Spring 2014

Concept Based Semantic Search Engine

Pradeep Roy
San Jose State University

Concept Based Semantic Search Engine

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Pradeep Roy

May 2014

The Designated Committee Approves the Writing Project Titled

CONCEPT BASED SEMANTIC SEARCH ENGINE

By

Pradeep Roy

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2014

Dr. Tsau Young Lin	Department of Computer Science
Dr. Jon Pearce	Department of Computer Science
Mr. Amitaba Das	IBM SVL Research Center

ABSTRACT

In the current day and age, search engines are the most relied on and critical ways to find out information on the World Wide Web (W3). With the ushering in of Big Data, traditional search engines are becoming inept and inadequate at dishing out relevant pages. It has become increasingly difficult to locate meaningful results from the mind boggling list of returns typical of returned search queries. Keywords, often times, alone cannot capture the intended concept with high precision. These and associated issues with the current search engines call for a more powerful and holistic search engine capability. The current project presents a new approach to resolving this widely relevant problem - a concept based search engine. It is known that a collection of concepts naturally forms a polyhedron. Combinatorial topology is, thus, used to manipulate the polyhedron of concepts that are mined from W3. Based on this triangulated polyhedron, the concepts are clustered together based on primitive concepts that are geometrically, simplexes of maximal dimensions. Such clustering is different from conventional clustering since the proposed model may have overlapping. Based on such clustering, the search results can then be categorized and users allowed to select a category more apt to their needs. The results displayed are based on aforementioned categorization thereby leading to more sharply gathered and, thus, semantically related relevant information.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my academic advisor Dr. Tsau Young Lin, for his meticulous guidance and endless encouragement.

I am indebted to Dr. Jon Pearce and Mr. Amitabha Das for serving on my Masters' committee.

I could not have asked for a more relevant and supportive team of professors to keep me on the right track. Words cannot express my gratitude to my family who has always been there for me, and to the precious friends I made during my studies here.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Concept based search: Search engines are assumed to get a user the required information he is looking for from a stupendously huge collection of web pages and documents on the World Wide Web (W3). Research conducted by Cyveillance - a Washington D.C.-based market research firm - has revealed that more than 6,000,000 public pages are getting added to the W3 every single day. A search engine should be equipped to retrieve correct and meaningful results from this huge ever growing database which evidently doubles in volume every year. However, current search engines often return a list of documents based on individual keywords provided by users. Actually a user often want to retrieve author's concept and idea, in order to do so he supplies a list of keywords in the search query. The primary goal of this project is to develop a system that will capture the user's idea through his list of key words. Our first task is to identify the possible concepts that are in user's mind, then extract all articles containing these concepts. For instance the program should be able to relate "Zookeeper" to big data or Hadoop, not to a person who works as zookeeper, "Sound Investment" to finance not to music, "Holy Grail" to mythology not to the song "holy grail" etc. We believe that documents are carrier of ideas expressed by an author. According to this work the ideas are highly structured "concepts". We consider a common idea (highly repeated key words) as knowledge. So the main task of our project is to index an organized set of knowledge that are buried in "Big Data" of web. Our project consists of (1) Mining concepts, (2) storing the structured concepts, termed as KnowledgeBase in a persistent storage, (3) any required concepts and ideas of a user can be retrieved with high precision.

The scope of this project is to prove the credibility of algorithm proposed and creation of KnowledgeBase using sample documents and proposing a new approach to show the search results to the user.

2. RELATED WORK

W3 is the largest library of documents that are translated by machines and presented to the users. This has evolved from hypertext systems, the problem is that anyone can add documents to it. Therefore the quality of the documents cannot be guaranteed. Current W3 contains huge information and knowledge but machines only work well for delivery of the contents and are not responsible for extracting knowledge. People have to do that manually.

Semantic web is effort led by World Wide Web Consortium which is evolved in order to enhance current system of web in order to make sure that computers will have the capability to process the information and knowledge presented on W3. Semantic web is intended to create a huge knowledgbase and to share data instead of documents.

Semantic web architecture consists of XML, RDF, OWL and RIF.

2.1 XML: Extensible Markup Language layer along with XML-namespace and XML-schema definition imposes a common syntax usage in semantic web. It is a markup language which is generally used for creating documents containing structured information. XML documents have elements which can be nested and elements may contain attributes and contents. It allows to create many different markup vocabularies in one document. XML schema allows to express schema for a particular type of XML documents.

2.2 RDF: Resource Description Framework is the core data representation format made for semantic web. RDF presents the information about resources in graphs. It is intended to represent metadata about the resources like author, title. RDF can also be used for storing other data as well. It works on triplets subject-predicate-object that creates data graph. Semantic web data uses RDF as a representation language.

RDF works to describe the description of the graph formed by the above triplets. Anyone can describe and define the vocabulary of terms used. To allow standardized description of taxonomies and other ontological constructs, a RDF Schema (RDFS) was created together with its formal semantics within RDF. RDFS can be used to describe taxonomies of classes and properties and use them to create lightweight ontologies[11].

2.3 OWL: Web Ontology Language extends RDF & RDFS. OWL adds semantics to the schema. The primary goal of OWL is to bring the reasoning and expressive power of description logic (DL) to the semantic web. We cannot express everything from RDF to DL. For example, the classes of classes are not permitted in the (chosen) DL, and some of the triple expressions would have no sense in DL[11]. OWL is syntactic extension of RDF. To overcome the above problem, and in order to allow layering in OWL there are 3 species of OWL are present.

OWL lite is used to express simple constraints and taxonomy e.g. 0 & 1 cardinality. OWL lite is the simplest language a maps to description logic.

OWL DL supports maximum expressiveness along with computational completeness & decidability.

OWL Full has no constraints for expressiveness but it also not guaranteed to have any computational properties. It is created with full OWL vocabulary but it doesn't force nay syntactic constraints so it can use full syntactic freedom of RDF.

These three languages are layered in a sense that every legal OWL Lite ontology is a legal OWL DL ontology, every legal OWL DL ontology is a legal OWL Full ontology, every valid OWL Lite conclusion is a valid OWL DL conclusion, and every valid OWL DL conclusion a valid OWL Full

conclusion[11].Inverse of these above relations is generally true. Every OWL ontology is a valid RDF document but here also the inverse is not true.

3. PROPOSED APPROACH AND WORK

The work described in the report provides a fresh approach to develop a clustering technology which discovers the semantics of the documents present on W3 by applying text analysis. Through crawling the web and process the documents which finally creates a KnowledgeBase after processing. The KnowledgeBse will contain the keywords which represents the concept contained by the documents processed. It will help in creating a semantically aware concept based search engine. The approach described here doesn't require any special tags in web pages to extract the knowledge and also can work on existing web pages. This new approach to semantic learning is the main advantage over the previous described methodologies which require the authors to tag their pages with special constructs using RDF or the other methodologies defined by semantic web, which are labor intense. The work uses method derived by Dr. TY Lin's research paper on Concept Analysis in 2006(Tsau Young Lin, Albert Sutojo and Jean-David Hsu).This work focuses on applying the idea presented and create a concept based semantic search engine along with providing a new approach to present search results to user in which user can see the main concepts contained by the papers so he doesn't need to go through the documents to see if that is a correct result of his search query. The search engine works on asking user for the intended concept he is looking for rather than guessing.

Dr. Lin (2006) showcases mapping the knowledge contained in sequence of keywords (not necessary to be consecutive in text) into a mathematical model. He has suggested that semantics of keywords together can be mapped to Simplicial complex (a triangulated polyhedral Complex) [1].

Once all the keywords contained a document are transformed into this model, it can be treated as a knowledge base for semantic search.

3.1 Important Terms

Let's take top-down approach to understand the important terms used in this work.

3.1.1 KnowledgeBase: We are here referring to this term as an organized large collection of concepts. According to our proposal, after crawling and processing of all the pages on W3 we will be able to successfully extract all of the possible concepts or knowledge available in the web universe. After which, there will be a negligible possibility of new knowledge coming in to the KnowledgeBase. This assumption helps in computation of concepts for successive iterations. A KnowledgeBase will be formed by processing many documents through the program.

3.1.2 Document: A document here in this work can be defined as a dataset of meaningful text which contains one or many "concepts".

3.1.3 Concept: If we look at dictionary meaning of concept, then a concept is an abstract idea; a general notion. Here in this work a "concept" is the term used to represent the way humans express their thoughts, idea, or an abstract form of knowledge. Concept is the underlying meaning provided by the collection of keywords which is almost different from the individual words. If those words repeat for many times then they together represents a "concept". A point worth noting here is that a concept can also be expressed by single keyword. For example:

I. Computer Virus (A computer virus is a type of malware that, when executed, replicates by inserting copies of itself (possibly modified)

into other computer programs, data files, or the boot sector of the hard drive).

II. Biological Virus (A virus is a small infectious agent that replicates only inside the living cells of other organisms.)

III. Wall Street (Wall Street is the financial district of New York City) etc.

IV. White House

V. Pig Latin

The words which are together and doesn't represent any concepts are called as keywords set.

3.1.4 Keywords set: These sets are group of words which are close to each other in a defined proximity like a paragraph. The program has a limit to consider upto 6 tokens in a keywords set. This limit can be changed to lower value at run-time.

3.1.5 Keyword or Token: It is representing any English word with a definite meaning. E.g Virus, Biology, Computer, Disease, Wall, Street, Zookeeper, Pig, Latin etc. Once all the stop words are removed from tokens, tokens together form keywords set which further can qualify as concept if it has some properties contained.

3.1.6 Stop words: Some extremely common words which would appear to be of little value in helping select documents matching a user need and are excluded from the search vocabulary entirely.

