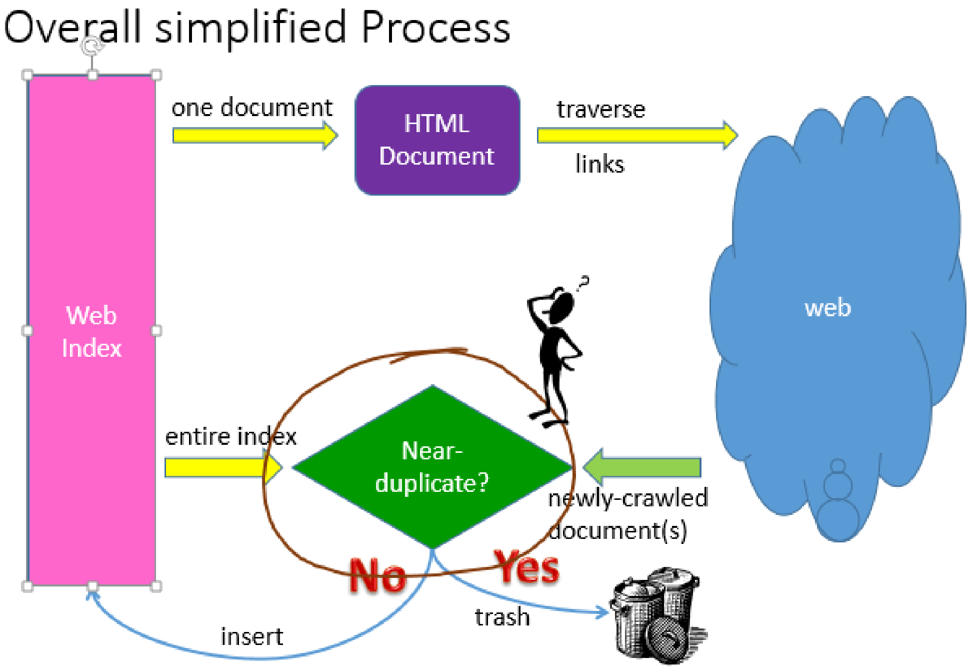
Cooper et al propose identification of phrases using a phrase-detection system and computing a document-vector that includes phrases as terms. They have tested their ideas on a very small collection (tens of thousands). The idea of using phrases also appears in the work of Hammouda and Kamel who build sophisticated indexing techniques for web-clustering. We chose to work with the document vector model; simhash converts document vectors into fingerprints. Augmenting the document vector by other signals (anchor text and connectivity information, for example) might improve the quality of our system. We leave these possibilities as future work.

1.6 *Definitions and Acronyms*

Acronym items should be included here. For each special term supply a definition here.

2 *Technical Description*

2.1 *Project/Application Architecture*



We have presented a novel and efficient approach for the detection of near duplicate web pages in web crawling in this paper.

The near duplicate web pages are detected followed by the storage of crawled web pages in to repositories.

The keywords are extracted from the crawled pages initially and on the basis of the extracted keywords, the similarity score between the two pages is calculated.

The documents are considered as near duplicates if its similarity scores are lesser than a threshold value. Memory for repositories has been reduced and the search engine quality has been improved owing to the detection.

2.1.1 FINGERPRINTING USING SHINGLING

he basic approach for computing resemblance has two aspects: First, resemblance is expressed as a set intersection problem, and econd, the relative size of intersections is evaluated by a process of random sampling that can be done independently for each document. (The process of estimating the relative size of intersection of sets can be applied to arbitrary sets). The reduction to a set intersection problem is done via a process called shingling. Via shingling each document D gets an associated set SD. This is done as follows:

Each document is viewed as a sequence of tokens. The letters, or words, or lines are taken as tokens. It is assumed that there is a parser program that takes an arbitrary document and reduces it to a canonical sequence of tokens. (Here "canonical" means that any two documents that differ only in formatting or other information that we chose to ignore, for instance punctuation, formatting commands, capitalization, and so on, will be reduced to the same sequence.) So from now on a document means a canonical sequence of tokens. A contiguous subsequence of w tokens contained in D is called a shingle. A shingle of length q is also known as a q-gram, particularly when the tokens are alphabet letters. Given a document D we can associate to it its w-shingling defined as the set of all shingles of size w contained in D. So for instance the 4-shingling of (a,rose,is,a,rose,is,a,rose) is the set {(a,rose,is,a), (rose,is,a,rose), (is,a,rose,is)} (It is possible to use alternative definitions, based on multi-sets.). Rather than deal with shingles directly, it is more convenient to associate to each shingle a numeric uid (unique id). This is done by fingerprinting the shingle. (Fingerprints are short tags for larger objects. They have the property that if two fingerprints are different then the corresponding objects are certainly different and there is only a small probability that two different objects have the same fingerprint. This probability is typically exponentially small in the length of the fingerprint.). For reasons it is particularly advantageous to use Rabin fingerprints[8] that have a very fast software implementation. Rabin finger¬prints are based on polynomial arithmetic and can be constructed in any length. It is important to choose the length of the fingerprints so that the probability of collisions (two distinct shingles getting the same fingerprint) is sufficiently low. In practice 64 bits Rabin fingerprints are sufficient. Hence from now on each document is associated D a set of numbers SD that is the result of finger printing the set of shingles in D. Note that the size of SD is about equal to the number of words in D and thus storing SD on-line for every document in a large collection is infeasible. The resemblance r(A, B)of two documents, A and B, is defined as

$r(A, B)= |S(A) \cap S(B) | / |S(A) U S(B)|$

Experiments seem to indicate that high resemblance (that is, close to 1) captures well the informal notion of "near-duplicate" or "roughly the same". (There are analyses that relate the "q-gram distance" to the edit-distance). The approach to determining syntactic similarity is related to the sampling approach developed independently by Heintze [4], though there are differences in detail and in the precise definition of the measures used. Related sampling mechanisms for determining similarity were also developed by Manber [5] and within the Stanford SCAM project. To compute the resemblance of two documents it suffices to keep for each document a relatively small, fixed size sketch. The sketches can be computed fairly fast (linear in the size of the documents) and given two sketches the re¬semblance of the corresponding documents can be computed in linear time in the size of the sketches. This is done as follows. Assume that for all documents of interest SD Π def{0,...,n− 1} =[n]. (As noted, in practice n=264.) Let Π be chosen uniformly at random over Sn , the set of permutations of [n]. Then Pr min{ Π (SA)} =min{ Π (SB)} = S(A) ∩S(B) / S(A) U S(B). [6]

So far it is seen how to estimate the resemblance of a pair of documents. For this purpose the shingle fingerprints can be quite short since collisions have only a modest influence on the estimate if applied first a random permutation to the shingles and then fingerprint the minimum value. However, sketches allow to group a collection of m documents into sets of closely resembling documents in time proportional to m log m rather than m, assuming that the clusters are well separated which is the practical case.

The clustering algorithm is performed in four phases. In the first phase, calculate a sketch for every document as explained. This step is linear in the total length of the documents. To simplify the exposition of the next three phases say temporarily that each sketch is composed of shingles, rather than images of the fingerprint of shingles under random permutations of [n].In the second phase, produce a list of all the shingles and the documents they appear in, sorted by shingle value. To do this, the sketch for each document is expanded into a list of shingle value, document ID∩ pairs. Simply sort this list. This step takes time O(m log m)where m is the number of documents. In the third phase, generate a list of all the pairs of documents that share any shingles, along with the number of shingles they have in common. To do this, take the file of sorted shingle, ID∩ pairs and expand it into a list of ID, ID, count of common shingles∩ triplets by taking each shingle that appears in multiple documents and generating the complete set of ID, ID, 1∩ triplets for that shingle. Then apply a merge-sort procedure (adding the counts for matching ID -ID pairs) to produce a single file of all ID, ID, count∩ triplets sorted by the first document ID. This phase requires the greatest amount of disk space because the initial expansion of the document ID triplets is quadratic in the number of documents sharing a shingle, and initially produces many triplets with a count of 1. Because of this fact must choose the length of the shingle fingerprints so that the number of collisions is small.

2.1.2. FINGERPRINTING WITH SIMHASH

Charikar's simhash [3] is a dimensionality reduction technique. It maps high-dimensional vectors to small-sized fingerprints. It is applied to web-pages as follows: First convert a web-page into a set of features, each feature tagged with its weight. Features are computed using standard IR techniques like tokenization, case folding, stop-word removal, stemming and phrase detection. A set of weighted features constitutes a high-dimensional vector, with one dimension per unique feature in all documents taken together. With simhash, can transform such a high-dimensional vector into an f-bit fingerprint where f is small, say 64.

Computation: Given a set of features extracted from a document and their corresponding weights, the simhash is used to generate an f-bit fingerprint as follows. An f-dimensional vector V is maintained, each of whose dimensions is initialized to zero. A feature is hashed into an f-bit hash value. These f bits (unique to the feature) increment/decrement the f components of the vector by the weight of that feature as follows: if the i-th bit of the hash value is 1, the i-th component of V is incremented by the weight of that feature; if the i-th bit of the hash value is 0, the i-th component of V is decremented by the weight of that feature. When all features have been processed, some components of V are positive while others are negative. The signs of components determine the corresponding bits of the final fingerprint.

Empirical results: Monika Henzinger conducted a study that compared simhash with Broder's shingle-based fingerprints [2]. An excellent comparison of these two approaches appears in Henzinger [7]. A great advantage of using simhash over shingles is that it requires relatively small-sized fingerprints. For example, Broder's shingle-based fingerprints [2] require 24 bytes per fingerprint (it boils down to checking whether two or more Rabin fingerprints [8] out of six are identical). With simhash, for 8B web pages, 64-bit fingerprints suffice;

The above is experimentally demonstrated by Manku in [1].

Properties of simhash: Note that simhash possesses two conflicting properties:

(A) The fingerprint of a document is a “hash" of its features, and

(B) Similar documents have similar hash values.

The latter property is atypical of hash functions.

For illustration, consider two documents that differ in a single byte. Then cryptographic hash functions like SHA-1 or MD5 will hash these two documents (treated as strings) into two completely different hash-values (the Hamming distance between the hash values would be large). However, simhash will hash them into similar hash-values (the Hamming distance would be small). In designing a near-duplicate detection system based on simhash, one has to deal with the quaintness of simhash described above. The strategy employed is as follows: The design of algorithms is done assuming that Property A holds, i.e., the fingerprints are distributed uniformly at random, and it is experimentally measured that the impact of non-uniformity introduced by Property B on real datasets.

After converting documents into simhash fingerprints, the following design problem is encountered: Given a 64-bit fingerprint of a recently-crawled web page, how to quickly discover other fingerprints that differ in at most 3 bit-positions? The problem is addressed in the next Section.

2.1.3. THE HAMMING DISTANCE PROBLEM

Definition: Given a collection of f-bit fingerprints and a query fingerprint F, identify whether an existing fingerprint differs from F in at most k bits. (In the batch-mode version of the above problem, a set of query fingerprints is considered instead of a single query fingerprint). As a concrete instance of the above problem2, consider a collection of 8B 64-bit fingerprints, occupying 64GB. In the online version of the problem, for a query fingerprint F, there is need to ascertain within a few milliseconds whether any of the existing 8B 64-bit fingerprints differs from F in at most k = 3 bit-positions. In the batch version of the problem, there is a set of, say, 1M query fingerprints (instead of a solitary query fingerprint F) and to solve the same problem for all 1M query fingerprints in roughly 100 seconds. This would amount to a throughput of 1B queries per day. Manku has explored in [1] the design space by considering two simpleminded but impractical approaches. One approach is to build a sorted table of all existing fingerprints. Given F, probe such a table with each F' whose Hamming distance from F is at most k. The total number of probes is prohibitively large: for 64-bit fingerprints and k = 3, need is 64C3 = 41664 probes. An alternative is to pre-compute all F' such that some existing fingerprint is at most Hamming distance k away from F'. In this approach, the total number of pre-computed fingerprints is prohibitively large: it could be as many as 41664 times the number of fingerprints.

A practical algorithm has been developed that lies in between the two approaches outlined above: it is possible to solve the problem with a small number of probes and by duplicating the table of fingerprints by a small factor.

Intuition: Consider a sorted table of 2d f-bit truly random fingerprints. Focus on just the most significant d bits in the table. A listing of these d-bit numbers amounts to “almost a counter" in the sense that (a) quite a few 2d bit combinations exist, and (b) very few d-bit combinations are duplicated. On the other hand, the least significant f – d bits are almost random.

