San Jose State University
SJSU ScholarWorks

Master's Projects | Master's Theses and Graduate Research

1-1-2007

# Concept Analysis in Web Documents

Rajesh Singh
*San Jose State University*

Follow this and additional works at: http://scholarworks.sjsu.edu/etd_projects

Recommended Citation
Singh, Rajesh, "Concept Analysis in Web Documents" (2007). *Master's Projects.* Paper 38.

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

# CS298 Report

# Concept Analysis in Web Documents

Doc. Ref. : cs298report
Version : 1.0
Status : Baseline
Created by : Rajesh Singh
Date : Oct/12/2007
Approved by : Not currently approved

**Version History**

| Revision | Release Date | Updated by | Remarks/Comments | Status |
|---|---|---|---|---|
| 0.1 | Oct/12/2007 | Rajesh Singh | First draft | Draft |
| 0.3 | | | | |
| 0.4 | | | | |
| 0.5 | | | | |
| 1.0 | | | | |
| 1.01 | | | | |
| 1.02 | | | | |

**Distribution**

| Name | Organisation/Location | Action/Information |
|---|---|---|
| Dr Cay Horstmann | Dept of Computer Science | for Approval |

**Abstract**

A Keyword within a text/web document represents some human thought. The interaction of keywords leads to narrowing of scope of human thought by forming a more precise semantic entity called concepts. Analyzing a set of document not only requires analysis of the keywords within those documents but also their interactions within a document. In this new approach a set of documents can be analyzed where by the interactions of its keywords is also considered in finding the important concepts. These concepts can be used to cluster them into smaller subsets such that documents in each cluster will be semantically similar.

**1 Introduction** ..... **5**
*1.1 Background* ..... *5*
*1.2 Scope of the Project* ..... *5*
*1.3 This Document* ..... *5*
**2 Concept Analysis** ..... **6**
*2.1 Background – To help search engines give more relevant results* ..... *6*
*2.2 Ideas Behind Finding Concepts* ..... *8*
**3 How Concept Analysis Works** ..... **10**
*3.1 Mathematical Foundation Simplicial Complex* ..... *10*
*3.2 Property of Simplexes in a Simplicial Complex* ..... *10*
*3.3 Simplicial Geometry of Keywords* ..... *10*
**4 Implementing the Concept Analysis Algorithms** ..... **12**
*4.1 Concept Analyzer* ..... *12*
*4.2 Design of Project* ..... *12*
*4.3 Project Flow Charts.* ..... *15*
*4.4 Graph Theory Approach* ..... *19*
*4.5 Geometrical Approach* ..... *21*
**5 Test Results** ..... **22**
*5.1 Effect of Simplex Size* ..... *22*
*5.2 Changing Association Rules for Simplex Generation* ..... *26*
*5.3 Comparison of Graph Theory and Geometrical Approach* ..... *30*
*5.4 Comparison with Yahoo Desktop Search* ..... *34*
**6 Conclusion** ..... **39**
**7 List of Literature References** ..... **40**

# 1 Introduction

## 1.1 Background

As part of my Masters Writing Project (CS297/CS298) at San Jose State University, I have decided to work in the field of text analysis. The topic of my work is "Concept Analysis in web documents" and my guide is Professor T. Y. Lin Department of Computer Science, SJSU.

## 1.2 Scope of the Project

The scope of this project involves: -

- Understanding the correlation between a set of documents with large item-set properties.
- Abstraction of keywords in a set of documents to a collection of simplexes, also known as simplicial complex.
- Reducing the problem of keyword analysis in a set of documents to a problem of simplicial complex analysis and then further reducing the problem of simplicial complex (a structure in n-dimensional Euclidean space) to a linear problem of graph.
- Implement concept analysis algorithm for the graph theory approach.
- Implement concept analysis by geometrical method also and do its comparison with the graph approach.
- Discuss the out put generated with a standard set of data taken from UCI KDD website.

## 1.3 This Document

The CS298 report is a technical deliverable the purpose of which is:

- To describe the project work done for CS297/CS298.
- To specify the design, implementation and algorithms used for implementing concept analysis.
- To explain how this technique is different than other text analysis techniques and what is the effect on the output as a result of this difference.

# 2 Concept Analysis

## 2.1 Background – To help search engines give more relevant results

Before improving the results given by a search engine it is important to know what are the plausible ways a search engine may work. The actual working and implementation of prominent search engines is proprietary and not open for public, yet the basics of what may go inside of a search engine is well known [1].

Let us suppose we have to pick out all the articles from a stack of articles or written literature that are related to *ground zero.* The probable way to do that would be to scan through each article word by word looking for the exact phrase "ground zero". One approach could be to just skim through the headlines of articles that are related to terrorism or war, and then reading them to find a connection.

In another instance suppose I have been handed a stack of chemical journals and asked to find journals that have to do with *explosive Compounds,* if I am not an expert in the field of chemistry then I will have to go through each article line by line looking for the phrase "explosive compounds" in a sea of jargon and chemical equations.

The two searches would yield quite different results. In the first example the search may end early with few misses of articles with the phrase "ground zero" if it will appear in an unlikely article say about presidential nominee Rudy Juliani. On the other hand the search will find related articles that may talk about Global Terrorism or Arab terrorists which could be very well related to the phrase "ground zero" even if it didn't contain that phrase. In the second example of chemistry journals the search will find each instance of the exact match with phrase "explosive compounds" but I may miss articles about compounds like Tri-Nitro-Toluene (TNT), Tri-Nitro-glycerol, picric acid etc. which are also very explosive compounds unless I have significant knowledge about Chemical compounds.

In the above example both the searches represent two totally different ways of searching a document set. The first one can be called a conceptual search where the heading or the title of the document may be related to the contents of the article in some understood way, whereas the second approach is purely mechanical based on the exhaustive search of the phrase in a much larger document set.

We see that both the approaches mentioned above have some serious limitations and the question is "What else can be done to mitigate the above mentioned issues?" Let us look at Taxonomy as a technique that may be applied to help searches. Something like a librarian does by assigning keywords to works or articles can be done on a large set of document. Rather than indexing the full text of each article the collection is assigned keywords in some sort of a fixed hierarchical structure and doing a comprehensive classification of sorts. This will definitely be helpful in improving the efficiency of the search engine because the user can use concepts rather than just individual keywords or phrases in their search, but this technique too has some serious limitations. Let us consider two sets of documents such that one set has articles about first half of Europe describing food habits of people based on geography and another set of articles about the second half of Europe describing food habits of people based on race. How can these be

merged? Either, I would have to choose any taxonomy from the two or come up with a totally new one. In both the scenarios I will be re-indexing a lot of data. One great solution for this problem of merging different taxonomies is to not merge them at all. Instead, have each document assigned multiple keywords or categories resulting in multiple ontology. Now this approach is also not without its share of problems. First of all having multiple taxonomies will raise system resource issues. Secondly, it is almost impossible to have an expert archivist review and classify every document in a collection moreover there is a very good chance that the taxonomy and keyword vocabulary may continue to grow.