Stop words are a crucial part of data mining and text processing endeavors. They are generally determined by collective frequency (the number of times each term appears in documents) and then taken out as the most frequent words. Generally, stop words should be picked in terms of their semantics and context. A list is prepared using the most frequent words called as 'stop list'. Removing of stop list from the documents being processed gives us two

advantages: a) it reduces computation for non-desired and senseless words and b) it helps in better extraction of concepts and knowledge from documents[4].

Generally, stop words are parts of speech which are used to link the words together in order to express the desired meaning E.g. Adjective, helping verb, conjunction, the prepositions, articles, modifiers etc.

Let's take an example here: The phrase "The big data technology is growing" contains two stop words "The" and "is". In typical IR scenarios, it has been observed that using a long stop list is better in getting better results. Web search engines generally do not use stop lists. In this work, a thorough stop list of 500 words has been compiled by merging different stop lists found over the internet.

3.2 Algorithm

The following algorithms feature in this work in order to get accurate results from data mining:

3.2.1 Apriori

The Apriori Algorithm helps in mining frequent keywords sets

Key Points

- **Frequent Keywords:** The keywords set which is repeated for at least threshold number of times
- **Apriori Property:** It is described as any subset of frequent keywords set will be frequent keyword(s) set.

The Algorithm in nutshell

First, find the frequent keywords sets that are the sets of keywords which possess minimum support.

- According to Apriori principle it is necessary that given a subset of a frequent key words must also be a frequent key words set. We have used negation of Apriori principle in our project in order to eliminate keywords set which are not going to be frequent during the candidate generation. This theory helps in removing possible concepts which are not frequent. Application of Apriori helps in reducing computation in early stage of token processing and concept generation.
- i.e., if {A, B, C} is a frequent keywords set, then all {A, B}, {B, C} and {A, C} will be a frequent keyword set. The negation would be if keyword {A} is not frequent set then any keyword set containing {A} would never be frequent meaning keywords sets {A, B}, {A, C} and {A, B, C} will never be frequent. This helps in reducing computation of token/keywords sets which will never qualify as concepts.

3.2.2 Term Frequency Inverse Document Frequency (TF-IDF)

Term Frequency Inverse Document Frequency (TF-IDF) is a value which is often used in IR keywords extraction. This work focuses on both these areas and, so, TF-IDF is used in our algorithm for extracting better and related concepts. The TF-IDF value is a statistical measure used to find out how important a token is to a document in a collection of documents. The importance of token increases in proportion to the number of times it appears in the document but it is also inversely proportional if a token repeats in large frequency in almost all the documents and it is a stop word.

Changing TF-IDF values are used as a primary tool in scoring the relevance of a document to the searched query by several search engines.

The simplest TF-IDF function is computed by summing it for each token. Several TF-IDF functions are available but we have used the following one.

As previously mentioned, TF-IDF can successfully filter stop words from all extracted tokens. Here is an example to understand this:

TF-IDF Computation

Typically, the TF-IDF values are calculated by two terms:

a. **Term Frequency (TF):** It is the frequency which represents the no. of times a token repeats in the document in process, to the total no. of words in the document. Term Frequency, which calculates the frequency of any token in a document. As every document is different in size it is highly likely that a token repeats more in longer documents than the smaller documents. That is the reason why term frequency is divided by document length i.e total no. of tokens in document in order to normalize the value.

   For a keywords set *K = {k1,k2,k3,k4….kn}* let *d* is the document containing *K*.

   For a document set of all documents D, d is subset of D

   F(K,d) is the frequency which represents the number of times K is repeated in d.

   TF(k,d) = F(K,d)/max(F{K,d})

b. **Inverse Document Frequency (IDF):** It is the natural logarithmic value of the no. of documents to the no. of documents in which the token in consideration is repeated. IDF measures the importance of a token.

   During the TF calculation all tokens are equally important. However it is known that stop words e.g "the", "for", and "have" etc. may appear alot but are of very less importance. In order to scale up the rare and important tokens we need to weigh down the very frequent tokens by computing IDF:

IDF(K,D) = ln(no. of documents D/ no. of documents containing K)

TF-IDF (K,D) = TF(K,d)*IDF(K,D)

E.g.

Let's consider a document having 500 words and token "hadoop" repeats for 10 times in it.

TF (hadoop) = (10/500) = 0.02

Now if we have 1 million documents to process and token "hadoop" appears in 1000 of the documents then IDF would be

IDF = ln(1000000/1000) = 6.907755279

Thus TF-IDF for "hadoop" is the product of TF*IDF

TF-IDF(hadoop) = 0.02* 6.907755279 = 0.138

3.2.3 Simplicial Complexes and polyhedra in search

This project proposes the method of capturing human concepts or simply concepts using simplicial or polyhedral complexes. According to Dr. TY Lin,

A simplicial complex C consists of a set {v} of vertices and a set {s} of finite nonempty subsets of {v} called simplexes such that

- *Any set consisting of one vertex is a simplex.*
- *Closed condition: Any nonempty subset of a simplex is a simplex.*

Any simplex s containing exactly q + 1 vertices is called a q-simplex. We also say that the dimension of s is q and write dim s=q. We will refer to C as a non-closed simplicial complex, if the closed condition is not fulfilled for all its constituting simplexes[1].

Simplicial Complex in Our Project

The method used in the project to extract concepts from documents is textual processing. The program looks for high frequency keywords which are either co-

occurring or may be little distant but not as much as a paragraph. We choose paragraph size to be 25-30 keywords.

The keywords captured by the program here are called as keywords set. As mention earlier, it is not necessary for keywords to be close to each other but they must satisfy the precondition for closeness. When a keywordset repeats itself more times than a threshold frequency in a document, it is stored by the program for further processing else it is discarded and the next keyword set is picked up and can be treated as human concept.

In the simplicial complex model of human concepts, all the vertices are taken by a keyword/token and an edge joining two keywords together representing a human concept. Each keyword set thus here forms an n-simplex (granule). Here n represents the number of distinct keywords in the keyword sets. The geometry represented by many keywords or keyword sets together is called as simplicial complex of human concept or knowledge complex. The point to be noted here is that even a single disjoint keyword can also represent a concept. We are mainly interested in maximal simplex. The figure below represents a keyword simplicial complex.

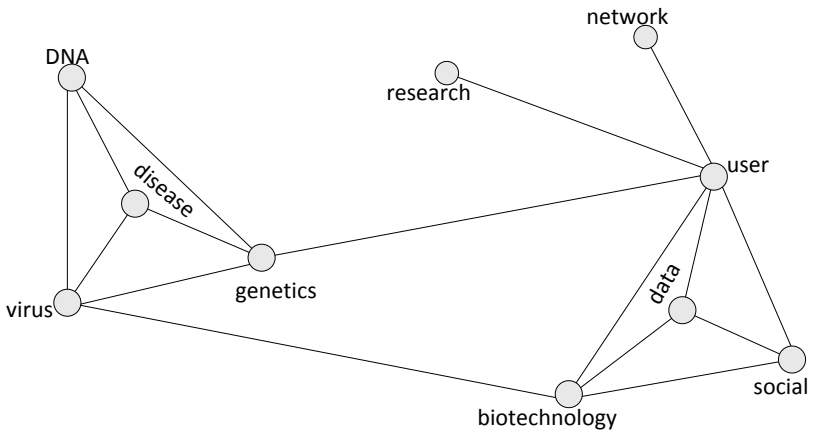

Figure-1-Simplicial Complex for Keywords set

The document contained in the simplex is representing a knowledge regarding biotechnology research paper.

Following simplicials can be found out of the above diagram.

1. Tetrahedron - 2 Maximal 3 Simplices
   a) DNA genetics virus disease
   b) Biotechnology user social data
2. Triangle – 3 maximal 2 simplices
   a) User virus biotechnology
3. 3 Maximal 1 simplices
   a) User research
   b) User network
   c) User genetics

4 PROJECT ARCHITECTURE AND COMPONENTS

4.1 High Level Architecture

The following high level architecture diagram explains the flow of data contained in the documents from crawling W3 through to storage in KnowledgeBase.

The components include WebCrawler, Parser, Concept Extractor, RawTokens database, and KnowledgeBase. The graphical user interface used to represent user search results is not a part of this work but is definitely an exploratory avenue; the main objective of this project is to successfully extract the concepts from documents and to create a KnowledgeBase. Here SQL queries are used to return results to the user.