Now choose d’ such that │d’ - d│ is a small integer. Since the table is sorted, a single probe suffices to identify all fingerprints which match F in d’ most significant bit-positions. Since │d’ - d│ is small, the number of such matches is also expected to be small. For each matching fingerprint, we can easily figure out if it differs from F in at most k bit-positions or not (these differences would naturally be restricted to thef – d’ least-significant bit-positions).

The procedure described above helps in locating an existing fingerprint that differs from F in k bit-positions, all of which are restricted to be among the least significant f- d’ bits of F. This takes care of a fair number of cases. To cover all the cases, it suffices to build a small number of additional sorted tables.

2.1.4 Algorithm for Online Queries

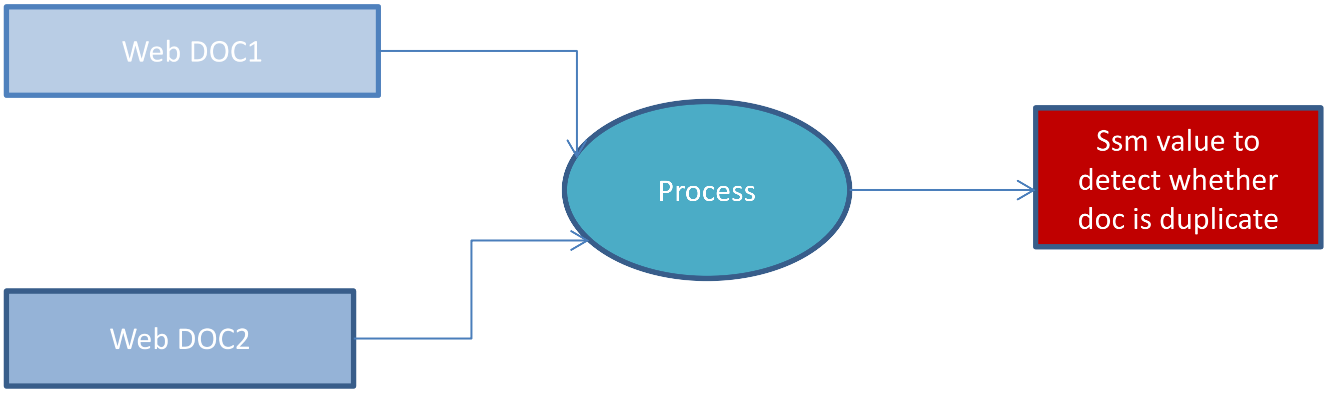
t tables are build: T1, T2, ….. Tt. Associated with table Tt are two quantities: an integer Pi and a permutation Πi over the f bit-positions. Table Ti is constructed by applying permutation Πi to each existing fingerprint; the resulting set of permuted f-bit fingerprints are sorted. Further, each table is compressed and stored in main-memory of a set of machines. Given fingerprint F and an integer k, we probe these tables in parallel:

Step 1: Identify all permuted fingerprints in Ti whose top Pi bit-positions match the top Pi bit-positions of Π(F).

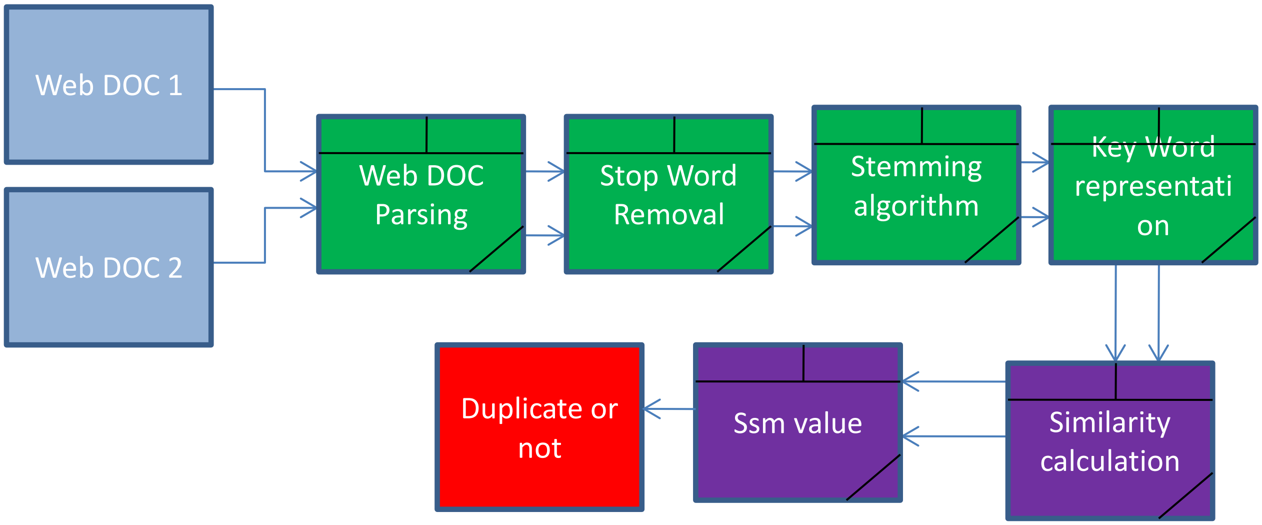
Step 2: For each of the permuted fingerprints identified in Step 1, check if it differs from Πi(F) in at most k bitpositions. In Step 1, identification of the first fingerprint in table Ti whose top Pi bit-positions match the top Pi bit-positions of Πi(F) can be done in O(Pi) steps by binary search. If we assume that each fingerprint were truly a random bit sequence, interpolation search shrinks the run time to O(log Pi) steps.

2.2 Project/Application Information flows

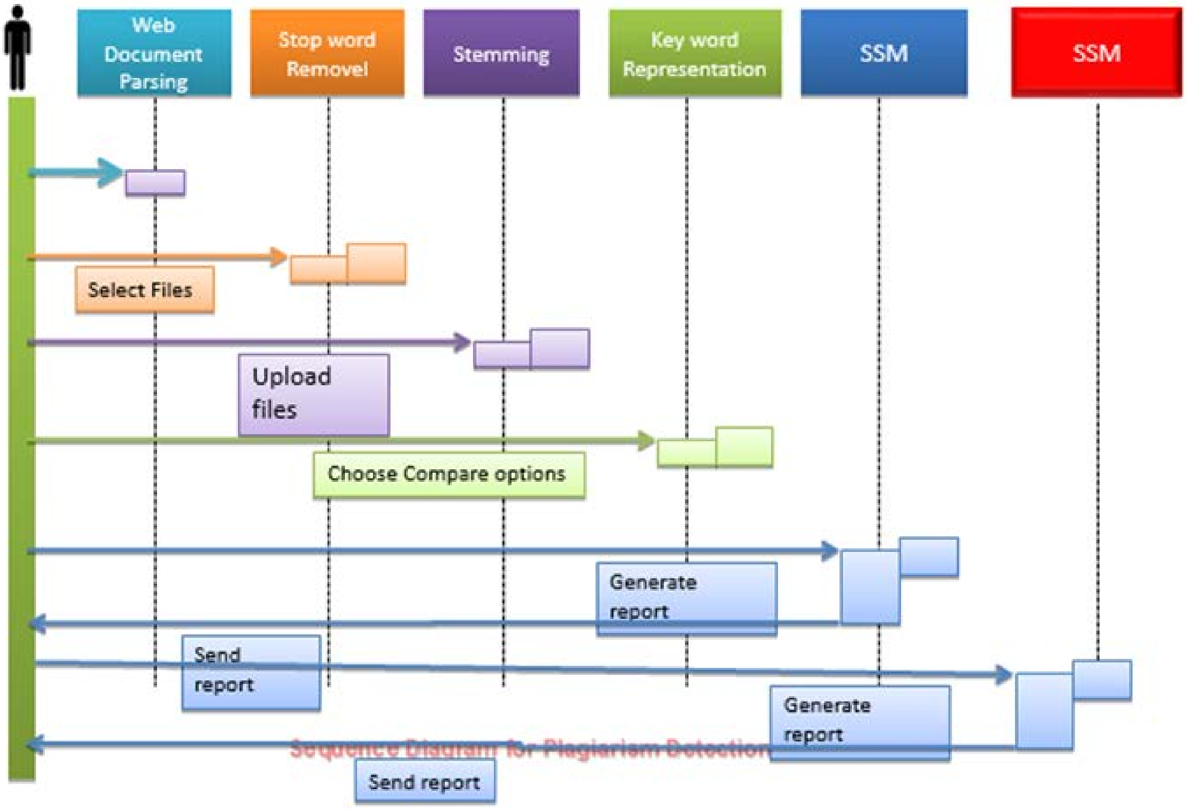
2.2.1 Process Flow



2.2.2 DATAFLOW DIAGRAM CONTEXT LEVEL 1



2.2.3 SEQUENCE DIAGRAM



Sequence Diagram for Plagiarism Detection

Stop word removal

It is necessary and beneficial to remove the commonly utilized stop words such as "it", "can", "an", "and", "by", "for", "from", "of", "the", "to", "with" and more. Stop listing aids in the reduction of size of the indexing file besides enhancing efficiency and value.

Stemming Algorithm

Stemming facilitates the reduction of all words possessing an identical root to a single one. This is achieved by removing each word of its derivational and inflectional suffixes.

For instance, "connect", "connected" and "connection" are all condensed to "connect".

Keywords

Initially the keywords and their number of occurrences in a web page have been sorted in descending order based on their counts. Afterwards, n numbers of keywords with highest counts are stored in a table and the remaining keywords are indexed and stored in another table.

2.3 Interactions with other Projects (if Any)

2.3.1 Web mirrors: For web search, successful identification of web mirrors results in smaller crawling/storage/indexing costs in the absence of near-duplicates, better top-k results for search queries, improvement in page-rank by reducing the in-degree of sites resulting from near-duplicates, cost-saving by not asking human evaluators to rank near duplicates. See Bharat et al for a comparison of techniques for identifying web-mirrors.

2.3.2 Clustering for \related documents" query: For example, given a news article, a web-surfer might be interested in locating news articles from other sources that report the same event. The notion of \similarity" is at a high level { one could say that the notion of similarity is \semantic" rather than \syntactic", quite different from the notion of duplicates or near-duplicates discussed above. One approach is to use Latent Semantic Indexing. Another approach is to exploit the linkage structure of the web (see Dean and Henzinger who build upon Kleinberg's idea of hubs and authorities). Going further along these lines, Kumar et al have proposed discovering \online communities" by identifying dense bipartite sub-graphs of the web-graph.

2.3.3 Data extraction: Given a moderate-sized collection of similar pages, say reviews at www.imdb.com, the goal is to identify the schema/DTD underlying the collection so that we can extract and categorize useful information from these pages. See Joshi et al (and references therein) for a technique that clusters web-pages on the basis of structural similarity. See Arasu and Garcia- Molina for another technique that identifies templates underlying pages with similar structure. Also note that metadata (HTML tags) was ignored in a) and b) above.

2.3.4 Plagiarism: Given a set of reports, articles or assignment submissions (both source-code and textual reports), the goal is to identify pairs of documents that seem to have borrowed from each other significantly. For some early work in this area, see articles by Baker, the COPS system by Brin et al and SCAM by Shivakumar and Garcia-Molina.

2.3.5 Spam detection: Given a large number of recently received e-mails, the goal is to identify SPAM before depositing the e-mail into recipients' mailboxes. The premise is that spammers send similar e-mails en masse, with small variation in the body of these e-mails. See

Kolcz et al, who build upon previous work by Chowdhury et al.

2.3.6 Duplicates in domain-specific corpora: The goal is to identify near-duplicates arising out of revisions, modifications, copying or merger of documents, etc. See Conrad and Schriber for a case-study involving legal documents at a firm. Manber initiated an investigation into identification of similar files in a file system. Our near-duplicate detection system improves web crawling, a goal not shared with any of the systems described above.

2.4 Interactions with other Applications

List all possible interaction with other application, and what are needed to address these interactions.

2.5 Capabilities

- The drastic development of the Internet and the growing necessity to incorporate heterogeneous data is accompanied by the issue of the existence of near duplicate data.
- Even if the near duplicate data don’t exhibit bit wise identical nature they are remarkably similar.
- The duplicate and near duplicate web pages either increase the index storage space or slow down or increase the serving costs which annoy the users, thus causing huge problems for the web search engines.