In the above paragraphs, regarding possible ways a search can be carried out, we can see that all the techniques mention so far doesn't do a good job. Some techniques only do text matching whereas others will do conceptual match provided someone (most likely a human) has already done some classification of documents based on concepts. We know that classification or tagging of the documents to some important keywords is not a trivial exercise given the enormity of data. We also know that computers are very efficient in doing repetitive tasks but the problem is they don't have brain or power to perceive things. How can this power of computers, which lies in doing repetitive work, be complimented with some form of intelligence or perception based approach. There is a totally separate branch of computer science that deals with the aspects of computers as human brains under *Artificial Intelligence.* I will not discuss artificial intelligence because it is beyond the scope of my work. However, there is a way in which computers can be made to pretend that it can perceive concepts. The technique is commonly known as Latent Semantic Indexing (LSI) [1]. The LSI approach is known to work decently well with textual data and the results are quite ok. I will just point out the basic idea behind LSI, how it pretends to perceive a concept. In LSI, instead of taking the each document one by one and building indexes on its keywords, the whole document collection is taken as a pool to find what keywords appear together in substantially large number of documents within the given document set. This approach is based on the assumption that if certain keywords are present together in many documents then it means some perception or some commonality of subject. It has been found that this assumption decently aligns with human interpretation of a document classification for most types of textual data. For example if the keywords *Saddam, Hussein, gulf, war*, and *bomb* appear in many documents in a document collection then there is a very good chance that above mentioned keywords help classify a subset of the whole document set. A human can here perceive that the subset of documents classified by the above mentioned keywords have something to do with Middle-East crisis. On the contrary the computer cannot perceive what these keywords, when present within all the documents of a subset, would mean. Therefore, we see that computer failed at understanding the meaning or perception of these common keywords within a subset of documents but it surely was able to find these keywords. In LSI this power of computer to find a certain keyword combinations that are present in each document of a document subset is coupled with the assumption that such keyword combinations or keyword patterns have some semantic. Indexes are built on such keyword patterns which are used to answer search queries. This is not a foolproof way to build indexes for conceptual search but it works well for certain types of document collections. There are lots of other assumptions and various methodologies for LSI implementation, for which the information is available on the Internet. A further

discussion of LSI is beyond the scope of my work. One interesting observation about this LSI technique is that it helps to consider the whole document set together for analysis rather than considering one document at a time.

## 2.2 Ideas Behind Finding Concepts

In my work the notion of word *concept* corresponds to a set of keyword combinations or phrases that can classify the given set of document into some meaningful group. I will be using the basic tenet of LSI which considers taking the whole document set together instead of going over each document individually. I will not be using the LSI algorithm in its entirety; instead I will combine some ideas of LSI with the properties of a mathematical entity called a simplex along with the algorithms of graph theory. Since *concepts* have the characteristics of being able to be perceived by humans so it will be a good idea to output results (concepts) in human understandable form. To make this possible it is important to remove all kinds of formatting present in the documents of the collection under study. Web documents normally have html/xml tags along with some additional header information. In my approach all the documents will be stripped of their metadata, including html/xml tags and other information like title or keywords. After the above cleansing operation the resultant output will be a collection of bare bone text data files. This approach can be easily fitted into the larger scheme of things apparently the search engine. In the real world it can be assumed that the *crawler* will get the web documents from the internet on to the disk in the form of a document set. This document set will also have all the metadata which can be stripped off by performing the cleansing operation as mentioned above to give a collection of text data documents. The implementation and use of the crawler is beyond the scope of my project, and I will be using text and web data available at the UCI Kdd website. At UCI Kdd both forms of data are available (text data files and html formatted textual data) and I will run my experiments with both types of data to compliment my claim that concept analysis can improve the quality of results returned by a search engine.

As described under section 2.1 above for a collection of documents, that has text data, keyword combinations or phrases that span across multiple documents will be found out. These keyword combinations will then be analyzed together to see if they are associated with each other or not. The ones that will be associated can be grouped together and each such unique group will be a *concept*. To find the associations between the keyword combinations, obtained as above, a correlation is developed with a mathematical entity called simplex [2] and then using the property of large item sets (from data mining) for finding associations [6].

The commonality between my approach for finding keyword combinations and LSI approach is that in both the approaches we consider keyword combinations or patterns that spans across multiple documents. The major difference between my approach and that of LSI lies in the fact that LSI is based on Single Value Decomposition (SVD) [1][5] or 0-simplex whereas my approach is not based on Single Value Decomposition as we will consider simplexes of higher order too (0-simplex, 1-simplex,….). To understand the difference more clearly lets consider an example: consider a case such that in a document collection the keywords “Wall” and “Street” together span across substantially large number of documents. According to the LSI

approach the set of documents that will contain both these keywords may be treated to be in same semantic space, the semantic space determination subject to other calculations of local and global weight. In LSI approach a list of documents is maintained for each keyword and hence for a keyword combination (like “Wall” and “Street”) or phrase an intersection of sorts is taken which is abstracted as its Latent Semantic Space. In my approach of simplexes the ordering of keywords is very important, also inherent in my hypothesis; hence “Wall Street” will not be given the same treatment as “Street Wall”. In my CS297 report (an account of my literature research and findings) and CS298 proposal (hypothesis and description of my project writing) I have mentioned that “A document can be seen as a collection of keywords where each keyword represents some human thought [2]. The interaction of these keywords leads to some concept formation, in other words capture the semantics of that document”. Since we are talking about interaction of keywords within a document their ordering should be taken into account. We have seen in the above example how keywords ‘Wall” and “Street” have totally different semantics or meaning by changing their orders. The phrase “Wall Street” represents a financial notion like New York stock exchange whereas the phrase “Street Wall” represents something totally different. Therefore in my approach the keyword combinations “Wall Street” and “Street Wall” will fall in different semantic spaces.

# 3 How Concept Analysis Works

## 3.1 Mathematical Foundation Simplicial Complex

The definition (verbatim) of Simplicial Complex as given by wikipedia (http://en.wikipedia.org/wiki/Simplicial_complex) is "In mathematics, a simplicial complex is a topological space of a particular kind, constructed by "gluing together" points, line segments, triangles, and their n-dimensional counterparts". For example a simplex {A,B,C,D} is a Set such that it contains all its subsets i.e. {A,B,C}, {A,C,D}, {B,C,D}, {A,B}, {A,C}, {A,D}, {B,C}, {B,D}, {C,D}, {A}, {B}, {C}, {φ}.

Each document can be seen as a collection of keywords. When considering the whole document collection, keyword combinations that span multiple documents can be obtained based on high Term Frequency Inverse Document Frequency (TFIDF) [2]. These keyword combinations will be frequent item sets of length q. In the above mentioned example {A,B,C,D} is an item set of length q = 4. All such keyword combinations of the document collection will form an abstract simplicial complex [2]. In a simplicial complex the length of all the item sets will not be the same i.e. {q = 1,2,3, …} the item set length can be any positive integer. One variation that I have applied in my approach is that I am preserving the ordering of keywords in my analysis whereas in a simplex the order is unimportant because a simplex is a set. So my variation can be seen as a modified simplex where all the other properties of simplex still holds true.

## 3.2 Property of Simplexes in a Simplicial Complex

A simplicial complex is topologically equivalent to a triangulation (Linear simplicial complex) of a polyhedron in Euclidean space [2] and this polyhedron is topologically equivalent to the notion of human thoughts that are formed by the keyword combination in the documents. This notion of human thought can be seen as the Latent Semantic space (LSS) of the collection. So we can see how an n-dimensional structure of simplexes in Euclidean space is equivalent to the semantic space of the documents. We can also see that this approach of finding the LSS of documents is different than the LSI technique discussed in section 2.2 above. Some of the important properties of simplexes as taken from wikipedia along with the idea of LSS topology are:

1. Any face of a simplex from is also in the simplex.
2. The intersection of any two simplices is a face of both simplices.
3. A simplex represents a primitive concept.
4. A maximal dimensional simplex will represent a maximal primitive concept.
5. A connected component will represent a complete concept.

## 3.3 Simplicial Geometry of Keywords

One great use of this approach can be seen in dealing with document sets of different languages. In this paper I am using keyword combinations that are filtered from the document collection because this process is simple and automatable. We know that

simplicial complex is an n-dimensional polyhedron in Euclidean space [1]. The research paper [1] also says that the interaction of keywords within a document can be captured in a simplicial complex. These interactions are reflected in the geometry of a simplicial complex. Using this approach a simplicial complex can be generated for the document set of different languages. There is a very good chance that the polyhedron made by different language document sets will exhibit homeomorphism because the shape of the geometrical structure defines the semantics and hence there won't be a need for human translation. This can help identify semantically similar documents of different languages without the use of human translators (implementation not provided here).