All components and their associated roles are explained in the following section:

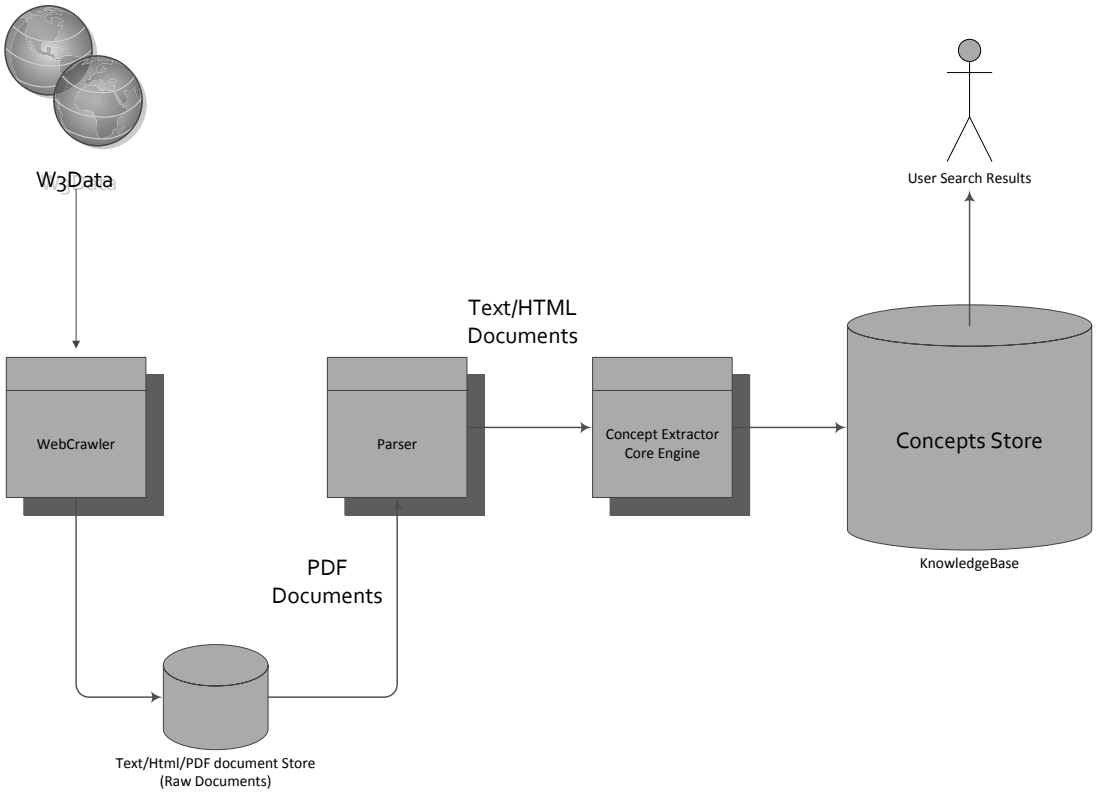

Figure-2-High level Architecture

4.2 Pseudo code

```
Load from RawTokens database
For each k, from k=1
{
     For each of the document in repository
     {
     parse and tokenise document d;
     tokens <= remove stop words;
     tokens <= lowercase and perform stemming;
     tokens <= remove newly formed stop words;
     keywords sets <= permute paragraph;
     keywords sets <= prune using TF-IDF;
     keywords sets <= prune non maximal simplices;
     store concepts in KnowledgeBase;
     if no more new concepts
          exit the loop;
     }
}
```

4.3 Components and System Architecture

4.3.1 WebCrawler

A Web crawler is program which automatically browses the W3 by following a defined system. Its objective is to index World Wide Web data. Crawlers have the capability to copy all data from the pages visited for processing by a search engine which then indexes all of the downloaded documents so that a user can search them very quickly. Crawlers can validate hyperlinks and HTML code. This work also includes creation of an intelligent crawler for better retrieval of pages from W3 and analysis and clustering of data according to the concepts contained in the documents. This crawler can identify specific URLs to visit and crawl on and leave out non-desired URLs as needed. It copies all the page data into simple text files along with the information of the urls from which data is collected. The crawler is provided with a list of URLs that it should visit, which are referred to as seeds. As the crawler goes through the seed URLs, it identifies all hyperlinks present on the page and append them to list of URLs to visit already being maintained by it. For instance, if we wish to create a concept cluster containing data from only educational websites, it's always better to provide '.edu' websites data to the core engine to process rather than all of WebSphere data. This can be incorporated in the designed crawler. One can also specify crawling only PDF files which mostly contain solid concepts regarding the text in them. As the core can process only text and html files (which are also a type of text files), the crawler can also be configured to leave out other data formats like PNG, JPEG, JPG, AVI, etc. The crawler created also has a capability to leave a URL if it is has already been crawled on before. This again helps in maintaining the quality of data and prevents repetition in the index and wrong calculation of TF-IDF.

4.3.2 Parser

The core engine of the project or concept extractor expects the data to be

stored in either text or html files but a web crawler can crawl many types like .doc, .pdf, jpeg, .png etc. of files from W3 and the data for which a user is looking for can be in any format. In order to use that content as well, we have to first pass the files through different parsers which will convert them to text files before feeding to the concept extractor. A PDF parser has been developed as a component to this project as our test data is mainly obtained from IEEE papers (PDF files). We have architecture in place that can be extended in order to add different parsers in the future without affecting present components.

4.3.2.1 PDF Parser

To prove that the algorithm is working properly and we are successfully able to capture the concepts from documents, the program core is fed with IEEE papers in PDF format but using IEEE papers in testing is very crucial as we already know the content of the papers and we can easily verify the results by looking at the knowledge base created by concepts extracted from those papers. We tested the results on different domain papers e.g. big data, social network, biotechnology, software defined networking to name a few (findings in the Results section). In order to process PDF files and cluster the concepts they contain, the relevant text needs to be extracted out of the PDF without losing the text format and sequence of sentences a PDF parser is also created as a part of the project. The Pdf parser is capable of keeping the text sequence intact even after converting the PDF file to a text file which is essential, because if the text sequence changes from the original PDF content, the concept or knowledge interpreted by the core will not be correct. Many commercial pdf converters doesn't have this capability and, therefore, the need for creating a parser increases. Pdf parser takes a directory containing the relevant PDF files as input and stores the content in a central directory provided from where the core engine can pick the files to process.

4.3.3 Tokenizer and Core Engine (Concept Extractor)

The tokenizer is the core of the algorithm which analyzes each document provided. It only takes text/html document as input and starts scanning each tokens contained within. It checks if the fetched token is a stop word or not and discards the token if that is the case and not processed again during subsequent scans. This helps in reducing computation time/effort and provides better concepts.

As explained earlier stop words are auxiliary words and don't contribute to the core knowledge of tokens. Once a token is qualified to process it is first stemmed to its root word. Doing so helps in recognizing one word in different forms as one. For stemming, we use the porter stemmer algorithm. After stemming, the word is checked for being a stop word because there can be instances where a word isn't considered a stop word owing to being in a different form. If a token survives the second stop word test iteration, it is de-capitalized to lower case in order to identify same tokens written in different cases. E.g. "Hadoop", "hadoop", "HADOOP" should map to single token. After the document is tokenized then the term frequency of each token is recorded and the document frequency is also stored. While processing next and successive documents the token fetched are first checked into the list of tokens created by previous documents, if the token is already present in the list then its token frequency is incremented along with document frequency. All the tokens qualifying the least minimum threshold are kept, considered Raw Concepts and stored.

The same process is repeated for finding two or more same tokens as well. But it should be noted that while finding two tokens all the possible combinations to form two tokens within a paragraph are considered in order to extract context of the words.

4.3.3.1 Pruning of tokens

The actual stage of the core algorithm starts here. The idea proposed in the project is mainly focused on finding co-occurring tokens or keywords that are closer to each other and as described earlier not necessarily back to back. By close we mean to capture the tokens which are close in the paragraph range defined. We also record the corresponding TF and DF for the tokens. This stage is a iterative process which says that we make the tokens to pass through multiple passes till we capture the concepts and no new concepts are found. It is a breadth first search which we are adapting to; it finds all one keywords token first followed by two keywords and so on. Each pass will result into a new K-keywords list starting at K=1. The algorithm exits the loop till the maximum of K is reached or no new keywords are found. We have put a limit of keeping maximum keywords length to 6 (experimental conclusion).

In order to save computation time it is necessary to complete the pruning of K keywords sets before starting K+1th iteration.

Breadth First

A legitimate question arises here: why is breadth considered first and not depth when we can simultaneously produce multiple K-length candidate concepts. The answer to the question is that we get a chance to migrate the K-length keywords out of the memory before starting K+1, which makes the computation less expensive otherwise we will end up keeping all of the K+n {n; 0 to 6} until the end of computation and will be using memory which is not an efficient way of handling the case.

To understand the problem let's take an example where we will see the exponential growth in data to process. To find all the co-occurring tokens of length K within a paragraph, we need to consider all of the possible K permutations for the token in paragraph length. "Permutation" used is not in its pure definition. Here we are not permuting the words, but we are

considering the permutations which retain the relative order of the words in a paragraph.

Doing so helps in making the data growth to a limited size and serves our purpose well.

Still, the growth of possible concepts is exponential.

For a 30-words paragraph, the possible 2-keyword concepts are:

2 Keywords possible concepts

= (30*29) + (29*28) + (27*26) +……. + (3*2) + (2*1)

3-keyword possible concepts

= (30*29*28) + (29*28*27) +……+ (4*3*2) + (3*2*1)

The above example is sufficient to prove that higher the value of K the data size quickly becomes unmanageable.

Apriori and TF-DIF, as mentioned before, are two tools from data mining techniques to reduce the candidate space.

According to Apriori algorithm a k-keyword set can be frequent iff all of its sub sets of keywords are frequent. Using the negotiation here we can say if a keywords set K is not frequent its supersets containing K keywords sets can never qualify as frequent. Using this fact, we can quickly determine the keyword sets which can never become frequent and drop them from possible concepts list. The breadth first search uses this principle in each of the iteration to produce a list of retained candidates. This is another reason of applying the breadth first strategy. The Apriori algorithm is helpful in reducing the computation and memory expensive operations.

As for TF-IDF, we have modified its original formula to suit our requirements. Once all available documents from the internet have been processed and stored in the knowledge base, the value of document frequency will no longer be necessary since all concepts will be there in the KnowledgeBase. But for creating a KnowledgeBase we first use both TF as well as DF.

TF-IDF is more important and relevant to our requirement because it helps in discovering frequent keyword sets or tokens and as well as it helps in eliminating keyword sets that are too frequent, too commonplace or are stop words. Instead of using the actual IDF formula we only use a simple document frequency (DF) percentage calculation for each keyword set, and then set a minimum and maximum threshold points to determine if a keyword set is to be discarded.

By using these two tools to reduce the candidate set, we have effectively turned around an insurmountable problem into a manageable one. The actual pruning threshold (user configurable) determines how much of the candidate space has been reduced. Using an aggressive pruning threshold would give us smaller candidate space but can risk discarding the important candidates (false negatives). On the other hand, using a relaxed pruning threshold might bring back the cardinality problem if almost all of the candidates are kept that will give us many false positives.