2.6 Risk Assessment and Management

To reduce the risk, we have identified the risk factors and used the bellow approaches to reduce the risk factors. **FEATURESET**

2.6.1 Per Document

a) Shingles from page content: Consider the sequence of words in a document. A shingle is the hash-value of a k-gram which is a sub-sequence of k successive words. The set of shingles constitutes the set of features of a document. The choice of k is crucial3. Hashes of successive k-grams can be efficiently computed by using Rabin's fingerprinting technique. Manber created shingles over characters. The COPS system by Brin et al used sentences for creating shingles. Broder et al created shingles over words. The total number of shingles per document is clearly large. Therefore, a 3Small k makes dissimilar documents appear similar. Large k makes similar documents appear dissimilar. small-sized signature is computed over the set of shingles, as described in the next sub-section.

b) Document vector from page content: In contrast to shingles, a document can be characterized by deploying traditional IR techniques. The idea is to compute its document-vector" by case-folding, stop-word removal, stemming, computing term-frequencies and finally, weighing each term by its inverse document frequency (IDF). Next, given two documents, a \measure" of similarity is defend.

Hoad and Zobel argue that the traditional cosine-similarity measure is inadequate for near duplicate detection. They define and evaluate a variety of similarity measures (but they do not develop any signature-scheme to compress the document-vectors). A different approach is taken by Chowdhury et al who compute a lexicon (the union of all terms existing in the collection of documents). The lexicon is then pruned (a variety of schemes are studied by the authors). Each document-vector is then modified by removing terms that have been pruned from the lexicon. The resulting document-vectors are fingerprinted. Two documents are said to be near-duplicates if their fingerprints match. This scheme is rather brittle for near duplicate detection { a follow-up paper ameliorates the problem by constructing multiple lexicons (these are random subsets of the original lexicon). Now multiple fingerprints per document are computed and two documents are said to be duplicates if most of their fingerprints match. An issue to keep in mind when dealing with document vectors is that the IDF of any term is global information which changes as the collection changes.

c) Connectivity information: For the purpose of finding \related pages", Dean and Henzinger exploited the linkage structure of the web. The premise is that similar pages would have several incoming links in common. Haveliwala et al point out that the quality of duplicate detection is poor for pages with very few incoming links. This can be ameliorated by taking anchor text and anchor windows into account.

d) Anchor text, anchor window: Similar documents should have similar anchor text. Haveliwala et al study the impact of anchor-text and anchor-windows, where an anchor-window is the text surrounding the anchortext, for example, the paragraph it belongs to. The words in the anchor text/window are folded into the document-vector itself. A weighing function that diminishes the weight of words that are farther away from the anchor text is shown to work well.

e) Phrases: Cooper et al propose identification of phrases using a phrase-detection system and computing a document-vector that includes phrases as terms. They have tested their ideas on a very small collection (tens of thousands). The idea of using phrases also appears in the work of Hammouda and Kamel who build sophisticated indexing techniques for web-clustering. We chose to work with the document vector model; simhash converts document vectors into fingerprints. Augmenting the document vector by other signals (anchor text and connectivity information, for example) might improve the quality of our system. We leave these possibilities as future work.

2.6.2 Signature Schemes

a) Mod-p shingles: A simple compression scheme for shingle based fingerprints is to retain only those fingerprints whose remainder modulus p is 0, for a sufficiently large value of p. The number of fingerprints retained is variable sized. Moreover, it is important to ignore commonly occurring fingerprints since they contribute to false-matches. Drawback of this scheme is that the distance between successive shingles that are retained, is unbounded. This problem has been ameliorated by the \winnowing" technique by Schliemer et al. Hoad and Zobel compare a variety of other ideas for pruning the set of shingle-based fingerprints.

b) Min-hash for Jaccard similarity of sets: For two sets A and B, let the measure of similarity be, also known as the Jaccard measure. Interestingly, it is possible to devise a simple signature scheme such that the probability that the signatures of A and B match is exactly the Jaccard measure. Several experimental studies have tested the efficacy of min-hash in various settings (Cohen et al for association rule mining, Chen et al for selectivity estimation of boolean queries, Gionis et al for indexing set-value predicates and Haveliwala for web-clustering).

c) Signatures/fingerprints over IR-based document vectors: Charikar's simhash is a fingerprinting technique for compressing document vectors such that two fingerprints are similar if the document vectors are similar. Another technique for computing signatures over document-vectors is the I-Match algorithm by Chowdhury et al that we described earlier. An improved I-Match algorithm appears in. These algorithms have been tested on small document-collections (of the order of tens of thousands) and appear fairly brittle. d) Checksums: Pugh and Henzinger's patent contains the following idea: we divide words in a document into k buckets (by hashing the words, for example), and compute a checksum of each bucket. The set of checksums of two similar documents should agree for most of the buckets. We chose to work with simhash primarily because it allows us to work with small-sized fingerprints.

2.6.3 Crawling Web Pages

The crawler module (Figure 2) retrieves pages from the Web for later analysis by the indexing module. As discussed in the introduction, a crawler module typically starts o_ with an initial set of URLs *S*0. Roughly, it _rst places *S*0 in a queue, where all URLs to be retrieved are kept and prioritized. From this queue, the crawler gets a URL (in some order), downloads the page, extracts any URLs in the downloaded page, and puts the new URLs in the queue. This process is repeated until the crawler decides to stop. Given the enormous size and the change rate of the Web, many issues arise, including the following:

It should also contain recommendations to eliminate or minimize these risk

3 Project Requirements

3.1 Identification of Requirements

This section provides a brief explanation of the use of named and enumerated requirements to identify and number requirements. The following format is an example:

<GSU-GS_FA2015-1 User-Capability-000100>

Parse Web Document

Implementation: Mandatory

Information extracted from the crawled documents aid in determining the future path of a crawler. Parsing may either be as simple as hyperlink/URL extraction or complex ones such as analysis of HTML tags by cleaning the HTML content . It is inevitable for a parser that has been designed to traverse the entire web to encounter numerous errors. The parser tends to obtain information from a web page by not considering a few common words like a, an, the and more, HTML tags, Java Scripting and a range of other bad characters.

<GSU-GS_FA2015-1 User-Capability-000101>

Remove Stop Words

Implementation: Mandatory

It is necessary and beneficial to remove the commonly utilized stop words such as "it", "can" ,"an", "and", "by", "for", "from", "of", "the", "to", "with" and more either while parsing a document to obtain information about the content or while scoring fresh URLs that the page recommends. This procedure is termed as stop listing [31]. Stop listing aids in the reduction of size of the indexing file besides enhancing efficiency and value.

<GSU-GS_FA2015-1 User-Capability-000102>

Apply Stemming Algorithm

Implementation: Mandatory

Variant word forms in Information Retrieval are restricted to a common root by Stemming. The postulation lying behind is that, two words posses the same root represent identical concepts. Thus terms possessing to identical meaning yet appear morphologically dissimilar are identified in an IR system by matching query and document terms with the aid of Stemming [4]. Stemming facilitates the reduction of all words possessing an identical root to a single one. This is achieved by removing each word of its derivational and inflectional suffixes [26]. For instance, "connect," "connected" and "connection" are all condensed to "connect".

<GSU-GS_FA2015-1 User-Capability-000103>

Form Keyword Table

Implementation: Mandatory

We posses the distinct keywords and their counts in each of the crawled web page as a result of stemming. These keywords are then represented in a form to ease the process of near duplicates detection. This representation will reduce the search space for the near duplicate detection. Initially the keywords and their number of occurrences in a web page have been sorted in descending order based on their counts. Afterwards, n numbers of keywords with highest counts are stored in a table and the remaining keywords are indexed and stored in another table. In our approach the value of n is set to be 4. The similarity score between two documents can be calculated if and only the prime keywords of the two documents are similar. Thus the search space is reduced for near duplicates detection.

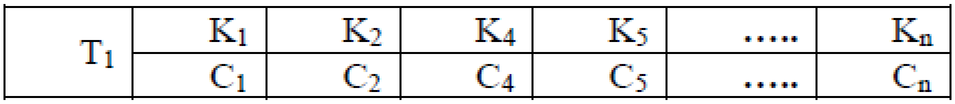
<GSU-GS_FA2015-1 User-Capability-000104>

Calculate Similarity Score

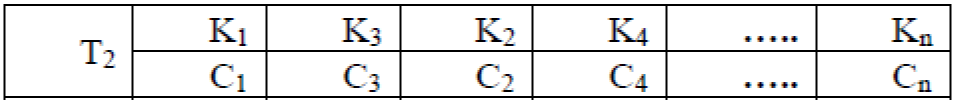
Implementation: Mandatory

If the prime keywords of the new web page do not match with the prime keywords of the pages in the table, then the new web page is added in to the repository. If all the keywords of both pages are same then the new page is considered as duplicate and thus is not included in the repository. If the prime keywords of new page are same with a page in the repository, then the similarity score between the two documents is calculated. The similarity score of two web documents is calculated as follows:

Let T1 and T2 be the tables containing the extracted keywords and their corresponding counts.



T_1	K_1	K_2	K_4	K_5	…..	K_n
	C_1	C_2	C_4	C_5	…..	C_n



T_2	K_1	K_3	K_2	K_4	…..	K_n
	C_1	C_3	C_2	C_4	…..	C_n

The keywords in the tables are considered individually for the similarity score calculation. If a keyword is present in both the tables, the formula used to calculate the similarity score of the keyword is as follows.

<GSU-GS_FA2015-1 User-Capability-000105>

Compare Similarity Score

Implementation: Mandatory

$$a = \Delta[K_i]_{T_1}$$
$$b = \Delta[K_i]_{T_2}$$
$$S_{D_C} = \log(count(a)/count(b)) * \log(abs(1+(a-b)))$$

Here ‘a’ and ‘b’ represent the index of a keyword in the two tables respectively. If the keywords of T1 \ T2 ≠ φ, we use the following formula to calculate the similarity score. The amount of the keywords present in T1 but not in T2 is taken as

$$S_{D_{T_1}} = \log(count\ (a)) * \log\left(1+|T_2|\right)$$

If the keywords of T2 \ T1 ≠ φ, we use the below mentioned formula to calculate the similarity score. The occurrences of the keywords present in T2 but not in T1 is taken as

$$S_{D_{T_2}} = \log(count(b)) * \log\left(1+|T_1|\right)$$

<GSU-GS_FA2015-1 User-Capability-000106>

Set Threshold

Implementation: Mandatory

<GSU-GS_FA2015-1 User-Capability-000107>

Compare SSM with threshold

Implementation: Mandatory

The similarity score (SSM) of a page against another page is calculated by using the following equation.

$$SS_M = \frac{\sum_{i=1}^{|N_C|} S_{D_C} + \sum_{i=1}^{|N_{T_1}|} S_{D_{T_1}} + \sum_{i=1}^{|N_{T_2}|} S_{D_{T_2}}}{N}$$

Where $N = \left(|T_1|+|T_2|\right)/2$. The web documents with similarity score lesser than a predefined threshold are near duplicates of documents already present in repository. These near duplicates are not added in to the repository for further process such as search engine indexing.

The key factors for the success of the World Wide Web are its large size and the lack of a centralized control over its contents. Both issues are also the most important source of problems for locating information. The Web is a context in which traditional Information Retrieval methods are challenged, and given the volume of the Web and its speed of change, the coverage of modern search engines is relatively small. Moreover, the distribution of quality is very skewed, and interesting pages are scarce in comparison with the rest of the content.

<GSU-GS_FA2015-1 User-Capability-000108>

Discard or created output

Implementation: Mandatory

We argue that the number of pages on the Web can be considered infinite, and given that a Web crawler cannot download all the pages, it is important to capture the most important ones as early as possible during the crawling process. We propose, study, and implement algorithms for achieving this goal, showing that we can crawl 50% of a large Web collection and capture 80% of its total Pagerank value in both simulated and real Web environments. We also model and study user browsing behavior in Web sites, concluding that it is not necessary to go deeper than five levels from the home page to capture most of the pages actually visited by people, and support this conclusion with log analysis of several Web sites. We also propose several mechanisms for server cooperation to reduce network traffic and improve the representation of aWeb page in a search engine with the help of Web site managers.

3.2 Operations, Administration, Maintenance and Provisioning (OAM&P)

This is section to describe the capabilities/requirements you will be providing for your user or customer to administrate and maintain the usage of your project. (e.g., user data backup, fault recovery, routine maintenance ………)

3.3 Security and Fraud Prevention

Replace this section with description and requirement to address possible internal and external security issues.

3.4 Release and Transition Plan

Explain how the project will be deployed to customer, or update from current release to newer release.

4 Project Design Description

The analysis of the structure and informatics of the web is facilitated by a data collection technique known as Web Crawling. The collection of as many beneficiary web pages as possible along their interconnection links in a speedy yet proficient manner is the prime intent of crawling.
Web crawling becomes a tedious process due to the subsequent features of the web, the large volume and the huge rate of change due to voluminous number of pages being added or removed each day.
The duplicate and near duplicate web pages either increase the index storage space or slow down or increase the serving costs which annoy the users, thus causing huge problems for the web search engines.