# 4 Implementing the Concept Analysis Algorithms

## 4.1 Concept Analyzer

As part of a search engine, concept Analysis can be seen as a process that works on the data downloaded by the crawler from internet. Data that is downloaded from the internet usually has structural as well as metadata with it too. The concept analyzer's scope of work do not require understanding of the metadata and structural information hence the data downloaded from the internet also needs to be massaged before concept analysis can be done on it. Massaging of the downloaded data is done by a separate helper program. The massaged data will be in textual form and stored in lots of text files. For sake of simplicity the concept analyzer is implemented in a way that the root folder of the massaged data (the text files) needs to be specified. The program will then read all the data files in the root folder and all its subfolders recursively. This program is tested on various data sets comprising of document collection from 20,000 to 50,000 documents.

## 4.2 Design of Project

There are three major steps that constitute the whole functionality of concept analyzer

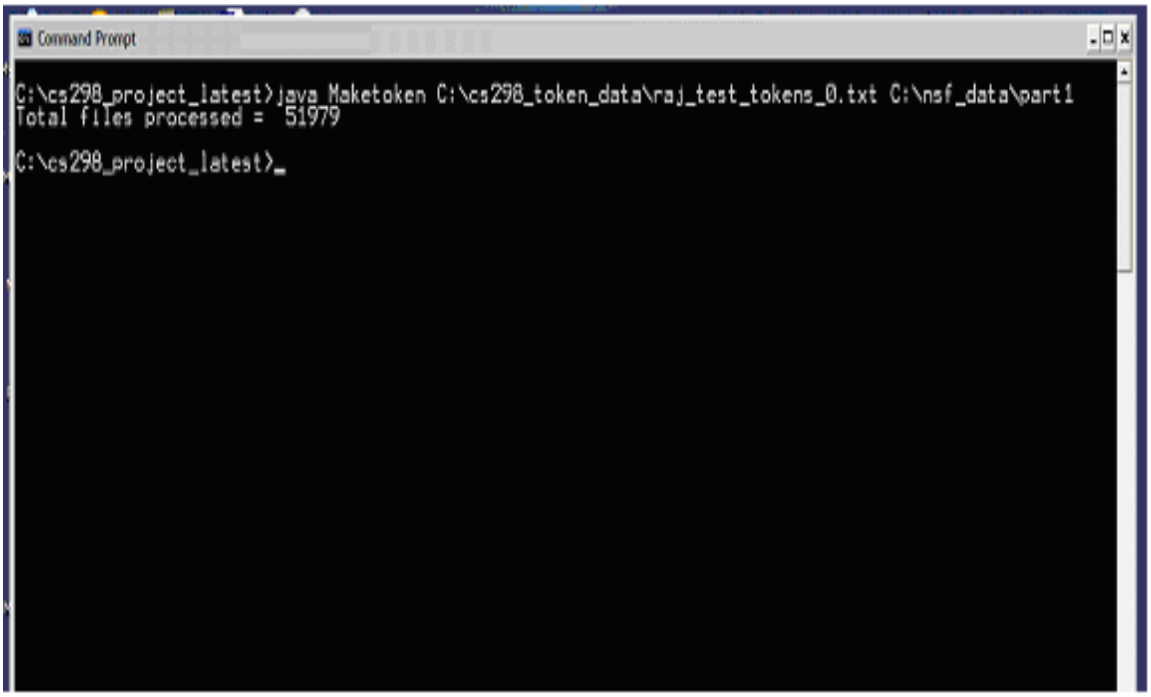
1. Tokenizing the data.
2. Creating the simplexes.
3. Finding the concepts.

### 4.2.1 Tokenizing the Data

The starting point for tokenizing is cleaned data after massaging so that all the structural and metadata information is absent form the data files (text files). Every word in a document is read and its position within that document is recorded along with the document name. This is done for all the documents in the document collection. In one variation some words will be discarded like articles, preposition, conjunctions, pronouns, and verbs. The tokenizing program is written in java and run on the command prompt as shown below.

Command window>java Maketoken  output_pathname  input_folder

In the above command Maketoken is the class file that is run to tokenize the data. The program takes two command line arguments. The first argument, output_pathname, is the fully qualified name of the files that will contain the tokens after program (Maketoken) has finished execution. Each line will contain the document name, keyword (token), and the position (offset) within a document. The second argument, input_folder, is the root folder that contains all the documents of the document collection under study.




### 4.2.2 Creating the simplexes

This is the most time taking exercise in the whole process. Simplex creation is done using SQL-92. The whole process of simplex creation is as follows. The tokens that gets created, as defined in section 4.2.1, are read into a SQL table such that each row in the table contains the document name, token, and position. This will be a huge table with few million records for data size of 20,000 files or above. Simplex generation through SQL 92 is an iterative process that needs to be done in successive steps, which also puts severe restrictions on processing the whole table data in one pass. So data needs to be pruned as early as possible [7], since this table is the first one so pruning will start from here itself. The approach used for pruning data from this table is TFIDF (Term frequency Inverse document frequency). There are several flavors of TFIDF algorithm or formula. I will use the one mentioned at the online Wikipedia. According to the wikipedia "*The tf-idf weight (term frequency-inverse document frequency) is a calculated value which is used in information retrieval and text mining. This value is a statistical measure that is used to evaluate how important a word is to a document with respect to the whole document collection. According to this notion the calculated value or weight is directly proportional to the number of times a word appears in the document and somewhat inversely proportional to the number of documents, which offsets the calculated value. Search engines use different variations of tf-idf weighting schemes to rank documents based on a given user query.*
*Frequency of a term in a given document simply means the number of times that term appears within that document. But taking this frequency will lead to a bias towards longer documents (longer document can lead to higher count regardless of the term's overall importance in that document), so this frequency is normalized to give a correct measure of the importance of the term ti for that particular document*".

$$tf_i = n_i / \Sigma_k n_k$$

In the above formula $n_i$ represents the number of times a concerned word appears in a document and the denominator $\Sigma_k n_k$ represent the total number of all terms in that document.

*"The inverse document frequency (idf) is statistical quantity that gives the general importance of the term. It is obtained by dividing the total number of documents by the number of documents containing the term, and then taking its logarithm"*

$$idf_i = Log ( |D| / |\{d : t_i \varepsilon d\}| )$$

Here $|D|$ represents total number of documents in the collection, and $|d : t_i \varepsilon d\}|$ is the number of documents where the term $t_i$ appears.
Therefore we have the final formula by multiplying the above two equations as below.

$$TFIDF = tf_i * idf_i$$

The above formula will be used to calculate weight of all the terms (tokens) in the table and then only terms that have a TFIDF value higher than a certain value will be considered. This step will prune the table significantly and the right value of TFIDF will be considered after trying several values and looking at the final result. It is not possible to get a universally correct value that would work in all the circumstances, as it will depend on the document sizes as well as the total number of documents in the collection.

Once pruning is done based on TFIDF values the table (SQL table) will contain all the important tokens such that each row will have the document name, token, and its position. Now I will apply the apriori principle [6] of data mining on the table data so that in the end we can get simplexes from the tokens. The apriori approach is done by pairing all the tokens that are equal to or less than 'd' distance apart. Again there is no universally correct value of 'd' so I have chosen d = 5 for my experiment for which the results are very reasonable. The result of applying the apriori approach on the pruned SQL table will generate a SQL table that will have document name, token1, token2, pos1, pos2, and diff (pos2 – pos1) in each row. One of the problems with this approach is that it will again cause the table to swell, but fortunately for us we can prune this table too based on our notion of concept, mentioned in section 2.2, that says about commonly occurring keyword patterns in multiple documents of a document set [1]. The pruning again here would require some sort of quantitative criterion for which again there is no universal rule. I am assuming that keyword patterns occurring in 20 or more documents for my document collection are important so rest of them can be ignored.