Another strategy we employ is the maximal simplex property, which helps in purifying the captured concepts or knowledge. According to maximal simplex property, if a K-length human concept is determined as frequent and kept as concept then any concept formed by subsets should be discarded as we have already captured and kept a more solid human concept.

Let's take an example from our analysis to better understand the usage of maximal simplex.

ID	Tokens	TokenCount	Frequency	Discard
1	big data analysi process	4	100	
2	big data analysi	3	210	Yes
3	Big data	2	300	Yes
4	data	1	410	Yes
5	big	1	380	Yes

Table-1-Simplicial complex maximal keywords sets

In above example as per maximal simplex property we should discard all subsets simplices “big”, “data”, “big data” and “big data analysi” as we have captured “big data analysi process” which is the maximal simplex. While the maximal simplex property is clear in mathematics it is very difficult and inappropriate to apply the same version in text processing. We have slightly modified this property in order to capture all possible human concepts. We are allowing the subsets of maximal simplex to qualify as human concept if satisfy in the following equation. Rather than blindly discarding the subsets, we take into account their frequency before deciding whether to discard them from human concepts. In this fashion, we determine if the subset is another concept altogether or not. Here is the formula:

Let M_K is maximal simplex of size **K**
M_{K-1} is subset of **M_K** of size **K-1.**

In order to keep M_{K-1} as a human concept it should have the following property. If Frequency F_{K-1} is equal or more than double the frequency F_K then keep both M_K and M_{K-1} else discard M_{K-1}.

Again consider above example and see which concepts can be retained.

ID	Tokens	TokenCount	Frequency	Discard
1	big data analysi process	4	100	No
2	big data analysi	3	210	No
3	Big data	2	300	Yes
4	data	1	410	Yes
5	big	1	380	Yes

Table-2-Simplicial complex maximal keywords sets our version

Here we are keeping both “big data analysi process” and “big data analysi” even though “big data analysi” is subset keywords. But as the frequency (210) of “big data analysi” is more than double of that (100) of “big data analysi process”; so we keep both. But we are discarding “Big data” because its

frequency (300) is less than double of frequency (210) of “big data analysi”. Same applies for “data” & “Big” concepts. By this modification in maximal simplex property, we are able to capture more solid human concepts.

After passing through pruning, a Token set or token can be considered as human concept and is stored in KnowledgeBase.

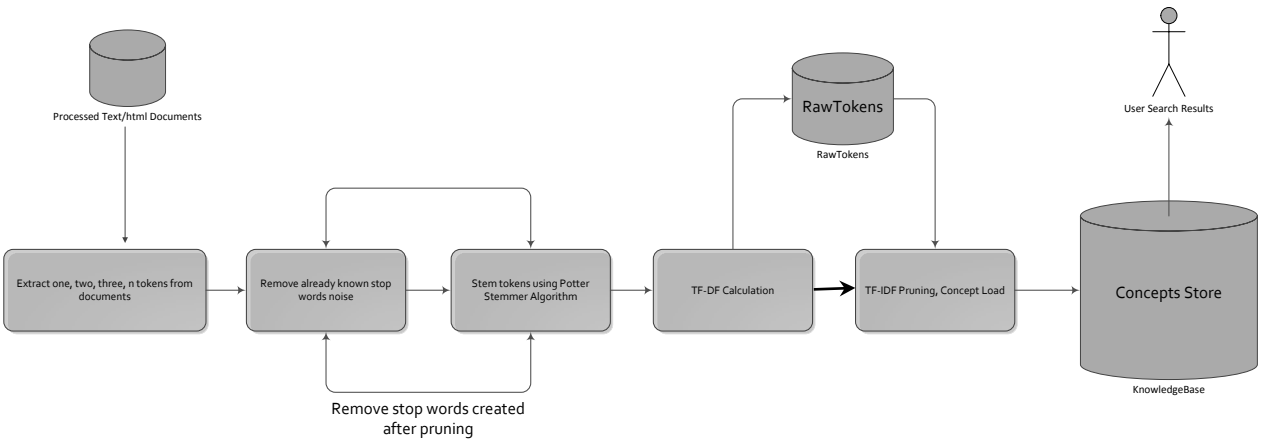

Figure-3-Core Engine

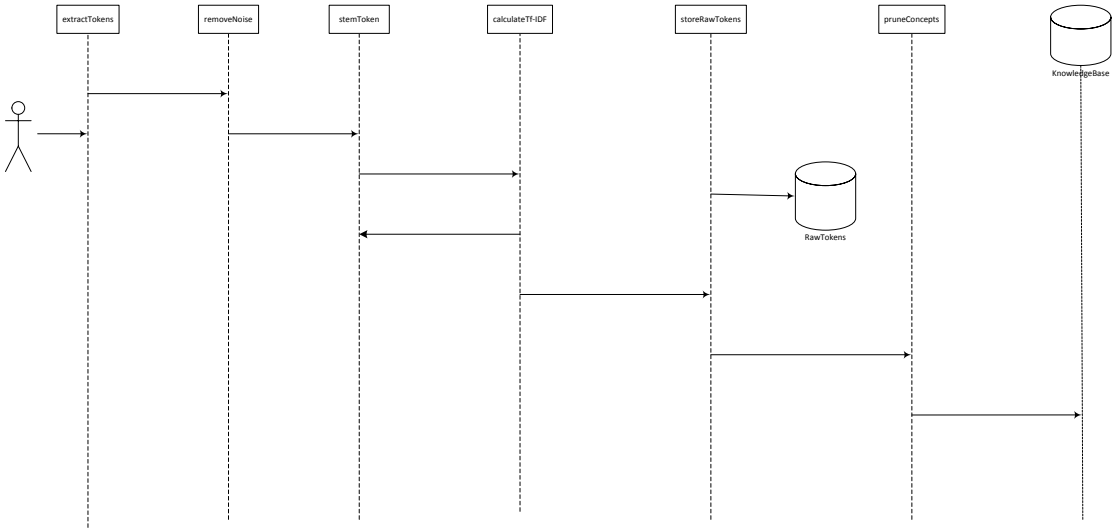

Figure-4-Sequence Diagram

4.3.4 Rawtoken Database

Until now we have created a knowledgeBase. However, if a new document is introduced, the existing system should be equipped to handle it.

For this, the following two approaches have been experimented with:

4.3.4.1 First Approach

In accordance with the existing approach used in this project, we already have a knowledge base setup, so why not just use that to cluster the concepts contained in the new document.

The following scenarios are possible:

I. The new document is entirely new and there is no previous knowledge stored about it in the already existing knowledge base.

II. The document is similar to the previously processed data sets and contains same concepts which we have already stored in the knowledge base.

III. The new document contains some old/previously captured concepts along with some new concepts.

All of these have one thing is common: from the project perspective, it will first be tokenized and then after processing it will be stored in knowledge base.

The overall approach in this situation of handling a new document can be illustrated by the steps explained below.

Here the first step is to read the document and stored in buffer for processing. After reading the document in to the buffer, all non-alphabetical characters are removed from it leaving behind only English words to process which saves processing time and memory. The first iteration of the processing is to remove STOP words from the raw text data - always read into lower case to remove ambiguity. After removing STOP

words the tokens are fed to the stemmer routine where the root/stem word of each token is identified. Once stemming is done, we again send the stemmed tokens for STOP words removal (to remove possible STOP words from popping up after stemming). Let's take an example:

Suppose we have tokens like, "User is working on big data and he is having a great time."

Step 1: Read tokens, lower case, and remove special characters if any.

user is working on big data and he is having a great time

Step 2: Remove STOP words.

user working big data having great time

Step 3: Stem the tokens and find out root words.

user work big data have great time

Step 4: Again remove STOP words.

user work big data great time

As we can see that removing STOP words helps in saving processing and memory in step 3 as the program has less tokens to process. But having step 4 for removing STOP words again helps in pure knowledge base creation as it removes the words which have become STOP words after stemming as here "having" was not a STOP word but "have" is a STOP word so it should be removed.

After processing the raw data into parsed tokens, we start counting the frequency for the tokens and insert into tokens database without removing any tokens even if they repeat for very less frequency. As a small document will have fewer repeating tokens but it may contain concepts which are new to knowledge base. So we keep all the tokens obtained by the new document. Doing this way and not calculating TFIDF for new documents again makes the processing very fast.

We repeat the same methodology for all new documents and we keep on

increasing the tokens database. Once any token or token set count in the table increases more than the threshold value for frequency it can be considered as human knowledge and will be inserted into knowledge base and will be removed from tokens table.

In this way we have an efficient and fast method to handle new documents.

4.3.4.2 Second Approach

To handle the situation of new document, we made some changes in the architecture of the system and modified the way it handles any tokens extracted from the documents. Here we added a new RawTokens database for all the tokens which can be candidate concepts but are not qualified as a concept because of two main reasons:

I. Either the documents set getting processed is not very large to pick those tokens as concept.
II. Or the value of TF-IDF is set so large that the tokens don't qualify for concepts.

According to the previous architecture, when we finish processing the tokens out of the documents, the tokens which are not qualified as concept are thrown out and then load the selected concepts to concepts database. But what if we keep all the potential tokens along with the information of frequency and document frequency for that token.

Doing so will help us in adding new documents or concepts in the table and results will improve significantly.