In recent times, significant research has been contributed towards the vital issue, the elimination of duplicate and near duplicate web documents. The near duplicate web pages are detected followed by the storage of crawled web pages in to repositories. The keywords are extracted from the crawled pages initially and on the basis of the extracted keywords, the similarity score between the two pages is calculated. The documents are considered as near duplicates if its similarity scores are lesser than a threshold value.

.

5 Project Internal/external Interface Impacts and Specification

- **Software Requirements:**

Language:	Java (JDK 1.7)
Database:	MS Access
Operating System:	Microsoft Windows 2000 or later

- **Hardware Requirements:**

Processor:	Intel Pentium 4 or More
RAM:	1 GB Ram
Hard Disk:	PC with 80GB

6 Project Design Units Impacts

Even though the near duplicate documents are not bitwise identical they bear striking similarities. The near duplicates are not considered as “exact duplicates” but are files with minute differences.

Typographical errors, versioned, mirrored, or plagiarized documents, multiple representations of the same physical object, spam emails generated from the same template, and many such phenomenon's may result in near duplicate data. A considerable percentage of web pages have been identified as be near-duplicates according to various studies [3, 196 and 24]. These studies propose that near duplicates constitute almost 1.7% to 7% of the web pages traversed by crawlers. The steps involved in our approach are presented in the following subsections.

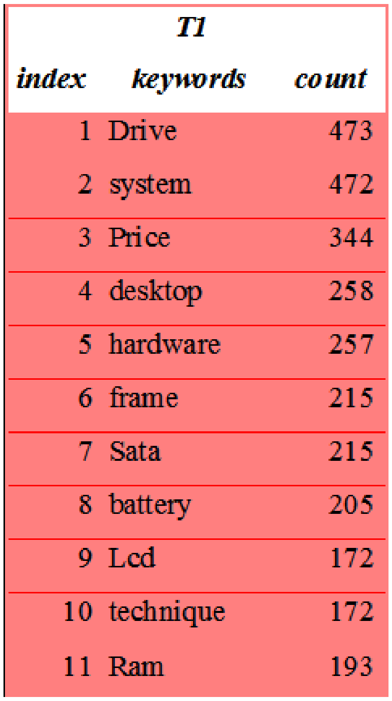
6.1 Functional Area/Design Unit A

6.1.1 Functional Overview

The results of our experiments are presented in this section. The proposed approach is implemented in Java with MS Access as backend. The keywords extracted from the web documents are stored in MS Access. In our experiments, we have analyzed the proposed approach using many urls. The near duplicates have been detected efficiently by the proposed approach. This has been achieved with the aid of the similarity scores. The similarity score would find all pairs of web pages whose duplications are identified by a given predefined threshold. Upon considering the similarity scores, a lower distance measure indicates that the documents are more similar and hence are regarded as near duplicates. If the extracted keywords from the two web documents are almost same, then it is considered as near duplicate and their distance is of minimum value. One such example for near duplicate web documents, the extracted keywords and their calculated similarity scores are as follows:

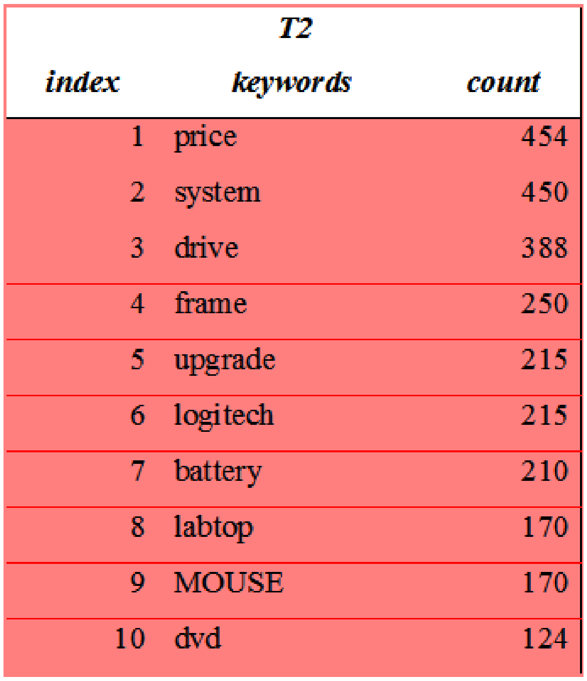
The keywords extracted from URL 1 and URL 2 are given in Table T1 and T2 respectively.

Table 1: Keywords extracted from URL 1.



	T1	
index	*keywords*	*count*
1	Drive	473
2	system	472
3	Price	344
4	desktop	258
5	hardware	257
6	frame	215
7	Sata	215
8	battery	205
9	Lcd	172
10	technique	172
11	Ram	193

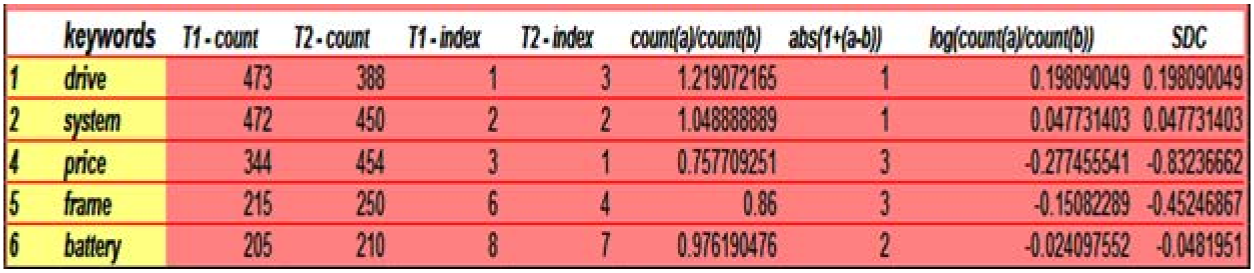
Table 2: Keywords extracted from URL 2.



	T2	
index	*keywords*	*count*
1	price	454
2	system	450
3	drive	388
4	frame	250
5	upgrade	215
6	logitech	215
7	battery	210
8	labtop	170
9	MOUSE	170
10	dvd	124

The similarity scores calculated for the keywords present in both the tables are as follows:

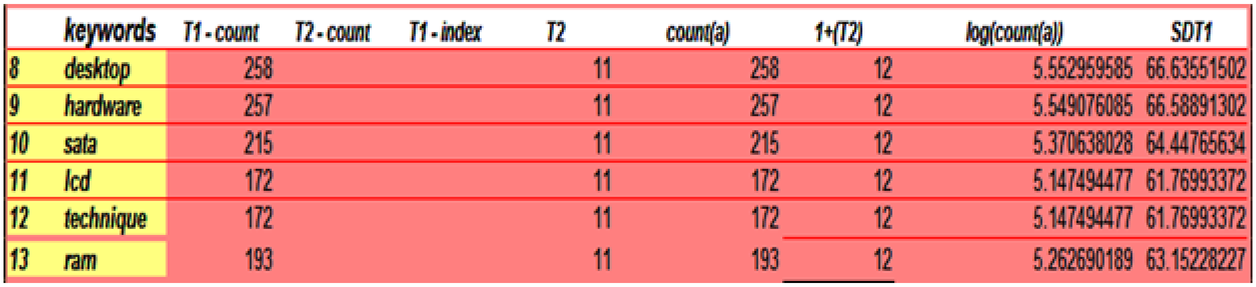
Table 3: Similarity scores of the keywords present in T1 and T2.



	keywords	T1 - count	T2 - count	T1 - index	T2 - index	count(a)/count(b)	abs(1+(a-b))	log(count(a)/count(b))	SDC
1	drive	473	388	1	3	1.219072165	1	0.198090049	0.198090049
2	system	472	450	2	2	1.048888889	1	0.047731403	0.047731403
4	price	344	454	3	1	0.757709251	3	-0.277455541	-0.83236662
5	frame	215	250	6	4	0.86	3	-0.15082289	-0.45246867
6	battery	205	210	8	7	0.976190476	2	-0.024097552	-0.0481951

The similarity scores calculated for the keywords present in Table T1 but not in Table T2 are as follows:

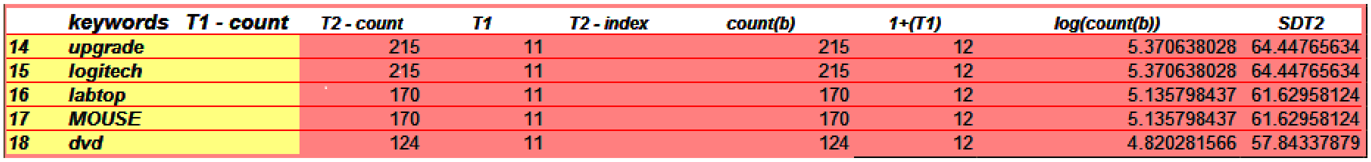
Table 4: Similarity scores of the keywords present in T1 and not in T2.



	keywords	T1 - count	T2 - count	T1 - index	T2	count(a)	1+(T2)	log(count(a))	SDT1
8	desktop	258			11	258	12	5.552959585	66.63551502
9	hardware	257			11	257	12	5.549076085	66.58891302
10	sata	215			11	215	12	5.370638028	64.44765634
11	lcd	172			11	172	12	5.147494477	61.76993372
12	technique	172			11	172	12	5.147494477	61.76993372
13	ram	193			11	193	12	5.262690189	63.15228227

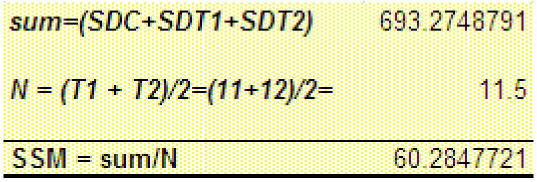
The similarity scores calculated for the keywords present in Table T2 but not in Table T1 are as follows:

Table 5: Similarity scores of the keywords present in T2 and not in T1.



	keywords	T1 - count	T2 - count	T1	T2 - index	count(b)	1+(T1)	log(count(b))	SDT2
14	upgrade		215	11		215	12	5.370638028	64.44765634
15	logitech		215	11		215	12	5.370638028	64.44765634
16	labtop		170	11		170	12	5.135798437	61.62958124
17	MOUSE		170	11		170	12	5.135798437	61.62958124
18	dvd		124	11		124	12	4.820281566	57.84337879

Finally, we have calculated the similarity score SSM between URL 1 and URL 2 with the aid of the similarity scores in Table 3, 4, 5 and the SSM value is



sum=(SDC+SDT1+SDT2)	693.2748791
N = (T1 + T2)/2=(11+12)/2=	11.5
SSM = sum/N	60.2847721

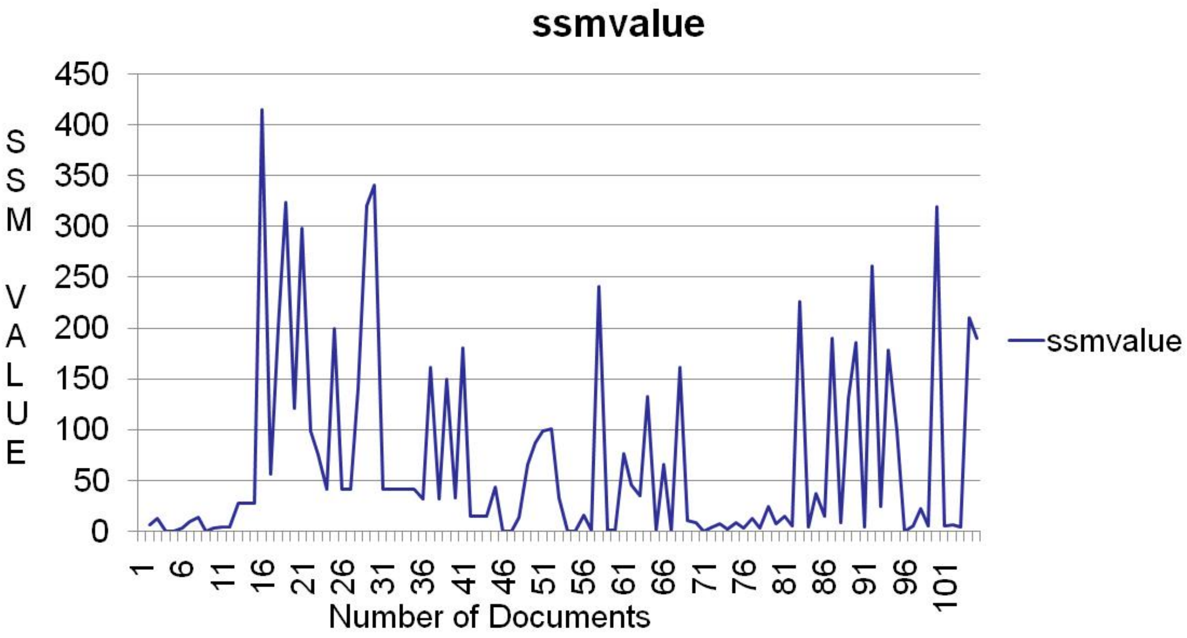
Consequently we have compared this SSM value with the predefined threshold value and this Value is lesser than the predefined threshold. Based on this, we have concluded that the URL1 and URL 2 are near duplicates.