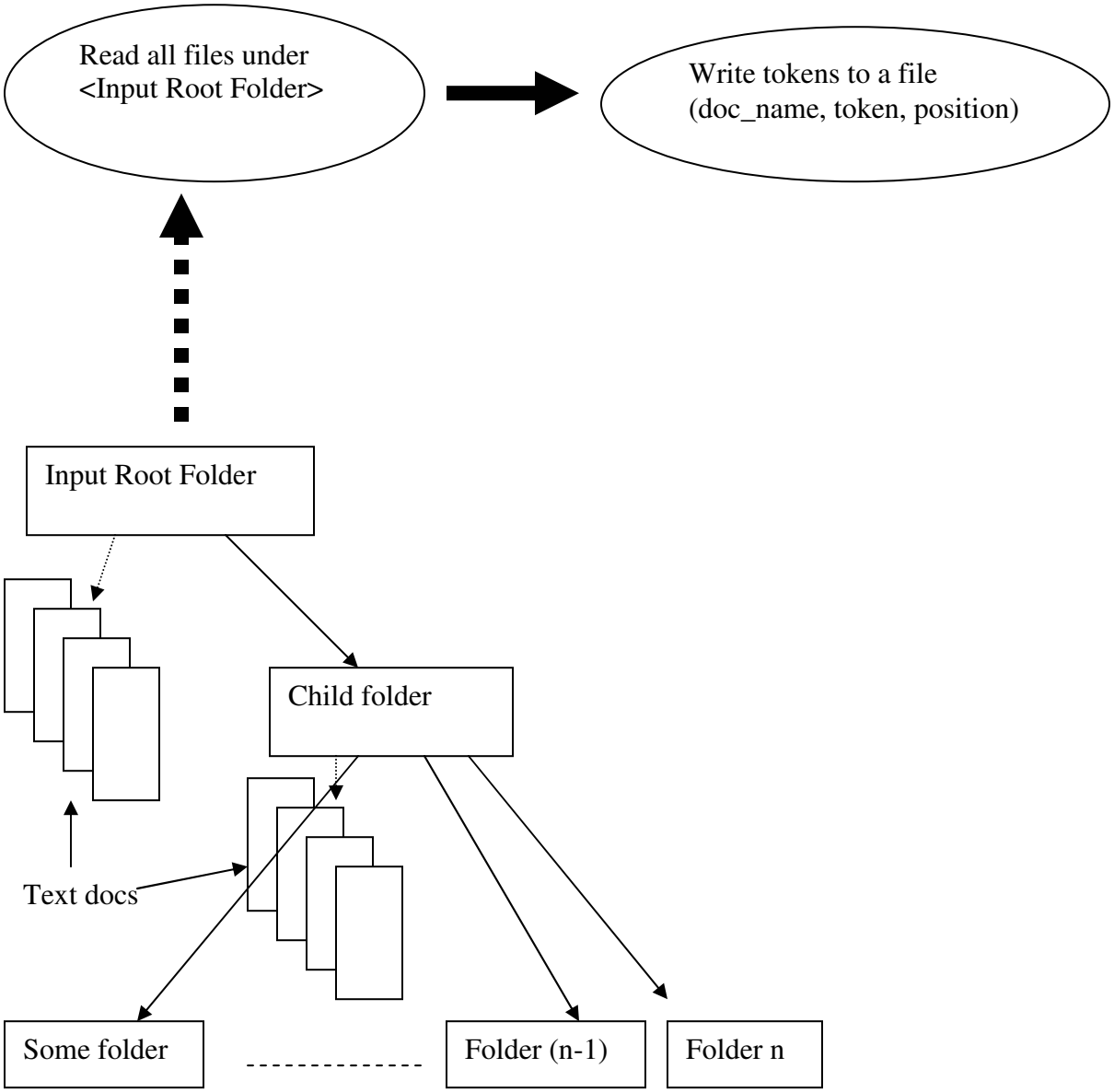
At this stage what we have is a pruned table of keyword pairs. This table can be used to find higher pairing of keywords i.e. using the n-pair tokens to get (n+1)-pair tokens. For example, a 2-pair tokens table will be used to give a 3-pair tokens by making SQL self joins on the tokens, document name, and their positions. This step can be successively performed to get higher token pairing. At each step of generating (n+1)-pair tokens by n-pair tokens table, the (n+1)-pair tokens table can be pruned by using the notion of section 2.2 of commonly occurring patterns in multiple documents of a document set. Successive pruning will significantly reduce the query execution time for higher order pairing. I will continue this process till getting 5-pair tokens. While generating all the token pairs, I will not only keep the final result pair tokens but also the intermediate token pairs. For example if my final resulting pair is 5-pair tokens table then I will also keep 4-pair, 3-pair, and 2-pair tokens table respectively. These tables will then be used to give 4-simplex, 3-simplex, 2-simplex, and 1-simplex respectively. The simplexes will be stored in separate text files, depending on the simplex size (n-simplex), where each line will contain the document names and the respective keyword pairs or group. There will be 4 separate files for all the four different simplex size, as mentioned above, respectively. These simplexes will be further used by the project to find *concepts* by running the graph theory approach and geometrical approach respectively.

## *4.3 Project Flow Charts.*

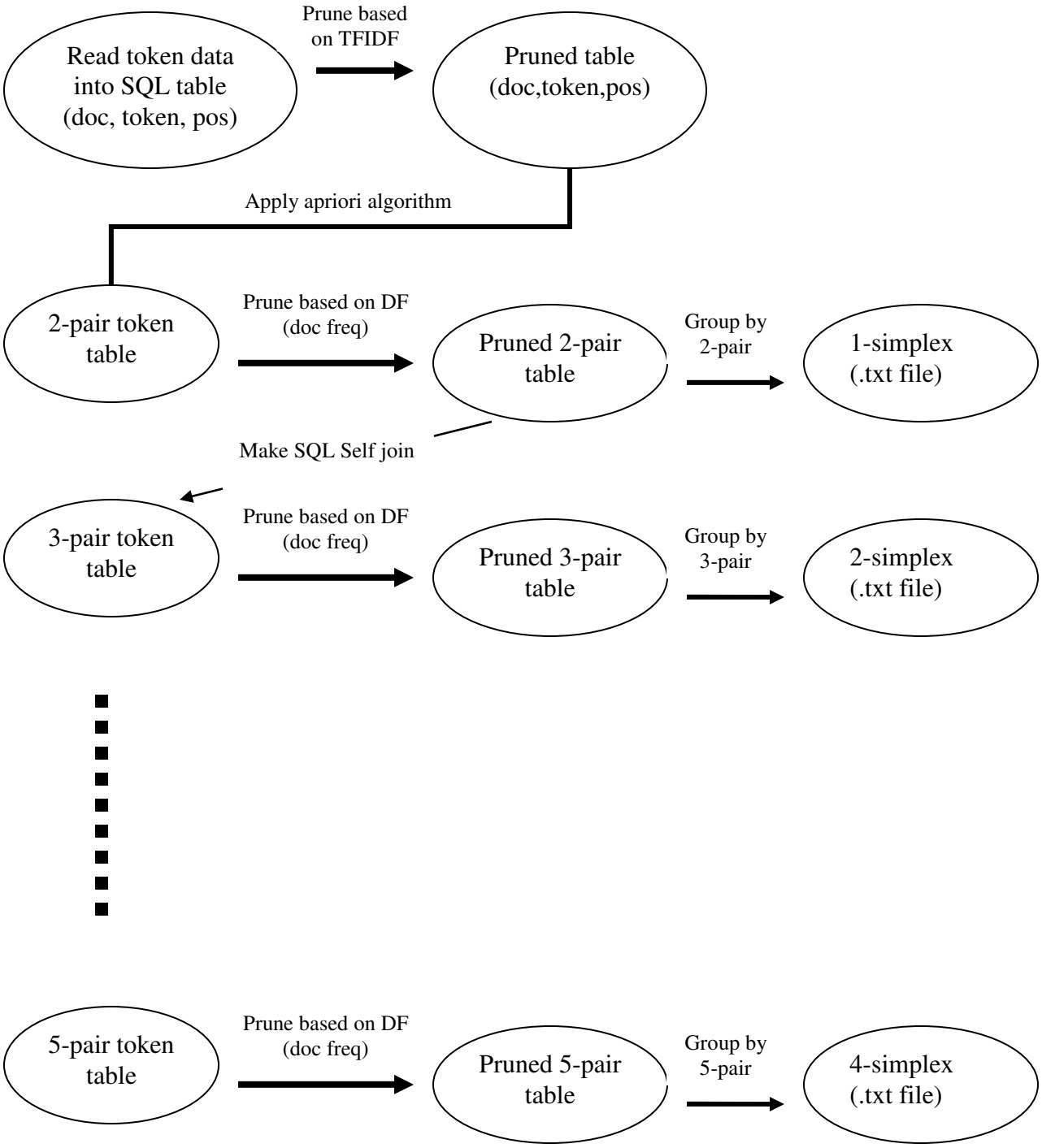
There are three major parts of the project