Suppose we are processing new documents and we extract all the possible concepts out of the document. Now we consider the RawTokens database to calculate TF-IDF for the new tokens. This can be explained by following example:

Tokens from old documents

Tokens	Frequency	IDF	TF-IDF	IsConcept
Big data	10	0.04	0.4	YES
Hadoop Analytics	20	0.03	0.6	YES
Data Warehousing	9	0.06	0.36	NO
Business Movement	8	0.01	0.08	NO

Table-3- Tokens from old documents

TF-IDF = 0.04

Concepts in KnowledgeBase

Tokens	Frequency	IDF	TF-IDF	IsConcept
Big data	10	0.04	0.4	YES
Hadoop Analytics	20	0.03	0.6	YES

Table-4- Concepts in KnowledgeBase

Tokens from new Documents

Tokens	Frequency	IDF	TF-IDF	IsConcept
Social Network	8	0.02	0.16	NO
Dataset	9	0.04	0.36	NO
Data Warehousing	3	0.04	0.12	NO

Table-5- Tokens from new documents

Tokens Table after considering RawTokens

Tokens	Frequency	IDF	TF-IDF	IsConcept
Social Network	8+0=8	0.02	0.16	NO
Dataset	9+0=9	0.04	0.36	NO
Data Warehousing	3+9=12	0.05	0.6	YES

Table-6- Tokens Table after considering RawTokens

Concept from new documents

Tokens	Frequency	IDF	TF-IDF	IsConcept
Data Warehousing	3+9=12	0.05	0.6	YES

Table-7- Concept from new documents

Updated Concepts

Tokens	Frequency	IDF	TF-IDF	IsConcept
Big data	10	0.04	0.4	YES
Hadoop Analytics	20	0.03	0.6	YES
Data Warehousing	3+9=12	0.05	0.6	YES

Table-8- Updated Concepts

By applying this architecture of RawTokens we have successfully dealt with two problems:

Firstly, we can handle large datasets of documents by processing small chunks from those documents. Consequent results will, thus, be accurate and the concept stored will be very strong as more data is used in the formation of the concepts and as the TF-IDF suggest larger the dataset is more correct the concepts are.

Second issue tackled is the better and accurate way of handling a new document. This contributes to dynamic growth of the KnowledgeBase.

This approach also does away with the problem of re-tokenizing entire documents in order to find concepts as we are already considering old documents from RawTokens table.

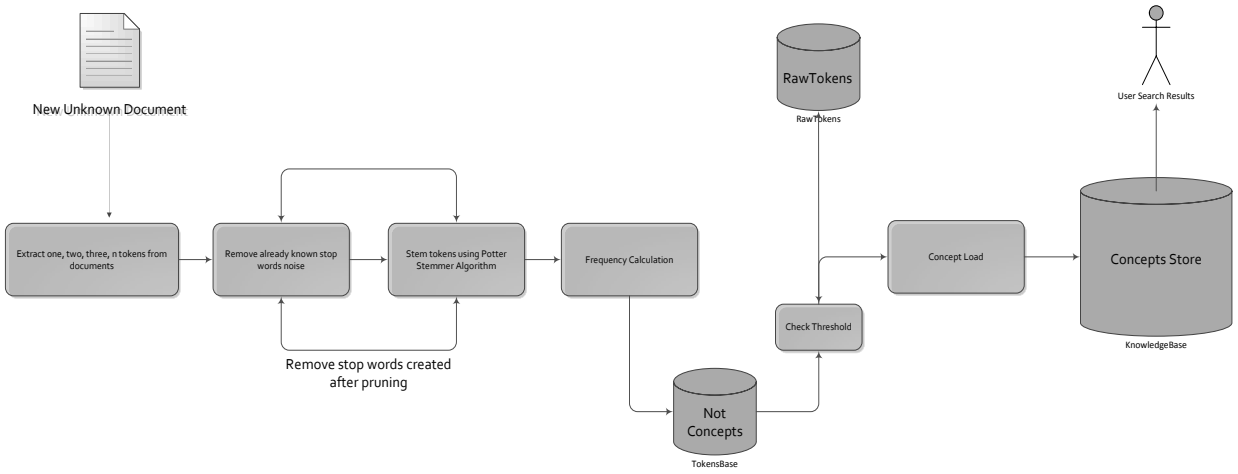

Figure-5-RawTokens Database

4.3.5 KnowledgeBase

Once the last pruning is performed on the concepts extracted from the documents we have successfully extracted the human concepts which can be called the KnowledgeBase. The knowledge base will be used to return the correct and related results for the user search queries. It will offer suggestions to users in the form of concepts related to the query from which they can chose their intended concept post which the knowledge base will return the documents containing relevant information. The following table shows top 150 concepts extracted from 600 IEEE papers on Big Data. We can clearly see that all the concepts captured are related to big data and say that program is

successfully able to capture the concepts. The point worth noting here is the top 150 concepts are 3 words which is due to the rule we applied during pruning the concepts of keeping the maximal simplex. That is the reason why concept containing only big data is pruned.

Top 150 concepts from "Big Data" KnowledgeBase

ID	Tokens	TokenCount	Frequency	DocFrequency	TokensOrigin
2228	big data analyt	3	409	38	Big Data. analytics
2308	big data process	3	349	48	big data processing
2482	big data applic	3	265	33	big data application
2271	big data comput	3	230	40	BIG DATA Computer
2413	big data manag	3	218	34	Big Data Management
2334	big data analysi	3	204	30	big data analysis
2270	big data cloud	3	171	30	BIG DATA CLOUD
2324	big data technolog	3	170	30	BIG DATA Technology,
2415	big data model	3	155	23	big data model
2322	big data storag	3	141	19	big data storage
2461	big data inform	3	130	31	Big Data information.
2737	big data servic	3	129	16	big data services
2244	big data platform	3	127	25	Big Data platforms
2234	big data challeng	3	124	34	Big Data. challenges
2262	big data busi	3	116	16	big data business
2418	big data new	3	115	25	big data new
2424	big data provid	3	113	20	big data provides
2302	big data need	3	108	22	big data need
2335	big data architectur	3	96	15	big data architectural
2246	big data research	3	94	25	Big Data research.
2303	big data network	3	92	13	big data network.
2404	big data gener	3	89	25	big data general
2529	big data framework	3	88	20	big data framework
2499	big data larg	3	84	21	big data large
2409	big data infrastructur	3	81	11	Big Data Infrastructure
2347	big data paper	3	78	27	big data paper,
2364	big data distribut	3	77	17	big data distributed
2723	big data traffic	3	77	6	big data. Traffic
2798	big data user	3	76	12	Big Data user

2530	big data hadoop	3	76	17	big data Hadoop
2321	big data set	3	76	27	big data set
2425	big data requir	3	75	20	big data requirements.
2512	big data secur	3	73	8	big data security.
2522	big data cluster	3	73	9	big data cluster,
2393	big data collect	3	72	18	Big data collection
2656	big data integr	3	72	13	big data integration
2450	big data differ	3	71	16	Big Data different
2372	big data http	3	70	15	Big Data http
2941	big data center	3	70	7	Big Data center
2764	big data sourc	3	69	14	big data. Sources
2320	big data scienc	3	69	16	BIG DATA Science,
2507	big data problem	3	67	18	big data problem
2229	big data analyz	3	66	17	Big Data analyze
2584	big data base	3	65	21	Big Data based
2500	big data mapreduc	3	65	14	big data MapReduce
2462	big data issu	3	63	12	Big Data Issues
2486	big data complex	3	63	13	big data complex
2479	big data volum	3	63	17	Big data volume,
2250	big data store	3	62	17	Big Data store
2408	big data includ	3	61	14	Big Data including
2316	big data result	3	60	18	big data result
2305	big data perform	3	60	22	big data performance
2582	big data type	3	58	14	Big Data types
2381	big data valu	3	57	10	big data value
2357	big data approach	3	56	18	Big Data Approach
2423	big data propos	3	55	18	big data proposes
2769	big data access	3	54	11	Big Data accessing
2558	big data relat	3	53	17	big data. related
2430	big data section	3	53	14	Big Data Section
2792	big data structur	3	52	10	Big Data. structured
2274	big data creat	3	52	12	big data creates
2488	big data cost	3	51	14	big data. cost
2282	big data environ	3	51	14	big data environment,
2414	big data method	3	50	13	BIG DATA methods
2496	big data increas	3	49	13	Big Data increasing
2699	big data develop	3	48	16	big data. Development
2291	big data ieee	3	47	20	big data IEEE
2607	big data high	3	47	12	Big Data, high

2635	big data govern	3	47	9	Big Data governance
2705	big data introduct	3	47	17	Big Data, Introduction
2373	big data industri	3	46	10	Big Data industry
2505	big data present	3	46	15	big data present
2580	big data tradit	3	45	12	BIG DATA traditional
2516	big data work	3	45	16	big data work,
2625	big data workload	3	45	7	Big Data workload
2345	big data organ	3	44	13	big data organizations
2732	big data make	3	44	13	big data makes
2258	big data algorithm	3	44	15	big data algorithm
2240	big data enterpris	3	42	6	Big Data enterprise
2474	big data techniqu	3	42	13	big data, techniques
2644	big data social	3	41	9	big data social
2845	big data onlin	3	40	6	Big Data online
2527	big data featur	3	40	10	Big Data feature
2778	big data heterogen	3	40	7	Big Data heterogeneity,
2608	big data import	3	39	12	Big Data, important
2431	big data semant	3	38	9	Big Data semantic
2540	big data time	3	37	13	Big data times.
2508	big data public	3	37	6	big data publication
2509	big data reduc	3	35	8	big data. reduced
2610	big data job	3	35	6	Big Data jobs
2503	big data parallel	3	34	8	big data parallel
2403	big data follow	3	34	11	Big Data following
3053	big data opportun	3	33	10	Big Data Opportunities
2352	big data refer	3	33	12	Big Data refers
2735	big data resourc	3	33	10	big data resource
2720	big data stream	3	33	8	Big Data, Stream
2476	big data varieti	3	33	10	Big data variety.
2528	big data file	3	32	8	big data File
2398	big data discuss	3	32	9	Big Data discussion.
2748	big data effect	3	32	10	big data effectively
2520	big data associ	3	32	5	Big data Associate
2652	big data exampl	3	31	10	big data example,
2538	big data softwar	3	31	9	big data software
2427	big data scientif	3	31	7	Big Data Scientific
2327	big data world	3	31	8	big data world
2799	big data vol	3	31	6	Big Data VOL.
2355	big data variou	3	30	12	big data various