6.1.2 Impacts

6.1.3 Requirements

6.2 Experimental results

6.2.1 Results Overview



And then using that SSM values calculating the threshold.

Things to be considered to fix the threshold are

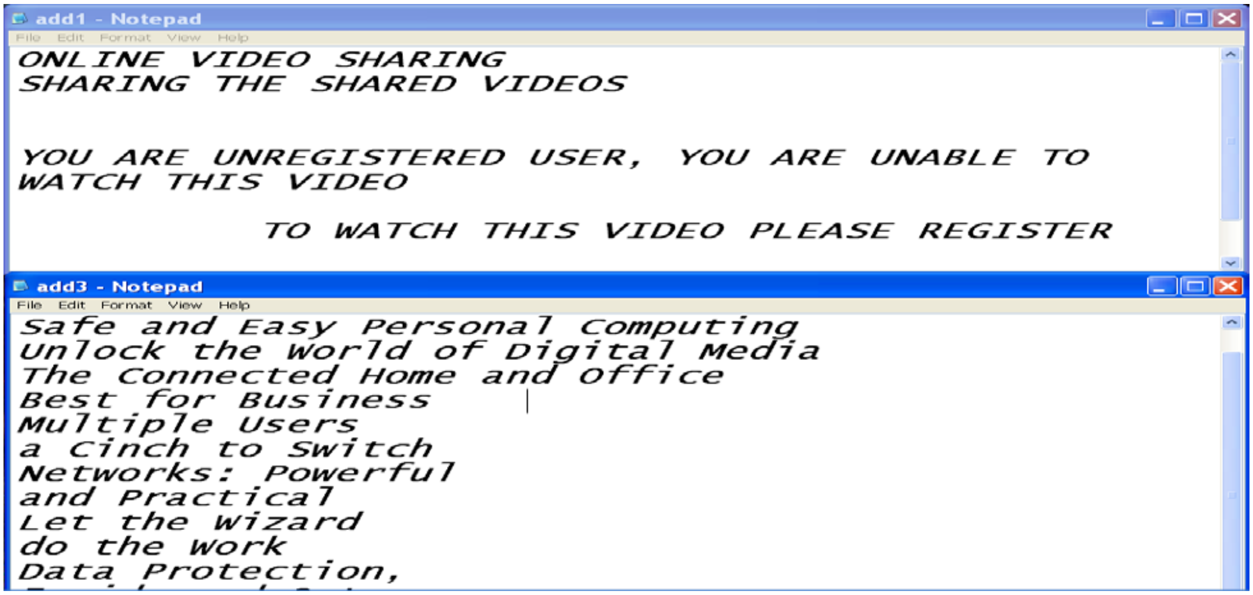
- Timestamps
- Advertisement

- Counters
- SSM value

Considering two identical web documents with same content but adding advertisements and timestamps.

Calculating SSM value.

Two example documents with same content but difference in advertisements



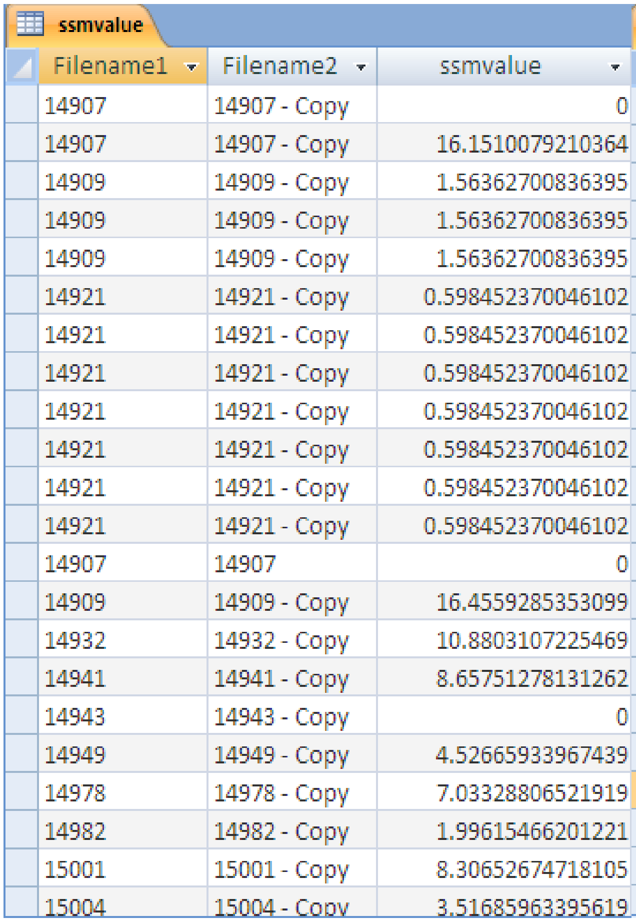
```
ONLINE VIDEO SHARING
SHARING THE SHARED VIDEOS


YOU ARE UNREGISTERED USER, YOU ARE UNABLE TO
WATCH THIS VIDEO

          TO WATCH THIS VIDEO PLEASE REGISTER
```

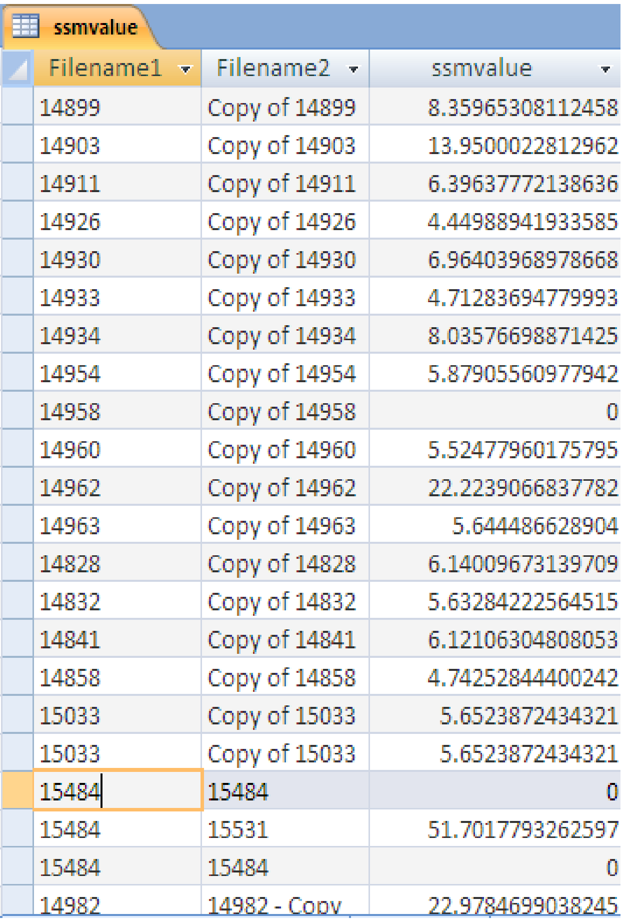
```
Safe and Easy Personal Computing
Unlock the World of Digital Media
The Connected Home and Office
Best for Business
Multiple Users
a Cinch to Switch
Networks: Powerful
and Practical
Let the Wizard
do the Work
Data Protection,
```

By comparing large number of documents we formulated a database



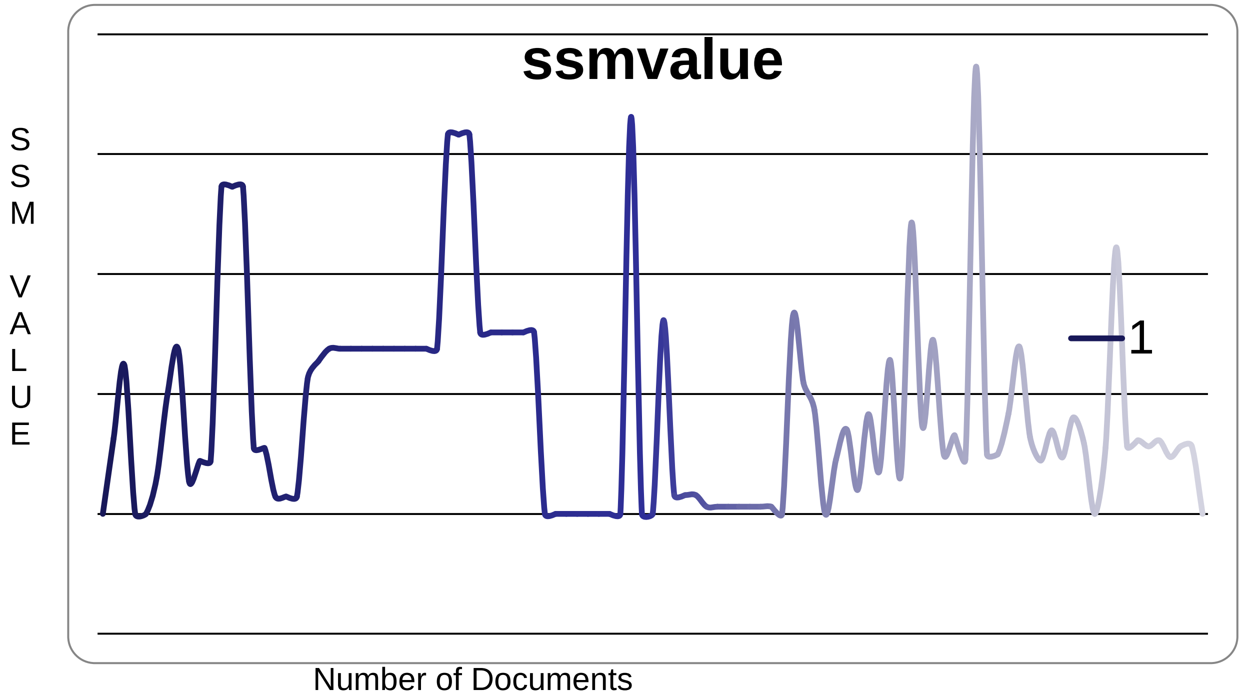
ssmvalue

Filename1	Filename2	ssmvalue
14907	14907 - Copy	0
14907	14907 - Copy	16.1510079210364
14909	14909 - Copy	1.56362700836395
14909	14909 - Copy	1.56362700836395
14909	14909 - Copy	1.56362700836395
14921	14921 - Copy	0.598452370046102
14921	14921 - Copy	0.598452370046102
14921	14921 - Copy	0.598452370046102
14921	14921 - Copy	0.598452370046102
14921	14921 - Copy	0.598452370046102
14921	14921 - Copy	0.598452370046102
14921	14921 - Copy	0.598452370046102
14907	14907	0
14909	14909 - Copy	16.4559285353099
14932	14932 - Copy	10.8803107225469
14941	14941 - Copy	8.65751278131262
14943	14943 - Copy	0
14949	14949 - Copy	4.52665933967439
14978	14978 - Copy	7.03328806521919
14982	14982 - Copy	1.99615466201221
15001	15001 - Copy	8.30652674718105
15004	15004 - Copy	3.51685963395619



ssmvalue

Filename1	Filename2	ssmvalue
14899	Copy of 14899	8.35965308112458
14903	Copy of 14903	13.9500022812962
14911	Copy of 14911	6.39637772138636
14926	Copy of 14926	4.44988941933585
14930	Copy of 14930	6.96403968978668
14933	Copy of 14933	4.7128369477993
14934	Copy of 14934	8.03576698871425
14954	Copy of 14954	5.87905560977942
14958	Copy of 14958	0
14960	Copy of 14960	5.52477960175795
14962	Copy of 14962	22.2239066837782
14963	Copy of 14963	5.644486628904
14828	Copy of 14828	6.14009673139709
14832	Copy of 14832	5.63284222564515
14841	Copy of 14841	6.12106304808053
14858	Copy of 14858	4.74252844400242
15033	Copy of 15033	5.6523872434321
15033	Copy of 15033	5.6523872434321
15484	15484	0
15484	15531	51.7017793262597
15484	15484	0
14982	14982 - Copy	22.9784699038245