1. Tokenizing
2. Simplex Creation
3. Finding Concepts

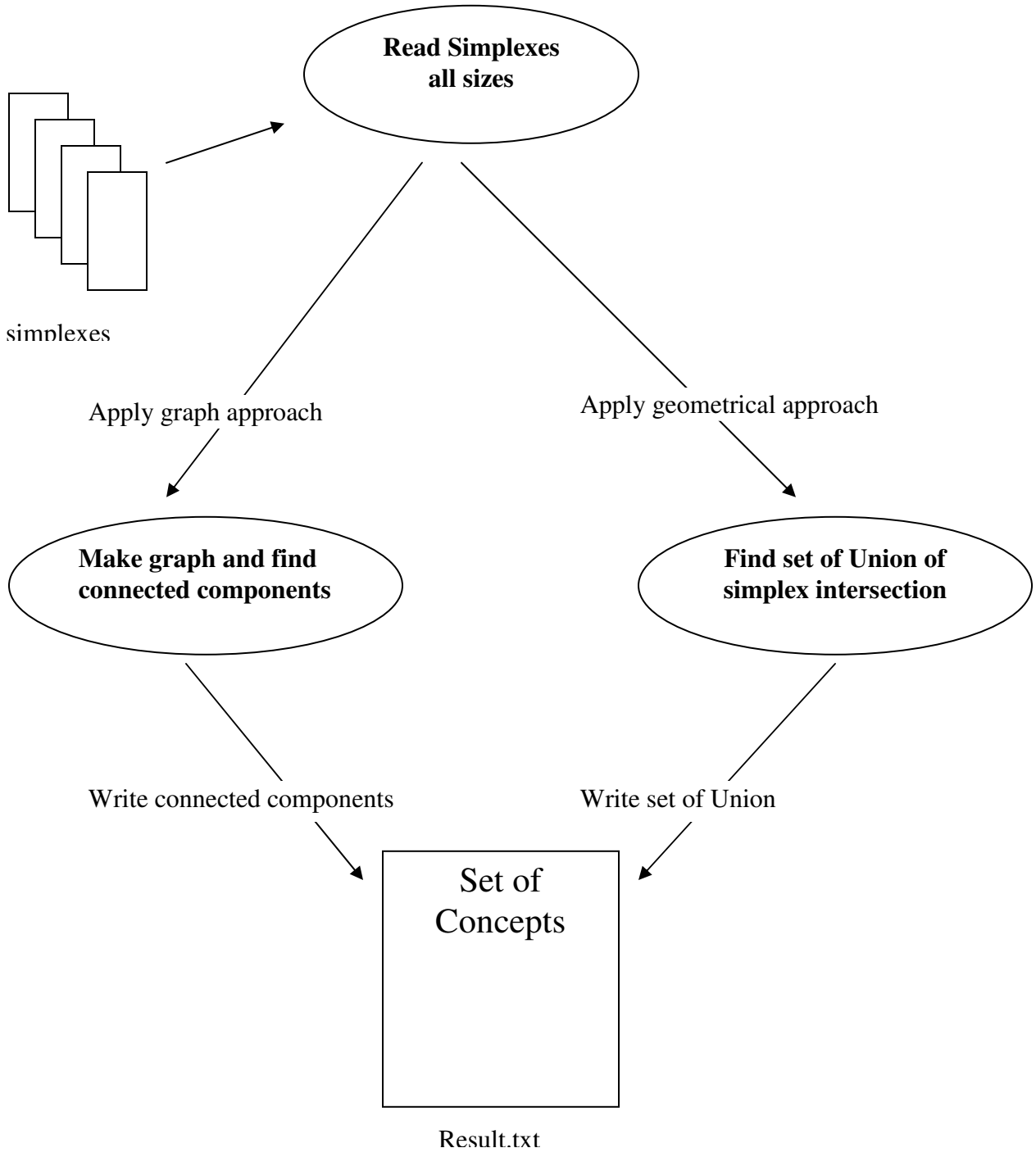
### 4.3.1 Tokenizing Flow Diagram



### 4.3.2 Simplex Creation Flowchart



### 4.3.3 Finding Concepts Flowchart



## 4.4 Graph Theory Approach

As mentioned under section 3.3, the interaction of keywords within a document can be captured in a simplicial complex. Since simplicial complex is a polyhedron in Euclidian space [2], it is too complex for human analysis. A simplicial complex can be reduced to a graph where each keyword set of a simplex will be a vertex and their relation, also called a face-off, will be shown by an edge between the vertices (keyword set). Here in my approach the relation, which is represented by an edge in the graph, will be ascertained between two keyword set if one is a subset of the other. For example, consider a simplicial complex that contains the keyword sets {ABCDE}, {UVWXY}, {ABE}, {BCE}, {DCB}, and {B}. In a graph representation all the six keyword sets will be represented by a vertex. According to our definition of relation between these vertices following edges will exit.

1. {ABCDE} ----------{ABE}
2. {ABCDE} ---------- {BCE}

The explanation for the two edges shown above is as follows:
The first edge is between {ABCDE} to {ABE} because we can see that {ABE} is a subset of {ABCDE} plus the relative ordering of A, B, and E is same in both keyword set. We will not consider {B} here because we cannot find its relative order. Remember we are interested in the interaction of keywords within a document and hence their ordering is important. Also note that {DCB} is also a subset of {ABCDE} but the relative ordering of D,C, and B are different in both the keyword set so they will not form an edge.
One of the advantages of graph theory approach lies in the fact that it can be used even if the keyword set within a simplicial complex do not form a closed simplex. A closed simplex will contain all its subsets. In this approach the simplexes (that were generated from the input data according to the process mentioned in section 4.2.2) will be used to construct a graph. Graph construction requires reading all the keyword sets as vertices and then finding edges between them. Once all the edges have been found the algorithm to find connected component of a graph [6] can be run. In this project we are more interested in finding connected component that encompasses the maximal dimension simplexes. It is my anticipation that connected components containing maximal dimension simplexes will be more precise and crisp in clustering the document set.

In the above example of keyword sets and edges we will have the following connected components.

1. {ABCDE}, {ABE}, {BCE}.
2. {UVWXY}.
3. {DCB}.
4. {B}.

We are only interested in the maximal dimensional simplexes so we will discard {DCB} and {B}. Therefore, our result will contain connected components represented by {ABCDE} {ABE} {BCE} and {UVWXY} respectively. The documents represented by these connected components should be semantically similar as per the hypothesis of this project. The connected components can be made more precise and crisp with respect to

semantic clustering by discarding the lower dimensional simplexes while constructing the graph. Suppose the maximum dimensional simplex is of 5-keyword (4 – simplex) then we may decide to discard simplexes that are smaller than 3-keyword (2 – simplex). This can improve clustering in a sense that there will now be fewer documents represented by that connected component but at the same time the semantic similarity of these documents will be high.

## 4.5 Geometrical Approach

This method can be used to find the concepts only if the keyword sets generated from the document, as mentioned in section 4.2.2, forms a closed simplex. This method is based on properties of set theory namely intersection and union except that it also takes into account ordering which is insignificant in set theory.