2681	big data qualiti	3	30	7	Big Data quality,
3319	big data intellig	3	30	5	Big Data intelligence
2338	big data engin	3	30	7	Big Data Engineering
2671	big data control	3	29	9	Big data control
3034	big data studi	3	29	11	Big data study
2797	big data unstructur	3	29	7	Big Data unstructured
2561	big data wai	3	28	9	big data. ways
2911	big data queri	3	28	6	big data query
2611	big data knowledg	3	28	6	Big Data knowledge,
2886	big data oper	3	28	6	Big Data operation
2379	big data support	3	27	11	big data support,
2491	big data databas	3	27	10	big data database
3025	big data extract	3	27	4	BIG DATA extracting
2452	big data dynam	3	26	5	big data, dynamic
2879	big data futur	3	26	10	big data future
2808	big data initi	3	26	4	big data initiatives
2922	big data design	3	26	7	big data design
2968	big data consum	3	26	5	big data, consumed
2537	big data size	3	26	7	BIG DATA size
2432	big data solut	3	25	11	big data solutions
3101	big data real	3	25	7	big data real
2834	big data repres	3	25	6	big data representing
2822	big data context	3	25	5	big data context
2237	big data current	3	25	6	Big Data current
2292	big data improv	3	25	9	big data improve
2729	big data enabl	3	25	8	big data enables
2904	big data huge	3	24	6	big data huge
2377	big data scale	3	24	8	big data scaling,
2643	big data risk	3	24	4	Big Data Risk,
2433	big data specif	3	24	7	big data specific
2616	big data object	3	24	5	Big Data object
2590	big data pattern	3	24	5	Big Data patterns,
2618	big data predict	3	23	6	Big Data predictive
3005	big data ef	3	23	5	big data ef
2545	big data dataset	3	23	10	Big Data datasets
2268	big data china	3	23	4	BIG DATA China
2235	big data chang	3	23	6	Big Data changing.
2669	big data case	3	22	6	BIG DATA cases,
2272	big data confer	3	22	6	big data Conference.

3141	big data consid	3	22	6	big data Considering
2362	big data decis	3	22	6	big data decision
2673	big data econom	3	22	5	big data economics
2245	big data possibl	3	22	8	Big Data possibly

Table-09-Top 150 Concepts from KnowledgeBase

5 ANALYSIS AND RESULTS

5.1 Comparison with google search results

In order to verify the concepts obtained by the project we have fed some search result pages from Google search to the program on different topics, like “Apache Zookeeper”, “Holy Grail”, “Sound Investment”, “Pig Latin” etc. The following tables show the concepts extracted from those pages obtained by Google search. If we can see the same concepts in the KnowledgeBase which we searched for, we can conclude that the program successfully captures concepts and extracts knowledge from unknown documents.

All of the files which behaved as controlled input to the Concept Extractor are fed as obtained.

5.1.1 Concept extracted from “Apache Zookeeper” Google search data

Top 20 results obtained by Google search for “Apache Zookeeper” is used here as test data. These files are then fed to the program to extract human concepts. The expected result will include concepts related to Apache Zookeeper, Hadoop, Big data etc.

As expected, the top concepts are related to these topics. We see that “big data”, “Cluster”, “Open Source” also feature in top concepts and thus the proposed methodology was successfully able to relate Zookeeper to Big data which is a clear indication that program is able to extract human concepts.

	ID	Tokens	TokenCount	Frequency	DocFrequency	TokensOrigin
1	4255	servic zookeep	2	41	10	Services ZooKeeper
2	2885	applic zookeep	2	28	9	applications ZooKeeper
3	2237	coordin zookeep	2	28	11	coordination ZooKeeper
4	2894	big data	2	27	8	Big Data.
5	3098	cluster servic	2	25	6	cluster. services
6	3314	data zookeep	2	25	8	Data ZooKeeper
7	3920	open sourc	2	25	9	open source
8	3874	node cluster	2	24	4	node cluster,
9	4205	servic cluster	2	23	6	services cluster.
10	4200	server zookeep	2	23	10	servers. ZooKeeper
11	4210	servic distribut	2	22	10	services distributed
12	3030	client zookeep	2	22	8	client ZooKeeper
13	5959	new zookeep	2	21	7	new ZooKeeper
14	4505	watch znode	2	21	6	watch znode
15	4217	servic hadoop	2	20	4	services, Hadoop
16	3027	client server	2	20	6	client servers
17	5439	hadoop apach	2	20	6	Hadoop Apache
18	2363	hadoop zookeep	2	20	7	Hadoop Zookeeper
19	5195	engin apach	2	19	5	engineer Apache
20	4715	blog zookeep	2	18	8	Blog ZooKeeper
21	7880	implement zookeep	2	18	8	Implementing ZooKeeper
22	4207	servic configur	2	17	8	services configuration
23	4800	client znode	2	17	5	client znode
24	4431	synchron zookeep	2	17	6	synchronization ZooKeeper
25	9248	work zookeep	2	16	7	works ZooKeeper

Table-10-Concepts for Apache Zookeeper

5.1.2 Concept extracted from “Holy Grail” Google search data

Top 30 results obtained by Google search for “Holy Grail” are used as test data for the program. We have chosen these keywords as search query as Google returned many pages related to a song “Holy Grail” which is not the correct concept contained by words. The expected result will include concepts related to the Holy Grail dish and concepts related to mythology.

As expected the top concepts are related to the mythological Holy Grail. As we can see that “knight Arthur is also in top concepts. Still we can see some of the results related to video which is because of the pages

	ID	Tokens	TokenCount	Frequency	DocFrequency	TokensOrigin
1	102385	monti python holi grail	4	150	7	Monty Python Holy Grail,
2	16180	knight arthur	2	82	6	Knights Arthur
3	16923	sir launcelot	2	67	4	Sir Launcelot
4	16893	sir galahad	2	54	4	Sir Galahad
5	43802	black knight	2	52	4	BLACK KNIGHT
6	83788	new holi grail	3	48	13	NEWS Holy Grail,
7	83823	photo holi grail	3	48	9	PHOTOS Holy Grail,
8	102424	python monti holi grail	4	45	5	Python Monty Holy Grail.
9	12199	quest grail	2	44	10	quest Grail
10	83881	video holi grail	3	43	8	VIDEOS Holy Grail,
11	7463	new grail	2	39	15	NEWS Grail,
12	10477	arthur king	2	39	4	ARTHUR King
13	34391	teri jone	2	35	4	Terry Jones
14	10472	arthur grail	2	34	9	Arthur Grail
15	15230	arthur sir	2	32	5	Arthur Sir
16	16227	knight sir	2	32	4	Knights Sir
17	25453	movi tv	2	30	5	Movies, TV
18	15945	galahad sir	2	29	4	Galahad Sir
19	86719	python monti grail	3	29	5	Python Monty Grail.
20	101779	video photo holi grail	4	27	5	VIDEOS PHOTOS Hol...
21	16917	sir knight	2	27	5	Sir Knight,
22	90552	world holi grail	3	27	4	WORLD Holy Grail
23	12826	year grail	2	26	10	years Grail
24	85698	quest holi grail	3	26	7	quest Holy Grail
25	16566	old man	2	26	4	Old Man

Table-11- Concepts for Holy Grail

5.1.3 Concept extracted from “Pig Latin” Google search data

Top 44 results obtained by Google search for “pig latin” are used as test data. Those files were then fed to the program. The expected result will include concepts related to Apache Pig, Hadoop or Big data and the language pig latin etc. We chose this phrase as most of the results returned from google are related to language pig latin and not to Apache Pig. This is the reason we increased the no. of result files fed to program

As expected the top concepts are related to these topics. As we can see that “Hadoop pig latin”, “pig script” are also in top concepts we can clearly see that proposed methodology successfully able to relate pig latin to Big data.

	ID	Tokens	TokenCount	Frequency	DocFrequency	TokensOrigin
1	64496	translat pig latin	3	124	17	Translations Pig Latin
2	65994	languag pig latin	3	98	16	language. Pig Latin
3	96372	translat pig latin word	4	92	6	translation Pig Latin. Words,
4	37765	exampl data	2	90	4	Example data
5	64673	word pig latin	3	79	18	Word Pig latin
6	6740	translat latin	2	69	17	Translations Latin
7	64349	english pig latin	3	64	15	English Pig Latin
8	7543	word latin	2	61	18	Word latin
9	8628	data platform	2	59	5	Data Platform
10	66456	translat latin pig	3	58	8	translates Latin Pig
11	9358	oper data	2	55	6	operations, Data
12	65901	hadoop pig latin	3	54	4	HADOOP PIG Latin
13	38290	group kei	2	53	4	GROUP key
14	37563	data exampl	2	52	4	Data Example
15	71413	speak pig latin	3	51	9	speaks Pig Latin
16	69890	translat latin word	3	49	6	translation Latin. Words,
17	69896	translat pig word	3	49	6	translation Pig Words,
18	7794	word exampl	2	49	13	word examples
19	78303	foreach gener group	3	49	4	FOREACH GENERATE G...
20	38246	gener group	2	49	4	GENERATE GROUP
21	38078	foreach group	2	48	4	FOREACH GROUP
22	9718	script pig	2	48	4	scripting Pig
23	7789	word ay	2	44	15	words ay
24	65892	hadoop pig data	3	43	4	HADOOP PIG data
25	70630	load store data	3	42	5	LOAD stored data,

Table-12- Concepts for Pig Latin

5.1.4 Concept extracted from "Sound Investment" Google search data

Top 52 results obtained by Google search for "sound investment" are used here as test data. Those files then fed to the program. The expected result will include concepts related to financial domain data and not to music industry. Most of the results returned by google were related to music and investment separately and not related to financial domain.