The above graph represents the SSM value form database and the average of these SSM values are considered as threshold

Final approximate threshold value is 19.5043

6.2.2 Impacts

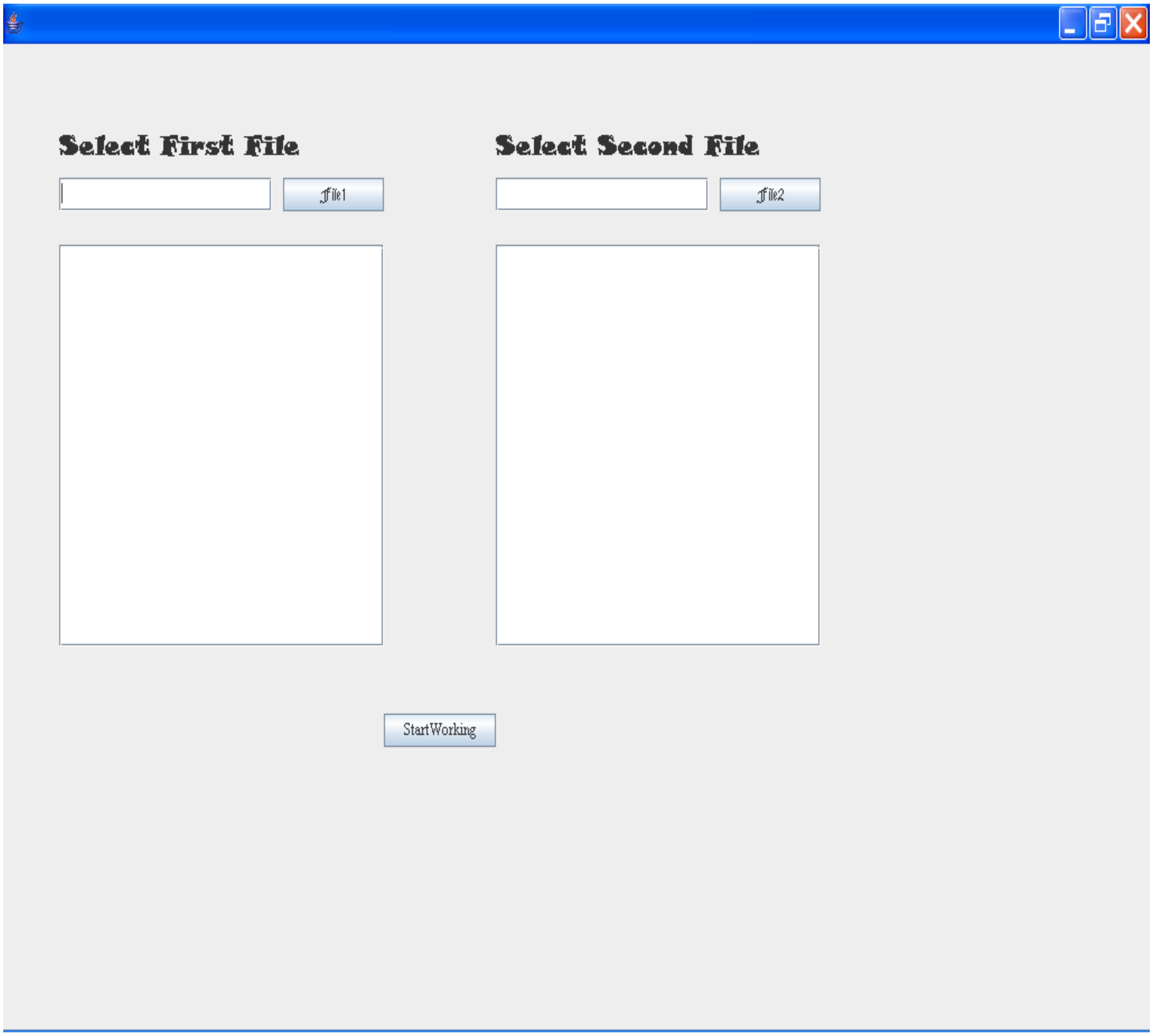
6.2.3 Requirements

7 Conclusion

Although the web is a huge information store, Information Retrieval has been posed with serious difficulties owing to its various features, for instance, the presence of huge volume of unstructured or semi-structured data; their dynamic nature; existence of duplicate and near duplicate documents and the similar ones. Huge challenges have been posed by the voluminous amounts of web documents swarming the web to the web search engines making their less appropriate to the users. The web crawling research community has extensively recognized the detection of duplicate and near duplicate web pages. We have presented a novel and efficient approach for detection of near duplicate web documents in web crawling in this paper. On the basis of the keywords extracted from the web pages, the proposed approach has detected the duplicate and near duplicate web pages efficiently. The proposed duplicates detection approach also accomplishes reduced memory spaces for web repositories and improved search engine quality.

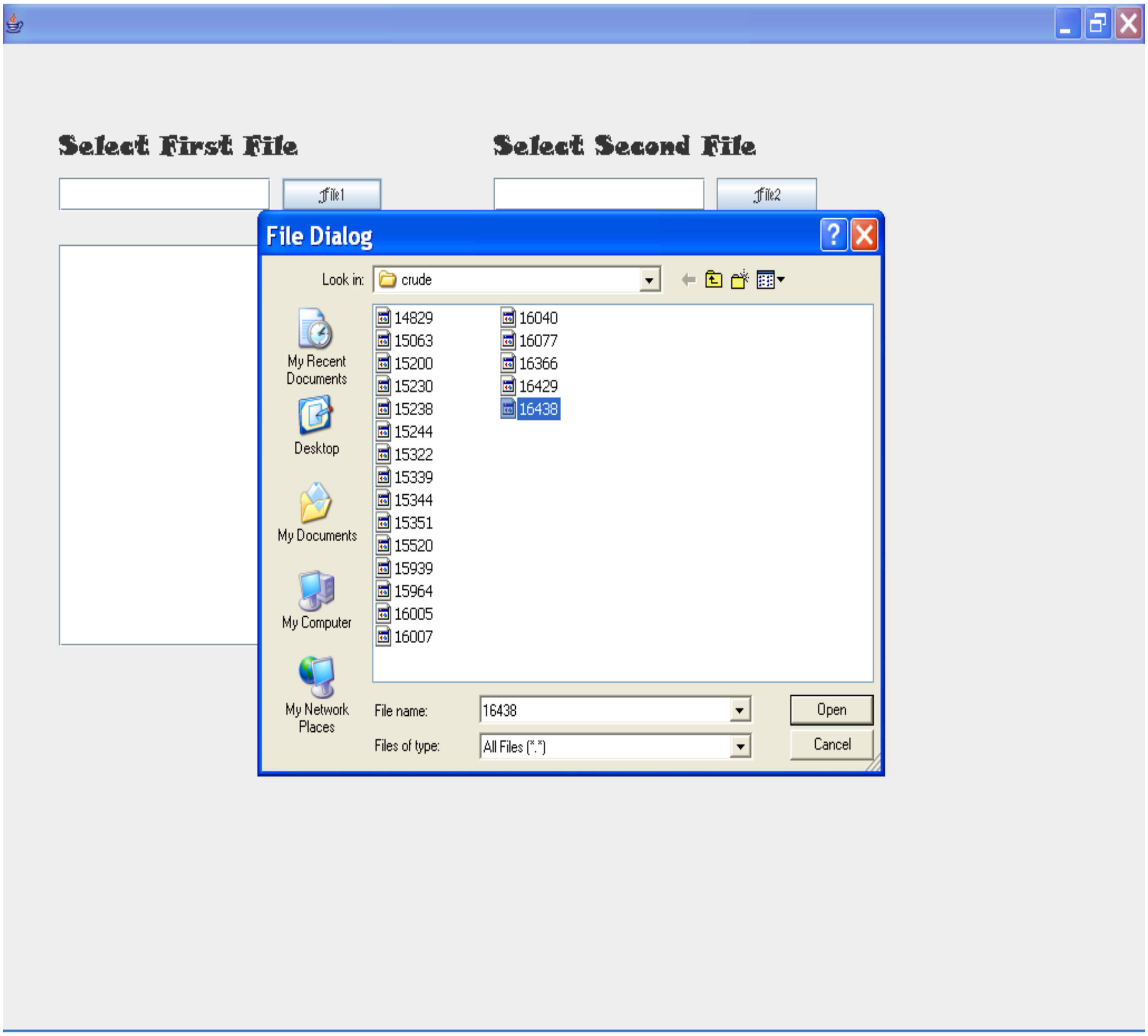
8 Output Screens

8.1 Main Window:

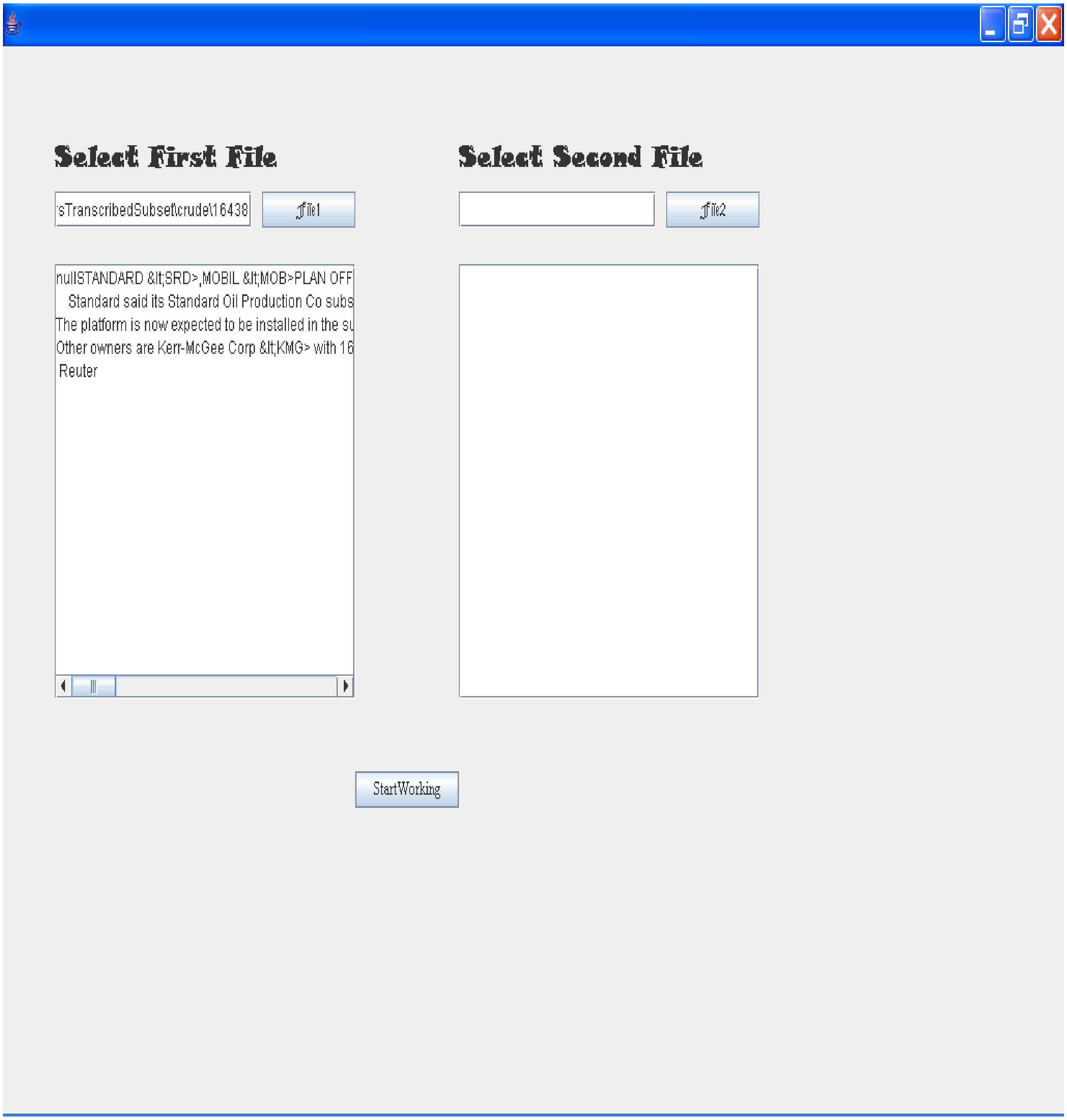


Here we can select two documents as input to the code so that it can calculate the SSM value