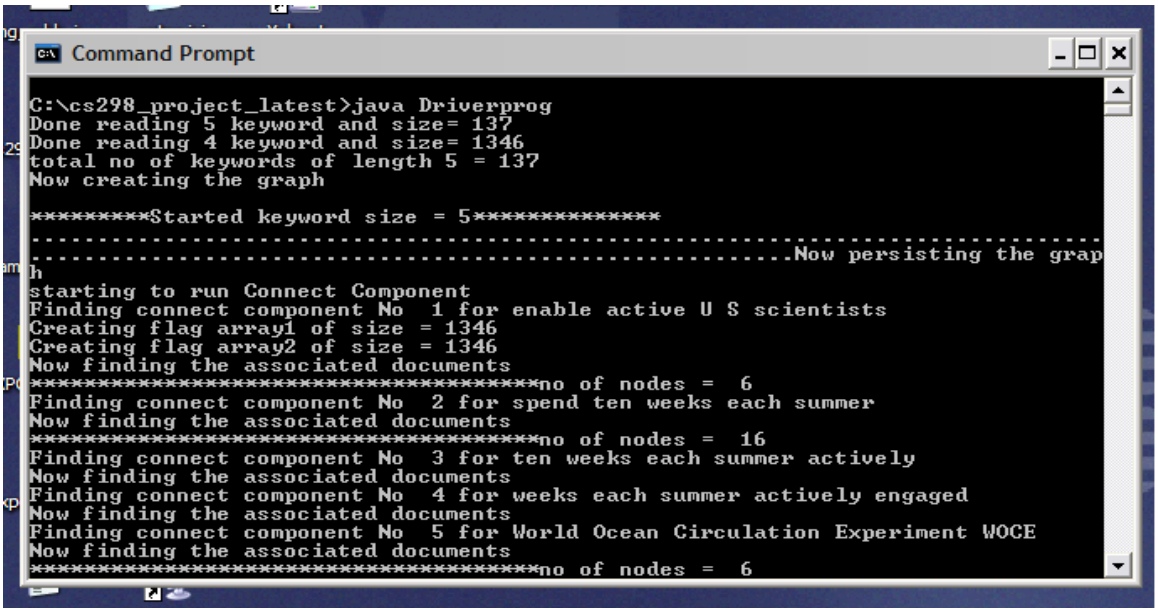
The exact algorithm of this approach can be understood by looking at the following example. Let us consider the keyword sets {ABCDE}, {UVWXY}, {ALMNB}, {CRSTU}, {MNQIJ}, {U}, {A} gets generated after performing steps mentioned in section 4.2.2. The algorithm starts by reading the first keyword set say {ABCDE} and stores it as an intermediate concept. In the next pass the program reads the next keyword set which is {UVWXY} and tries to find relationship with the existing concepts. The relationship is determined by searching for a common subset between the current keyword set and the existing concepts. Since the there are no common subsets between the existing intermediate concepts ({ABCDE}) and the present one ({UVWXY}) {UVWXY} will be stored as another intermediate concept. In the next pass the program will read {ALMNB}, which will be compared with all the existing concepts. We see that {ABCDE} and {ALMNB} both have a common subset which is {AB} because they both have {AB} with their relative ordering preserved (A comes before B). Therefore {ALMNB} will stick with {ABCDE}. After the third pass the intermediate concepts available will be ({ABCDE} U {ALMNB}) and {UVWXY}. We will continue this approach till all the keyword sets are consumed where by all the intermediate concepts during the program run will become the *concepts* after program has finished running. To sum it up the concepts that will form in the above example at the end of the program execution are: {ABCDE} U {ALMNB} U {MNQIJ}, {UVWXY}, {CRSTU}. We will discard the single keyword terms {U} and {A}. One interesting scenario that may happen in this approach is that if a keyword set has subset match relationship with more than one intermediate concept then the matching intermediate concepts will merge to form a union. The same approach as mentioned above in section 4.4 about maximum keyword size to minimum keyword size (discarding simplexes below a certain size, only considering subset match between max and min simplex size) can be applied here too for crisper and precise clustering of documents on the basis of their semantics.

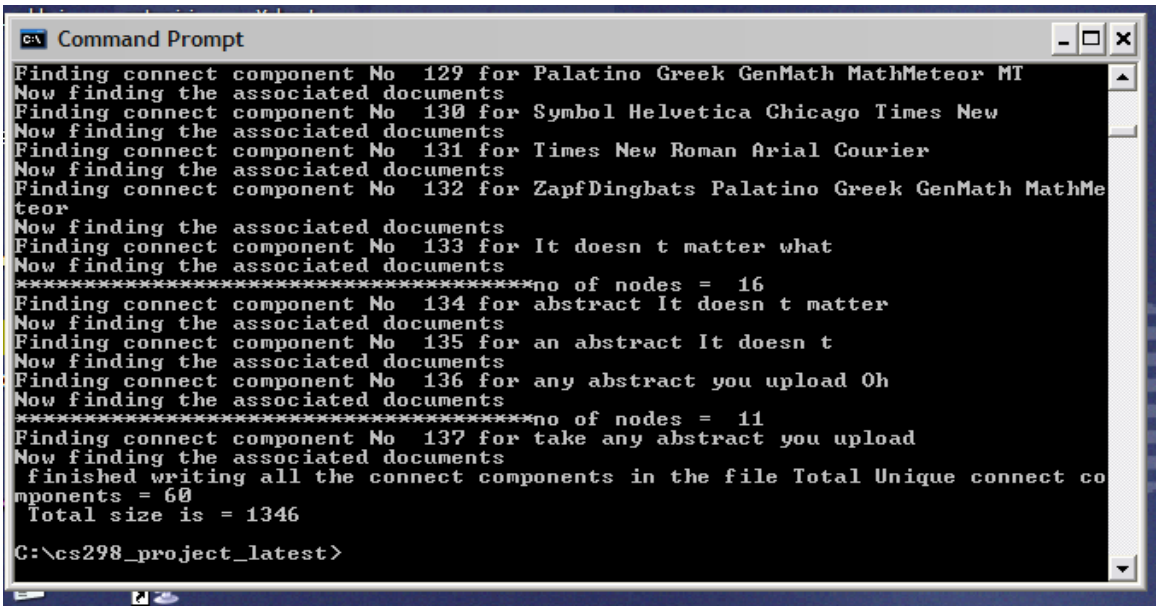
# 5 Test Results

The data used for this test run, Abstracts_part1.zip, was taken from the site http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html. The zip file is a collection of approximately 51,000 files.

## 5.1 Effect of Simplex Size

a) Results from running the program considering 5−keyword set (4−simplex) and 4−keyword set (3−simplex).