As expected the top concepts are related to these topics. As we can see that "stock market", "financial plan", "stock bond" are also in top concepts we can clearly see that proposed methodology successfully able to relate sound investment to finance.

	ID	Tokens	TokenCount	Frequency	DocFrequency	TokensOrigin
1	84957	home sound invest	3	57	17	HOME Sound Investment
2	5231	servic invest	2	38	16	Services Investment
3	86082	servic sound invest	3	35	12	Services Sound Investment
4	16465	stock market	2	24	6	Stock Market
5	4819	contact invest	2	23	14	Contact Investment
6	9517	dj servic	2	23	6	dj, services.
7	85211	event sound invest	3	22	6	events Sound Investment
8	8511	financi plan	2	22	7	FINANCIAL PLANNING
9	14880	manag invest	2	22	11	Manager investments
10	86875	music sound invest	3	22	7	musical Sound Investment,
11	10937	music invest	2	21	7	musical Investment,
12	85171	contact sound invest	3	21	12	Contact Sound Investment
13	27252	stock bond	2	21	5	stocks, bond
14	6535	wed music	2	20	5	wedding music
15	10476	wed servic	2	20	6	wedding services.
16	8851	plan invest	2	20	13	PLANNING Investment
17	27781	asset alloc	2	20	5	Asset Allocation
18	8502	financi invest	2	20	8	FINANCIAL Investment
19	9827	make invest	2	20	9	make Investment
20	14683	fund invest	2	19	4	funding, investments
21	5710	event invest	2	19	7	events Investment
22	4770	audio sound	2	19	5	audio, sound
23	5134	privaci polici	2	19	18	Privacy Policy
24	8975	retir invest	2	19	6	Retirement Investment
25	23910	year invest	2	19	9	years, invest

Table-13- Concepts for Sound Investment

5.2 Safe range for TF-IDF

In order to find out which threshold for term frequency or document frequency will be best capturing the solid concepts or pure knowledge we studied the relation between TF-DF by keeping constant data size in one experiment and repeating the same experiment to find out how data size can affect their relation.

Below are the result graphs explaining the relation. I must mention here that safe range for TF-DF is between TF ranging from 1.2% to 1.8% and for DF is from 0-95% to 10-95%. This value doesn't change much by change in data size and almost all the concepts are captured irrespective of data size.

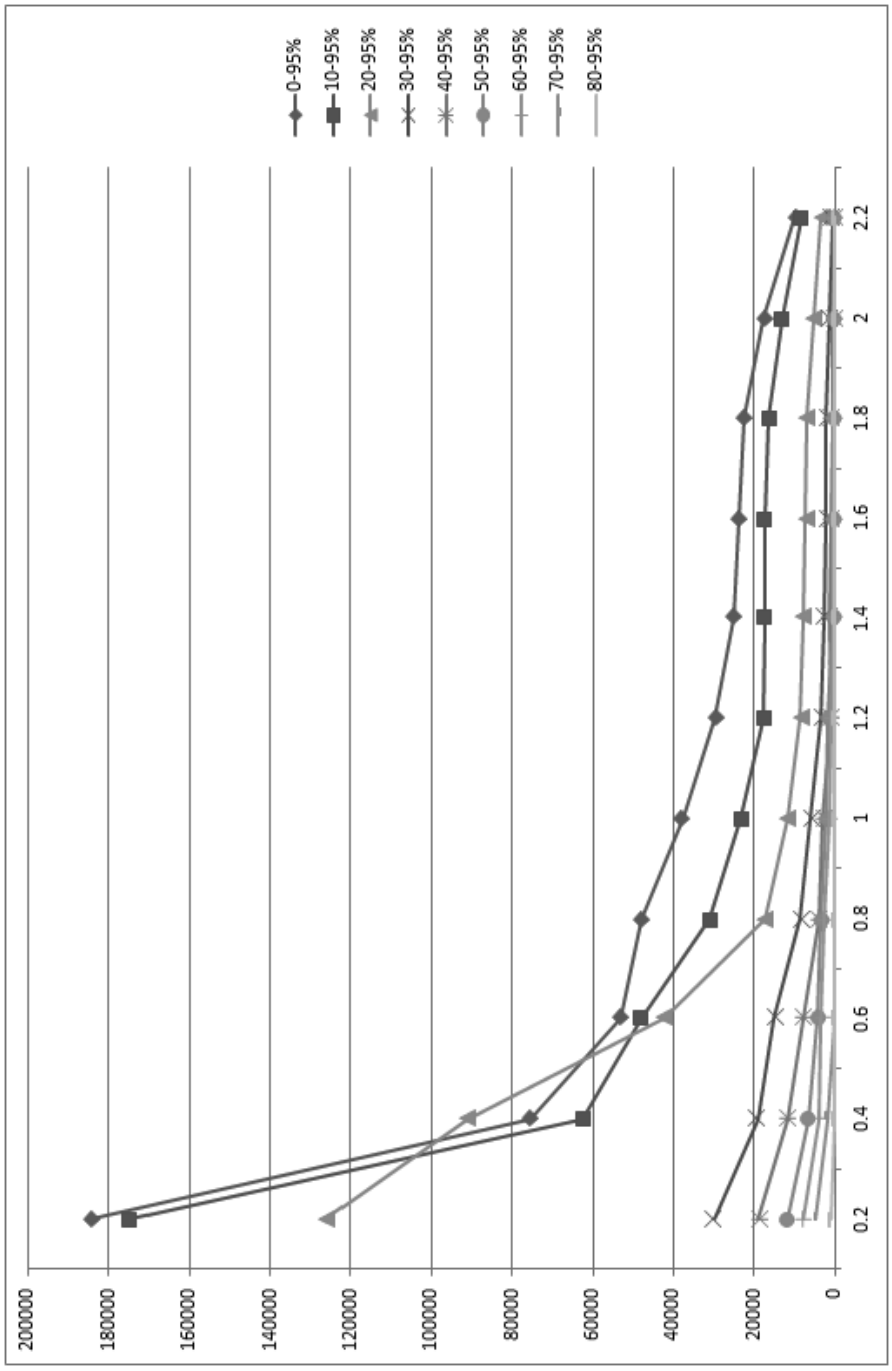

Figure-6-Graph representing relation between TF-IDF with DF

X axis – Token frequency | Y axis – No. of concepts extracted * Constant

5.3 Query retrieval and search results

The search queries retrieval is also a crucial part of every search engine. We propose a new approach in this section. Almost all of the search engine present today works on guessing the intended meaning of the search query like if I searched for “sound investment” google assumed that I am interested in pages related to music but my intension was to search for pages related to finance which talk about investment which can give me good returns.

Our search engine rather than giving the pages containing the concepts first provides a list of related concepts from the KnowledgeBase which relates to the search query so that user can pick up his intentional concepts. In above query the search engine developed will the user a list of concepts related to both finance and music from which user will choose one.

Once the user provides his intentional concepts from list of related concepts the engine will provide the list of pages containing the concept user search along with main concepts from those documents. By providing the list of concepts contained by each page the possibility of user looking into wrong document decreases to a large extent as he will know the central idea of the document returned by just looking at the concept list of that document. So our search engine works on “asking the correct questions to user rather than guessing”. Below sample will provide a better understanding of the approach.

```
Please Enter your search query..
Sound

The knowledgebase contains following concepts related to
your query..

home sound invest,  84957,HOME Sound Investment
servic sound invest,  86082,Services Sound Investment
event sound invest,  85211,events Sound Investment
music sound invest,  86875,musical Sound Investment,
contact sound invest,  85171,Contact Sound Investment
```

audio sound, 4770,audio, sound
home invest sound, 85694,Home Investment Sound
product sound invest, 88573,Products Sound Investment
wed sound invest, 85478,wedding Sound Investment
creat sound invest, 87122,create Sound Investment
home servic sound invest, 106460,Home Services Sound Investment
financi sound invest, 86364,FINANCIAL Sound Investment
compani sound, 14366,companies. Sound
plan sound invest, 86483,PLANNING Sound Investment
product servic sound invest, 107757,production services Sound Investment

Please enter the intended concept id.. (As I am interested in finance)

86364
C:\Users\Pradeep\Documents\CS298\Reports\TestData\SoundInvestment\new 31.txt
home sound invest, 57
stock market, 24
financi plan, 22
financi invest, 20
make invest, 20
plan invest, 20
privaci polici, 19
retir invest, 19
home invest sound, 16
portfolio invest, 16
creat invest, 15
financi sound invest, 13
new invest, 13
right reserv, 13
help invest, 12
C:\Users\Pradeep\Documents\CS298\Reports\TestData\SoundInvestment\new 26.txt
servic invest, 38
contact invest, 23
financi plan, 22
manag invest, 22
financi invest, 20
asset alloc, 20
plan invest, 20
retir invest, 19
market invest, 14
new york, 14
advisor invest, 13
financi sound invest, 13
team invest, 13

```
client invest,  12
need invest,  12
C:\Users\Pradeep\Documents\CS298\Reports\TestData\SoundInve
stment\new  19.txt
home sound invest,  57
servic invest,  38
manag invest,  22
financi invest,  20
make invest,  20
fund invest,  19
privaci polici,  19
home invest sound,  16
compani invest,  16
portfolio invest,  16
commun invest,  13
financi sound invest,  13
advisor invest,  13
compani sound,  12
need invest,  12
C:\Users\Pradeep\Documents\CS298\Reports\TestData\SoundInve
stment\new  13.txt
contact invest,  23
financi plan,  22
contact sound invest,  21
financi invest,  20
plan invest,  20
retir invest,  19
start invest,  17
financi sound invest,  13
team invest,  13
plan sound invest,  12
team sound invest,  12
save retir,  11
secur invest,  11
april invest,  10
plan financi,  10
```

6 FUTURE SCOPE

6.1 Applications

The built KnowledgeBase can be utilized in creating many useful semantically aware applications. The following applications are proposed which can be created with minor modifications in the core algorithm:

6.1.1 Automated Indexing of books

We can use the core engine of our work to create a system which automatically generates index for books. There are two methods currently available to create index for books:

a) The author has to create the index which can be automated by the project as it is successfully able to extract the concepts contained by the book.
b) A user reads and understands the book to create the index which can be very tedious human effort.