8.2 First File Browsing:



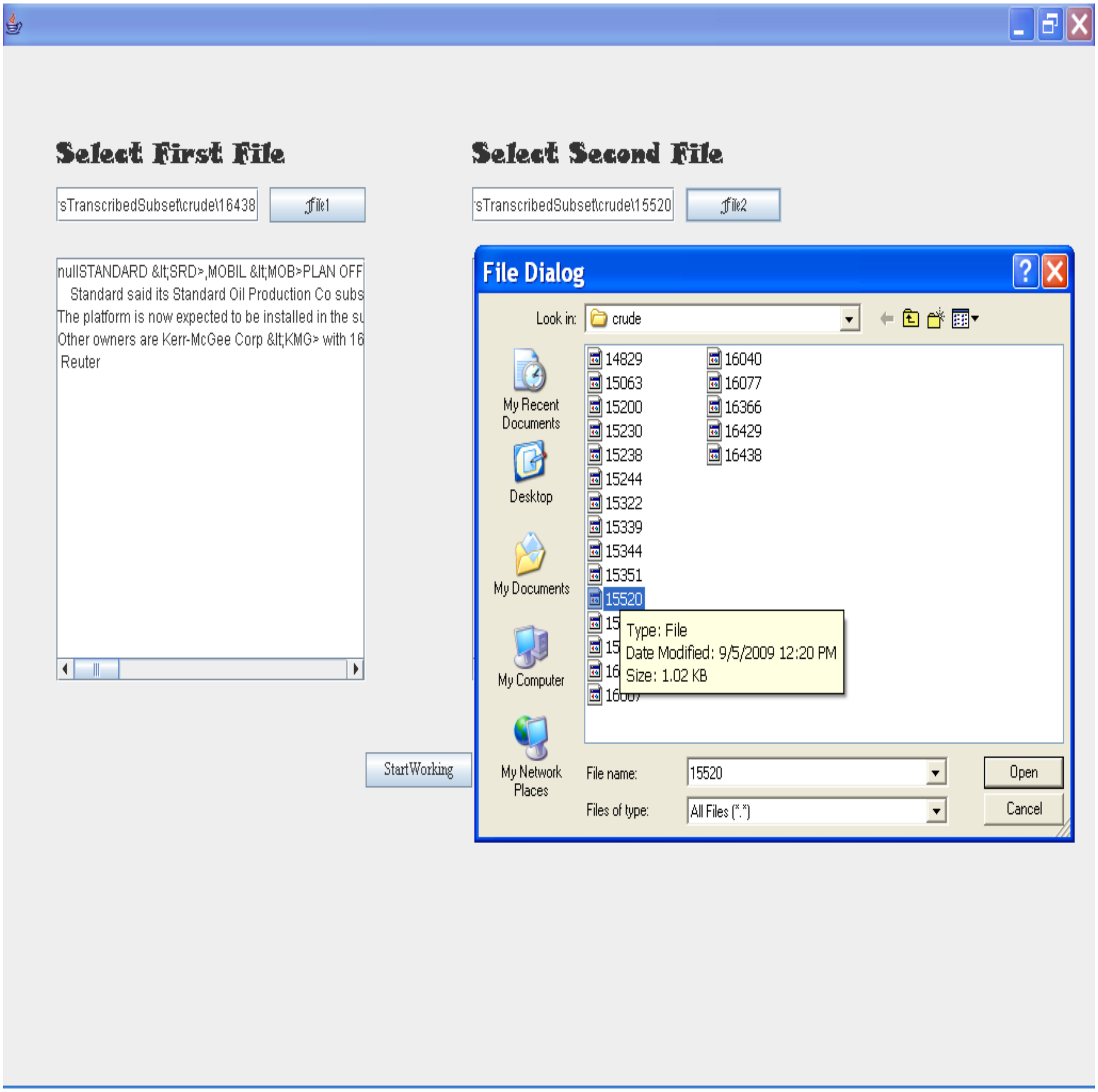
8.3 First File Reading



```
'sTranscribedSubset\crude\16438
```

```
nullSTANDARD &lt;SRD>,MOBIL &lt;MOB>PLAN OFF
  Standard said its Standard Oil Production Co subs
The platform is now expected to be installed in the su
Other owners are Kerr-McGee Corp &lt;KMG> with 16
 Reuter
```

8.4 Second File Browsing:



'sTranscribedSubset\crude\16438

'sTranscribedSubset\crude\15520

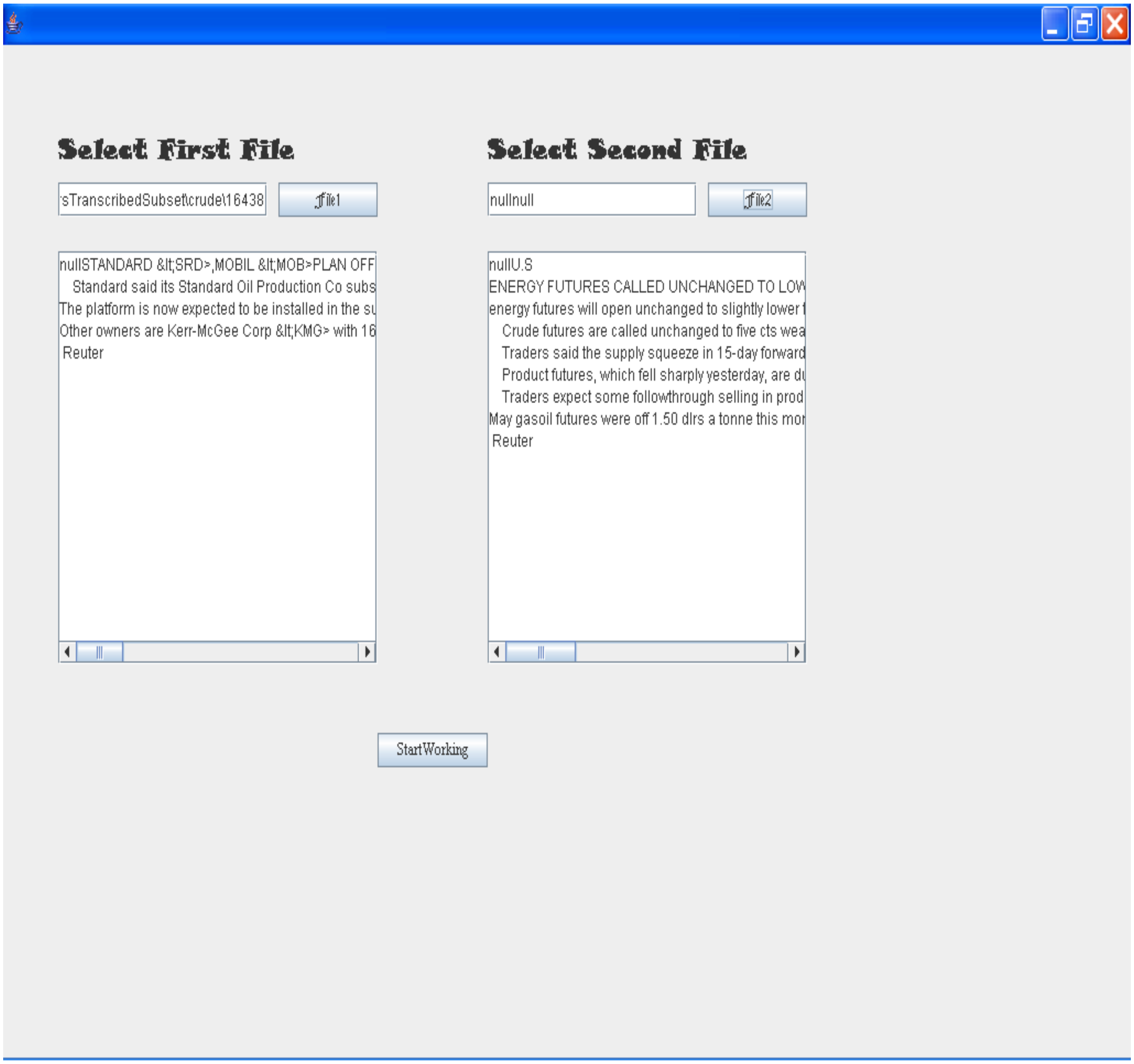
```
nullSTANDARD &lt;SRD>,MOBIL &lt;MOB>PLAN OFF
  Standard said its Standard Oil Production Co subs
The platform is now expected to be installed in the su
Other owners are Kerr-McGee Corp &lt;KMG> with 16
 Reuter
```

Type: File
Date Modified: 9/5/2009 12:20 PM
Size: 1.02 KB

File name: 15520

Files of type: All Files (*.*)

8.5 Second File Reading



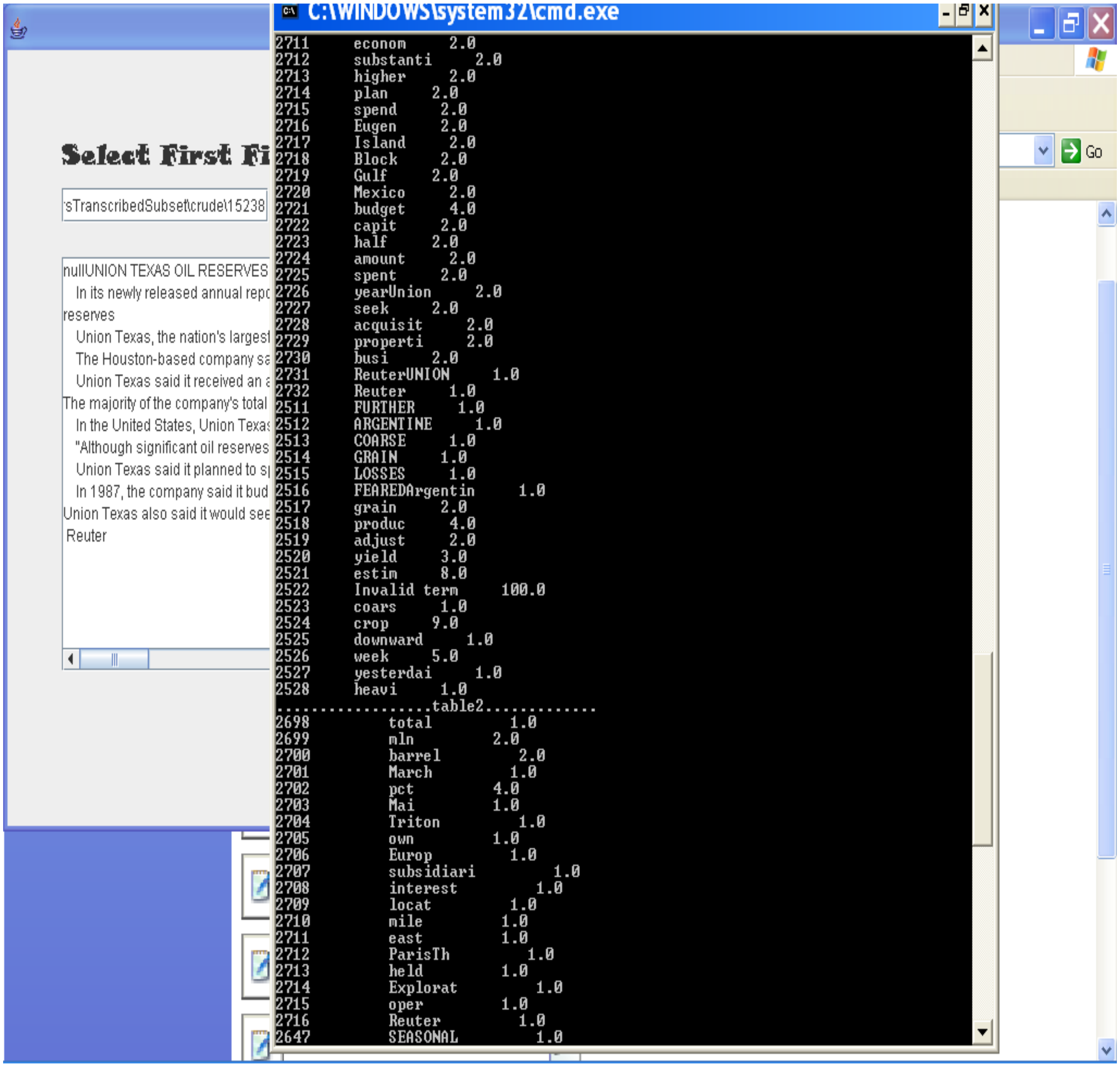
```
'sTranscribedSubset\crude\16438
```

```
nullnull
```

```
nullSTANDARD &lt;SRD>,MOBIL &lt;MOB>PLAN OFF
   Standard said its Standard Oil Production Co subs
The platform is now expected to be installed in the su
Other owners are Kerr-McGee Corp &lt;KMG> with 16
 Reuter
```

```
nullU.S
ENERGY FUTURES CALLED UNCHANGED TO LOW
energy futures will open unchanged to slightly lower t
   Crude futures are called unchanged to five cts wea
   Traders said the supply squeeze in 15-day forward
   Product futures, which fell sharply yesterday, are du
   Traders expect some followthrough selling in prod
May gasoil futures were off 1.50 dlrs a tonne this mor
 Reuter
```