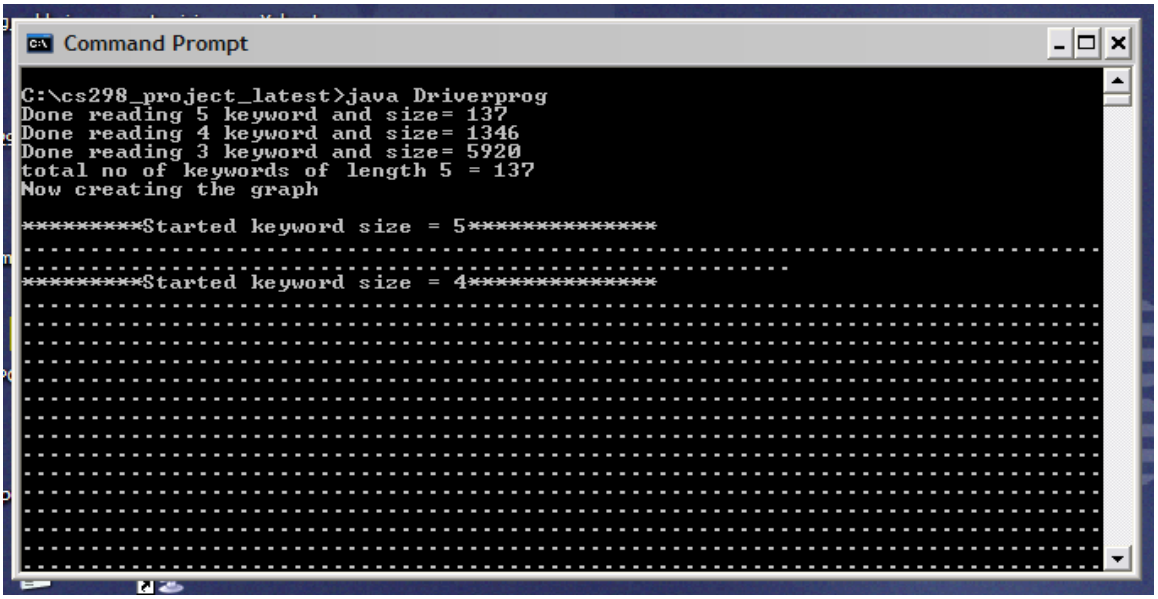



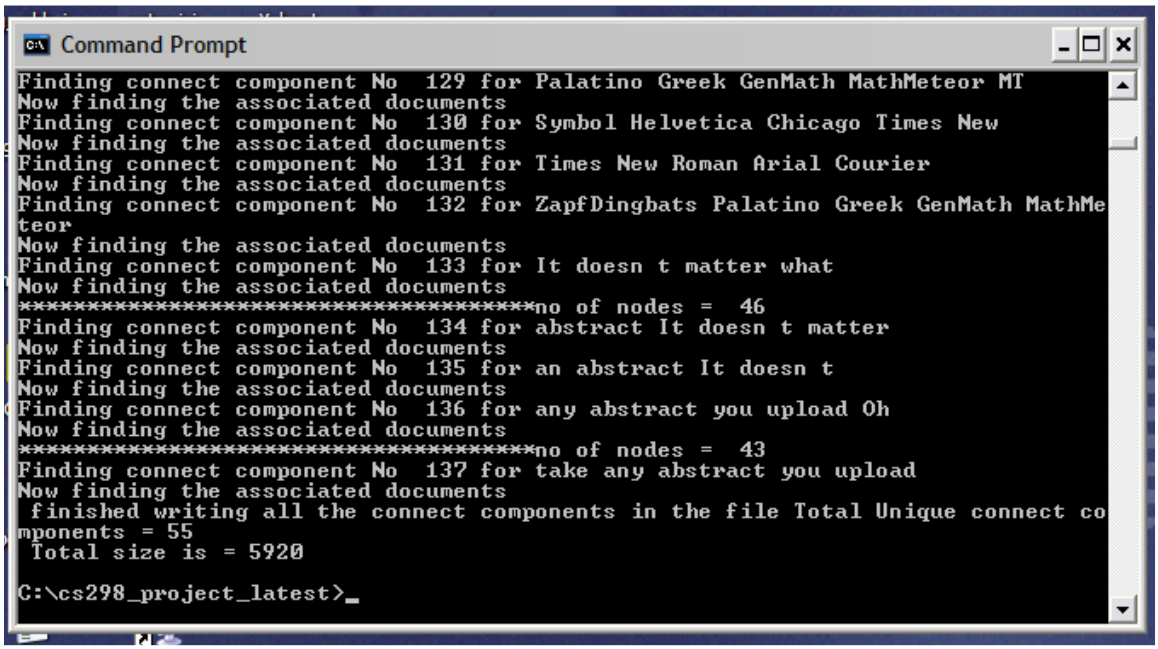
As shown in the above screen shots of the test run following are the results.

Total number of 5-keyword sets = 137.
Total number of 4-keyword sets = 1346 – 137 = 1209

The total number of connected components (***concepts)*** = 60

b) Results from running the program considering 5–keyword set (4–simplex), 4–keyword set (3–simplex), and 3–keyword set (2–simplex).



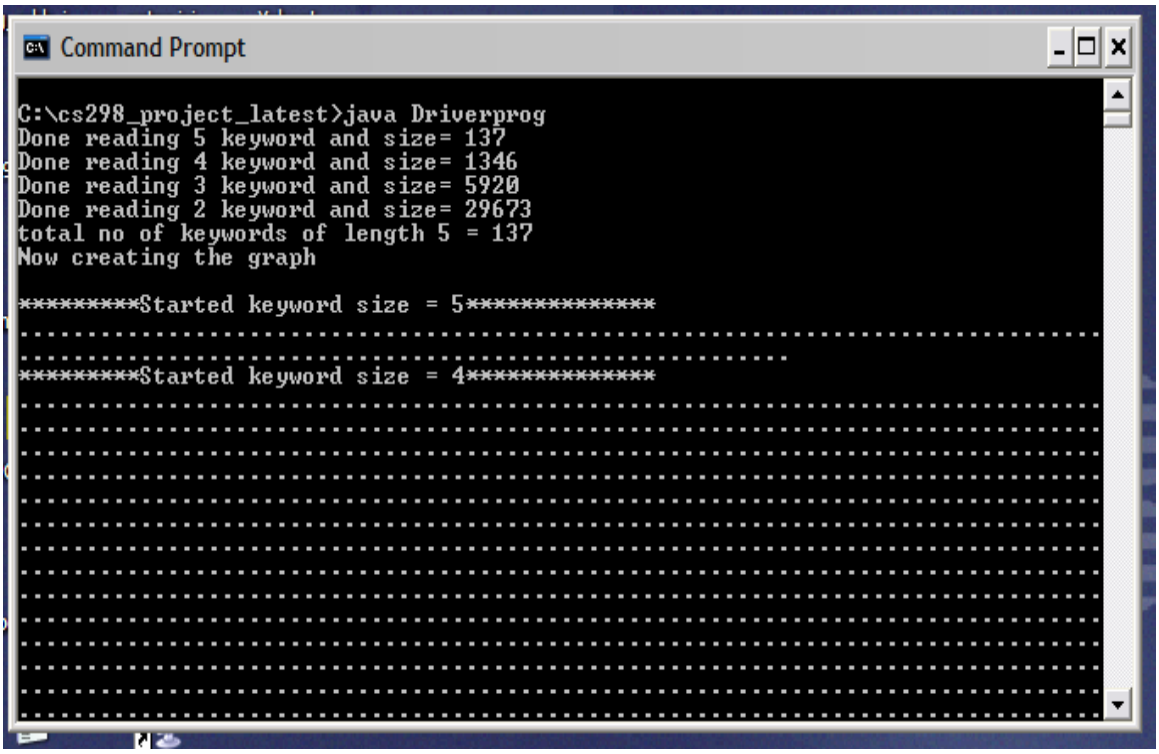



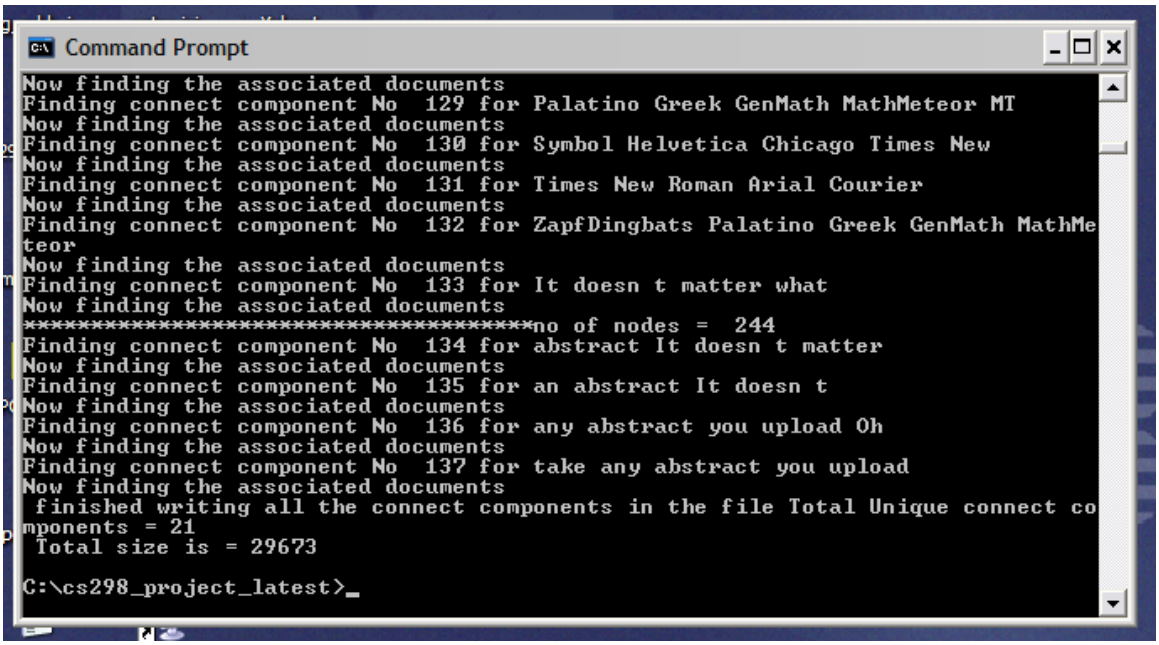
The screen shots of the test run with 5, 4, and 3 keyword sets is as follows.