Our project can help in saving this manual effort by providing a technique for automated indexing of books and ease the efforts put in by authors and publishers.

6.1.2 Preventing misuse of sale categories in e-commerce websites

The program created in this project can also be used to test if specific listings of products on e-commerce websites actually belong to the product category they are launched in by third-party sellers. Major e-commerce websites like Amazon, eBay, Alibaba charge a commission to the seller as per the category of the product. The commission may vary as per the category of the product listed. It may be less for some house hold product and more for electronic products.

To trick the commission charged what sellers do is that they list all the products in the category for which the commission is least so that they can save more. E.g. they will list electronics in house hold items. One thing a seller never lies about is the description of the product.

As the listings are not checked for correct category and, thus, generally not caught. But following this practice is bad for both end user as they will not find the item they are looking for in the specific category and owners will earn less which will hamper their profits.

Our program can extract the concepts from the description which are always true and find out the category for the product as description will contain information about the product which can be mapped to its category. Like a term “phone” should be listed in electronics and not in household.

This application will help in better online shopping experience and the owners will also earn the profits.

6.1.3 Document clustering by their knowledge

The application can be used in creating document clusters as per their knowledge, which our program can extract. Clustering the documents as per their contents will help in better search results. As a document in a cluster will be related to all other documents in that cluster so the user might be interested in other documents as well. We can list the related documents for him from the cluster and his query will be served better.

6.2 Migration to Hadoop

The central idea is to prove that the algorithm explained by the project work is relevant and utilizable in creation of semantic search engine. After the proof of concept and taking into account promising results, a platform has been created for future scope by building the foundation and successfully migrated the program to Hadoop by keeping the underlined vision that we would like to extend the project to create a better semantically aware concept based search

engine. We have seen that as per the results we can surely say that the project is successfully able to capture the concepts and it is correctly extracting knowledge from unknown documents. This project is to demonstrate that the same algorithm can be used for creating a concept based semantic search engine. Let's first revise the terms used here:

Concept means human knowledge represented in words in the documents as well as html pages on the web. If we are extracting that knowledge from the documents we can say that our system will know that knowledge and can behave as a brain, if you will, which has all the answers or links to the documents containing the answers for any query (Central assumption: the system has processed all of the available documents/html pages and stored all the concepts achieving which is discussed later in this section).

The second term is semantic which means that words which are written together (side by side to each other) or in a nearby paragraph are related to each other in the context which they are trying to represent. I will use an example here provided by Dr. Pearce during one of the discussions on project and it clearly explains the meaning. Let the sentence be "Pradeep Roy is skiing and swimming and he is nature lover." The words 'Pradeep' and 'nature lover' do not appear side by side but are still getting used to express the fact that Pradeep loves nature. So these words are semantically related to each other representing a concept. Our system also captures this scenario and successfully finds the semantics in the words. So the search engine made on top of the current setup will be semantically aware.

The third word is Search Engine which explains that the system developed and demonstrated by the project will be used in making a search engine to provide the better search results to user then most of the available search engines present today.

6.2.1 Building blocks of search engine

The basic and the most important building block of any successful search engine is the amount of data it knows and can search from. The quality of results is directly proportional to the data size, more the data more accurate the results are if processed correctly. In order to make our search engine we would require to process all of the data available on W3. We have already found out the limitation of the prototype created in the project that it cannot handle large amount of data. The only solution to the problem is to migrate to Hadoop where we can analyze Big Data without hitting the memory constraints. Hadoop is designed specifically to overcome these constraints. As a trend it is better to migrate Hadoop in the first place rather than after creating a stable product. The only requirement to migrate to Hadoop other than having the infrastructure is paralleling the task. If a task can be paralleled, the algorithm can be successfully paralleled as well. We can parallelly analyze the documents divided into many small documents and extract the potential concepts out of them and then finally we can merge all the potential tokens to find out actual tokens.

After a brief introduction to Hadoop we will see the pseudo code explaining the flow in terms of map reduce. Knowing Hadoop architecture will help us better understand the problems which we may face in future and their solutions.

6.1.2 Hadoop Architecture and MapReduce

Hadoop works on Hadoop Distributed File System (HDFS) which is designed and implemented to run on cheap commodity hardware. It has transformed by keeping some part with typical Distributed File System and in that way it is very much similar to it. Even then the differences are significant and prominent. HDFS is designed to be fault tolerant and is designed to run on low

cost hardware. HDFS is very much suitable to the applications which are designed to handle very large datasets like search engine. Google and Yahoo run on HDFS and many other social networking companies have large HDFS clusters of approximately 1000 nodes and process petabytes of data. HDFS has high throughput access for the application data.

HDFS provides the platform to run Hadoop MapReduce jobs. Hadoop MapReduce is a java framework which is built to write applications which process large amounts of data. Applications can process that large data in parallel on large clusters which are reliable and fault tolerant. A MapReduce job is basically divides the data into independent chunks which are then processed by map tasks in parallel. The output of the map tasks is the input to reduce tasks. The MapReduce framework is takes care of scheduling, monitoring and re-execution of failed tasks. Typically the data-nodes which store the data are also the compute nodes. MapReduce and HDFS run on the same nodes. This configuration enables the MapReduce to schedule the tasks on the nodes where data available in advance. This saves a lot of bandwidth for data transmission. MapReduce has two components:

I. Single master JobTracker,
II. One TaskTracker per node in the cluster

The JobTracker works for scheduling the jobs and deploy the tasks to slave nodes, monitoring, and re-executing failed tasks. The slave nodes perform the tasks. The applications developed specify input and output locations from where to get data and to where to store the results. All the applications have map and reduce functions via implementations of corresponding interfaces or abstract class. Hadoop job client is responsible for submitting the jobs and configuration to JobTracker. Hadoop MapReduce framework is written in Java but it is not mandatory to write the applications using it in Java. Our

application is written in Java.

6.1.3 Pseudo Code and Class explaination

For any project to run on Hadoop, it is mandatory that the task can be split into many parallel tasks so that Hadoop can utilize the underlying concept of map and reduce.

The pseudo code remains the same and the outline would be:

```
Crawl the web space and collect the documents
For each task do {
Documents <= parse to make usable format
Map: For documents extract tokens
          Calculate TF, DF for tokens
Reduce: Merge TF-DF for same tokens
          Calculate TF-IDF
Perform pruning
}
```

Map Classes

Mapper Class	Functionality
TokenCountInDoc	Many mappers reads a document in parrallel Calculates the frequency of each token in the document
TokenCountInDocs	Calculates token frequency of each token for many documents using TokenCountInDoc Keep a record of document frequency for each token

TokenTFIDF	Calculates TF-IDF for all the tokens

Table-14- Map Classes

Reduce Classes

Reducer Class	Functionality
TokenCountInDocReducer	Takes input from corresponding mappers and merge the results
TokenCountInDocsReducer	Takes input from corresponding mappers and merge the results
TokenTFIDFReducer	Takes input from corresponding mappers and merge the results

Table-15- Reduce Classes

6.1.4 Initial development

As migration to Hadoop is a future scope, we have successfully developed part of the project where TF-IDF from all of the documents can be calculated for all multiple keyword tokens sets.

Pruning of keywords which don't qualify for being designated a concept is an area that can be picked up along with implementing the above mappers and reducers.

6.1.4 Infrastructure suggested

The following recommendations are made based on my research:

DataNode or Slaves:

- 1-4 TB HDD configuration 2 quad-core processors having at least 2-2.5GHz 4-8 GB RAM
- Bonded Gigabit Ethernet card.

NameNode or Master Node:

- 1TB hard disks 2 quad core processors, at least 2-2.5GHz 16-32 GB of RAM
- Bonded Gigabit Ethernet card

7 REFERENCES

1. Tsau Young (T. Y.) Lin, Albert Sutojo and Jean-David Hsu; Concept Analysis andWeb Clustering using Combinatorial Topology (2006)
2. Tsau Young (T. Y.) Lin and Jean-David Hsu; Knowledge Based Search Engine Granular Computing on theWeb
3. Apriori algorithm; http://www.cs.sunysb.edu/~cse634
4. Introduction to Information Retrieval - By Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze ; Website: http://informationretrieval.org/ ; Cambridge University Press
5. Google Search, http://www.google.com.
6. Porter Stemmer Algorithm, http://tartarus.org/martin/PorterStemmer/, 2006.
7. https://pdfbox.apache.org/index.html
8. http://hadoop.apache.org/docs/r0.18.0/hdfs_design.pdf
9. http://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/
10. https://blog.cloudera.com/blog/2013/08/how-to-select-the-right-hardware-for-your-new-hadoop-cluster/
11. http://obitko.com/tutorials/ontologies-semantic-web/semantic-webarchitecture.html