8.6 Out Put On Command Prompt



sTranscribedSubset\crude\15238

nullUNION TEXAS OIL RESERVES
 In its newly released annual repo
reserves
 Union Texas, the nation's largest
 The Houston-based company sa
 Union Texas said it received an a
The majority of the company's total
 In the United States, Union Texas
 "Although significant oil reserves
 Union Texas said it planned to s
 In 1987, the company said it bud
Union Texas also said it would see
 Reuter

```
2711     econom     2.0
2712     substanti     2.0
2713     higher     2.0
2714     plan     2.0
2715     spend     2.0
2716     Eugen     2.0
2717     Island     2.0
2718     Block     2.0
2719     Gulf     2.0
2720     Mexico     2.0
2721     budget     4.0
2722     capit     2.0
2723     half     2.0
2724     amount     2.0
2725     spent     2.0
2726     yearUnion     2.0
2727     seek     2.0
2728     acquisit     2.0
2729     properti     2.0
2730     busi     2.0
2731     ReuterUNION     1.0
2732     Reuter     1.0
2511     FURTHER     1.0
2512     ARGENTINE     1.0
2513     COARSE     1.0
2514     GRAIN     1.0
2515     LOSSES     1.0
2516     FEAREDArgentin     1.0
2517     grain     2.0
2518     produc     4.0
2519     adjust     2.0
2520     yield     3.0
2521     estim     8.0
2522     Invalid term     100.0
2523     coars     1.0
2524     crop     9.0
2525     downward     1.0
2526     week     5.0
2527     yesterdai     1.0
2528     heavi     1.0
..................table2.............
2698         total         1.0
2699         mln         2.0
2700         barrel         2.0
2701         March         1.0
2702         pct         4.0
2703         Mai         1.0
2704         Triton         1.0
2705         own         1.0
2706         Europ         1.0
2707         subsidiari         1.0
2708         interest         1.0
2709         locat         1.0
2710         mile         1.0
2711         east         1.0
2712         ParisTh         1.0
2713         held         1.0
2714         Explorat         1.0
2715         oper         1.0
2716         Reuter         1.0
2647         SEASONAL         1.0
```

9 References

[1] V.A.Narayana,P.Premchand and A.Govardhan,(2009)" A Novel and Efficient Approach For Near Duplicate Page Detection in Web Crawling". IEEE International Advance Computing Conference.

[2] Arasu, A., Cho, J., Garcia-Molina, H., Paepcke, A., and Raghavan, S., (2001) "Searching the web", ACM Transactions on Internet Technology, vol. 1, no. 1: pp. 2-43.

[3] Brin, S., Davis, J., and Garcia-Molina, H., (1995) "Copy detection mechanisms for digital documents", In Proceedings of the Special Interest Group on Management of Data (SIGMOD 1995), ACM Press, pp.398–409, May.

[4] Broder, A. Z., Glassman, S. C., Manasse, M. S. and Zweig, G., (1997) "Syntactic clustering of the web", Computer Networks, vol. 29, no. 8-13, pp.1157–1166.

[5] Bacchin, M., Ferro, N., Melucci, M., (2002) "Experiments to evaluate a statistical stemming algorithm", Proceedings of the international Cross- Language Evaluation Forum Workshop, University of Padua at CLEF 2002.

[6] Broder, A. Z., Najork, M., and Wiener, J. L., (2003) "Efficient URL caching for World Wide Web crawling", Proceedings of the 12th international conference on World Wide Web, pp. 679 - 689.

[7] Berendt, B., Hotho, A., Mladenić, D., Spiliopoulou, M., and Stumme, G., (2004) "A Roadmap for Web Mining: From Web to Semantic Web", Lecture Notes in Artificial Intelligence, Springer-Verlag Heidelberg, Berlin; Heidelberg; New York, Vol. 3209, pp. 1-22. ISBN: 3-540-23258-3.

[8] Bayardo, R. J., Ma, Y., and Srikant, R., (2007) "Scaling up all pairs similarity search", In Proceedings of the 16th International Conference on World Wide Web, pp.131-140.

[9] Bar-Yossef, Z., Keidar, I., Schonfeld, U., (2007) "Do not crawl in the dust: different URLS with similar text", Proceedings of the 16th international conference on World Wide Web, pp: 111 - 120.

[10] Balamurugan, S., Newlin Rajkumar, and Preethi, J., (2008) "Design and Implementation of a New Model Web Crawler with Enhanced Reliability", Proceedings of world academy of science, engineering and technology, volume 32, August, ISSN 2070-3740.

[11] Cooley, R., Mobasher, B., Srivastava, J., (1997) "Web mining: information and pattern discovery on the World Wide Web", Proceedings of the Ninth IEEE International Conference on Tools with Artificial Intelligence, pp: 558 - 567, 3-8 November, DOI: 10.1109/TAI.1997.632303.

[12] Cho, J., Garcia-Molina, H., and Page, L., (1998) "Efficient crawling through URL ordering", Computer Networks and ISDN Systems, vol. 30, no. 1-7: pp. 161-172.

[13] Cho, J., Shivakumar, N., and Garcia-Molina, H., (2000) "Finding replicated web collections", ACM SIGMOD Record, Vol. 29, no. 2,pp. 355 - 366.

[14] Charikar, M., (2002) “Similarity estimation techniques from rounding algorithms”. In Proc. 34th Annual Symposium on Theory of Computing (STOC 2002), pp: 380-388.

[15] Chakrabarti, S., (2002) “Mining the Web: Discovering Knowledge from Hypertext Data”, Morgan-Kaufmann Publishers, pp: 352, ISBN 1-55860-754- 4.

[16] Conrad, J. G., Guo, X. S., and Schriber, C. P., (2003) “Online duplicate document detection: signature reliability in a dynamic retrieval environment”, Proceedings of the twelfth international conference on Information and knowledge management, New Orleans, LA, USA, pp. 443 - 452.

[17] Castillo, C., (2005) “Effective web crawling”, SIGIR Forum, ACM Press, Volume 39, Number 1, N, pp.55-56.

[18] David, G., Jon, K., and Prabhakar R., (1998) “Inferring web communities from link topology”, Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space-structure in hypermedia systems, Pittsburgh, Pennsylvania, United States, pp.225 - 234.

[19] Deng, F., and Rafiei, D., (2006) “Approximately detecting duplicates for streaming data using stable bloom filters,” Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pp. 25-36.

[20] Fetterly, D., Manasse, M., and Najork, M., (2003) “On the evolution of clusters of near-duplicate web pages”, Proceedings of the First Conference on Latin American Web Congress, pp. 37.

[21] Garcia-Molina, H., Ullman, J.D., and Widom, J., (2000) “Database System Implementation”, Prentice Hall.

[22] Gibson, D., Kumar, R., and Tomkins, A., (2005) “Discovering large dense subgraphs in massive graphs”, Proceedings of the 31st international conference on Very large data bases, Trondheim, Norway, pp. 721 - 732.

[23] Gurmeet S. Manku, Arvind Jain, Anish D. Sarma, (2007) “Detecting nearduplicates for web crawling,” Proceedings of the 16th international conference on World Wide Web, pp: 141 - 150

[24] Hoad, T. C., and Zobel, J., (2003) “Methods for identifying versioned and plagiarized documents”, JASIST, vol. 54, no. 3, pp.203–215.

[25] Henzinger, M., (2006) “Finding near-duplicate web pages: a large-scale evaluation of algorithms,” Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 284-291.

[26] Kosala, R., and Blockeel, H., (2000) “Web Mining Research: A Survey,” SIGKDD Explorations, vol. 2, Issue. 1.

[27] Lovins, J.B. (1968) “Development of a stemming algorithm”. Mechanical Translation and Computational Linguistics, vol. 11, pp. 22-31.

[28] Lim, E.P., and Sun, A., (2006) “Web Mining - The Ontology Approach”, International Advanced Digital Library Conference (IADLC’2005), Nagoya University, Nagoya, Japan.

[29] Menczer, F., Pant, G., Srinivasan, P., and Ruiz, M. E., (2001) “Evaluating topic-driven web crawlers”, In Proceedings 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,pp. 241-249.

[30] Metzler, D., Bernstein, Y., and Bruce Croft, W., (2005) “Similarity Measures for Tracking Information Flow”, Proceedings of the fourteenth international conference on Information and knowledge management, CIKM’05, October 31.November 5, Bremen, Germany.

[31] Pazzani, M., Nguyen, L., and Mantik, S., (1995) “Learning from hotlists and coldlists: Towards a www information filtering and seeking agent”, Proceedings, Seventh International Conference on Tools with Artificial Intelligence, pp. 492 - 495.

[32] Pant, G., Srinivasan, P., Menczer, F., (2004) “Crawling the Web”. Web Dynamics: Adapting to Change in Content, Size, Topology and Use, edited by M. Levene and A. Poulovassilis, Springer-verlog, pp: 153-178, November.

[33] Pandey, S.; Olston, C., (2005) “User-centric Web crawling”, Proceedings of the 14th international conference on World Wide Web, pp: 401 – 411.

[34] Shoaib, M., Arshad, S., “Design & Implementation of web information gathering system,” NITS e-Proceedings, Internet Technology and Applications, King Saud University, pp.142-1.

[35] Singh, A., Srivatsa, M., Liu, L., Miller, T., (2003) “Apoidea: A Decentralized Peer-to-Peer Architecture for Crawling the World Wide Web”, Proceedings of the SIGIR 2003 Workshop on Distributed Information Retrieval, Lecture Notes in Computer Science, Vol. 2924, pp. 126-142.

10 Appendices

1 Event-Driven programming will be based on the events, which will be performed (fired) by end user.

2 Each & every program can be attached with some event. So whenever the event occurs, the corresponding program will be executed. e.g.: button click

3 JAVA is a powerful front-end, with flexible coding methods.

4 JAVA Supports Client-Server technology.

5 JAVA Supports Internet programming (HTML,DHTML & JAVASCRIPT) & Web designing

6 Supports multiple databases (backend tools)

We can access data from multiple back ends (oracle, access, FoxPro,

Excel lotus, paradox, etc)

1 Three types of database engines (database programming concepts) to access local & remote databases

a. DAO PROGRAMMING (Data access Object)

b. RDO PROGRAMMING (Remote data object)

c. ADO PROGRAMMING (Active data object)

2 Supports the development of user controls

3 Provides commands for file handling

4 Provides number of special controls for creating interactive windows based applications.

Variables:

Named Memory Location is called variable. It is used to refer any values.

The Declaration of Variables:

Ex: dim|private|public <Variable name> as <data type>

dim / Private / Public A as Integer

Private variable:

Can be accessed only within the current form(Local of Particular Form).

Public variable:

Can be accessed by all forms of the project(Global for all Forms in Current Project)

Private & public modifiers can be specified in the General 'Declarations' part of Form

Within a subroutine, when we declare variable, it should start only with "Dim".

Based on the type of subroutine, variable is considered as private or public.

MDI (Multiple Document Interface):

In Many Application, the MDI is used to Handle more than one window at a time.

For Including this, the JAVA 7.0 Introduce, the new form such as Called "MDI Form"

For Achiving this, the user have to add the New Form called MDI Form.

MDI Form:

- This form is used to organize the other forms.
- Each project can contain only one MDI Form maximum.
- With this form, the user can place any Menu Bar or Tool Bar only.

  (no other Normal Controls).

Open Database connectivity:

ODBC is the concept of connecting remote database with our

client application.

1. User DSN (Data source Names):

Can be accessed only by the current winNT user (Login user)

2. System DSN:

Can be accessed only within the client system

3. File DSN:

File having extension .dsn, stores the ODBC connectivity details

and it can be used by any user in any system by copying the file.

Data manipulations in JAVA

For performing database manipulations in JAVA, we need to connect with any database.

Database is of two types

1. Local database
2. Remote database

1. Local database:

Database resides in local (current) system.

Eg: Ms-Access, MS-Excel, Visual FoxPro, etc

2. Remote database:

Database resides in remote location (system).

Eg: Oracle, Sybase, SQL-server, etc

For connecting different databases, JAVA provides three different kind of data access methods

1. DAO - Data Access Object