Total number of 5-keyword sets = 137.
Total number of 4-keyword sets = 1346 – 137 = 1209
Total number of 3-keyword sets = 5920 – 1346 = 4574

The total number of connected components (***concepts)*** = 55

c) Results from running the program considering 5–keyword set (4-simplex), 4–keyword set (3–simplex), 3–keyword set (2–simplex), and 2–keyword set (1–simplex).






The screen shots of the test run with 5, 4, 3, and 2 keyword sets is as follows.

Total number of 5-keyword sets = 137.
Total number of 4-keyword sets = 1346 – 137 = 1209
Total number of 3-keyword sets = 5920 – 1346 = 4574
Total number of 2-keyword sets = 29673 – 5920 = 23753

The total number of connected components (***concepts)*** = 21

| S.No | Keyword sets | Connected components |
|---|---|---|
| 1 | 5 keyword set, 4 keyword set | 60 |
| 2 | 5 keyword set, 4 keyword set, 3 keyword set | 55 |
| 3 | 5 keyword set, 4 keyword set, 3 keyword set, 2 keyword set | 21 |

**Table 5.1**

In the summary table (Table 5.1), for the three test run scenarios, we see that as the number of keyword set group size (column name Keyword sets) increases the total number of connected components also decreases. This implies that as the keyword set group becomes bigger the resulting connected component also becomes bigger (i.e. represents more documents) there by reducing the total number of unique connected components. The bigger each connected component becomes less precisely it represents the concepts of all the referenced documents, conversely the smaller a connected component becomes more precisely it represents the concepts of the referenced documents.

## 5.2 Changing Association Rules for Simplex Generation

We know that most of the languages have some rules which we commonly called as grammar. We know by our experience of English language that there are certain words that are used within the document so that a sentence adheres to a certain predefined structure. For example, according to English grammar laws every sentence must have a verb. A document (text document) is a collection of sentences and it is very likely that all the verbs used in all the sentences respectively may not contribute enough towards the semantics of that document. The same can be said about other grammatical constructs such as prepositions, pronouns etc. As per our hypothesis the semantics of a document will depend upon the interaction of keywords so we can neglect the words which are present because of the language grammar's requirement. This approach can be applied to documents of other languages too and relevant grammatical construct enforcing words can be neglected. We will apply this approach on English language text documents and neglect very common words as prepositions, pronouns, verbs etc and then analyze the end result by running our algorithm.

The words that were neglected for this test run are: *a, the, an, his, he, her, him, has, she, if, for, of, by, it, its, is, at, to, be, but, and, this, that, they, and them*. Neglecting these words effect the association rules for simplex generation which can be explained with an example as follows. Suppose a document contains a phrase "*hazards of earthquake",* since we are interested in near by keywords we can consider the relative positions of the words "*hazards*" and "*earthquake*" to be **n** and **n+1** respectively after neglecting the word "*of*". If we don't neglect the word "*of*" then the relative positions of the two words ("*hazards*" and "*earthquake*") will be **n** and **n+2.** Therefore the simplex generating algorithm, mentioned in section 4.2.2, will generate "*hazards earthquake*", as one of its two keyword set when the word "*of*" is neglected, or "*hazards of earthquake*", as one of its three keyword set when the word "*of*" is not neglected.
The test run results on the same data, similar to section 5.1, but under these changed simplex generation association rules are as follows.

a) Results from running the program considering 5−keyword set (4−simplex) and 4−keyword set (3−simplex).

```
C:\cs298_project_latest>java Driverprog
Done reading 5 keyword and size= 597
Done reading 4 keyword and size= 4050
total no of keywords of length 5 = 597
Now creating the graph

*********Started keyword size = 5**************
.......................................................................................
.......................................................................................
.......................................................................................
...........
C:\cs298_project_latest>
```

```
*************************************no of nodes =  6
Finding connect component No  589 for Abstract done override system error
Now finding the associated documents
*************************************no of nodes =  11
Finding connect component No  590 for Fellowship Abstract done override system
Now finding the associated documents
Finding connect component No  591 for The recipient may choose Research
Now finding the associated documents
Finding connect component No  592 for allows for full time support
Now finding the associated documents
Finding connect component No  593 for for full time support any
Now finding the associated documents
Finding connect component No  594 for months The recipient may choose
Now finding the associated documents
Finding connect component No  595 for summer months The recipient may
Now finding the associated documents
Finding connect component No  596 for the mathematical sciences expand their
Now finding the associated documents
Finding connect component No  597 for two summer months The recipient
Now finding the associated documents
 finished writing all the connect components in the file Total Unique connect components = 153
 Total size is = 4050

C:\cs298_project_latest>
```

From the above shown screen shots of the test run we see the following results.

Total number of 5-keyword sets = 597.
Total number of 4-keyword sets = 4050 – 597 = 3453
The total number of connected components (***concepts)*** = 153

b) Results from running the program considering 5–keyword set (4–simplex), 4–keyword set (3–simplex), and 3–keyword set (2–simplex).

```
C:\cs298_project_latest>java Driverprog
Done reading 5 keyword and size= 597
Done reading 4 keyword and size= 4050
Done reading 3 keyword and size= 12974
total no of keywords of length 5 = 597
Now creating the graph

*********Started keyword size = 5**************
.......................................................................................
.......................................................................................
.......................................................................................
```

```
*************************************no of nodes =  24
Finding connect component No  589 for Abstract done override system error
Now finding the associated documents
*************************************no of nodes =  27
Finding connect component No  590 for Fellowship Abstract done override system
Now finding the associated documents
Finding connect component No  591 for The recipient may choose Research
Now finding the associated documents
Finding connect component No  592 for allows for full time support
Now finding the associated documents
Finding connect component No  593 for for full time support any
Now finding the associated documents
Finding connect component No  594 for months The recipient may choose
Now finding the associated documents
Finding connect component No  595 for summer months The recipient may
Now finding the associated documents
Finding connect component No  596 for the mathematical sciences expand their
Now finding the associated documents
Finding connect component No  597 for two summer months The recipient
Now finding the associated documents
 finished writing all the connect components in the file Total Unique connect components = 104
 Total size is = 12974

C:\cs298_project_latest>
```

The screen shots of the test run with 5, 4, and 3 keyword sets shown above is as follows.

Total number of 5-keyword sets = 597.
Total number of 4-keyword sets = 4050 – 597 = 3453
Total number of 3-keyword sets = 12974 – 4050 = 8924

The total number of connected components (***concepts)*** = 104

c) Results from running the program considering 5–keyword set (4-simplex), 4–keyword set (3–simplex), 3–keyword set (2–simplex), and 2–keyword set (1–simplex).

```
C:\cs298_project_latest>java Driverprog
Done reading 5 keyword and size= 597
Done reading 4 keyword and size= 4050
Done reading 3 keyword and size= 12974
Done reading 2 keyword and size= 54920
total no of keywords of length 5 = 597
Now creating the graph

*********Started keyword size = 5**************
.....................................................................................
.....................................................................................
.....................................................................................
C:\cs298_project_latest>
```

```
****************************************no of nodes =  37
Finding connect component No  589 for Abstract done override system error
Now finding the associated documents
****************************************no of nodes =  41
Finding connect component No  590 for Fellowship Abstract done override system
Now finding the associated documents
Finding connect component No  591 for The recipient may choose Research
Now finding the associated documents
Finding connect component No  592 for allows for full time support
Now finding the associated documents
Finding connect component No  593 for for full time support any
Now finding the associated documents
Finding connect component No  594 for months The recipient may choose
Now finding the associated documents
Finding connect component No  595 for summer months The recipient may
Now finding the associated documents
Finding connect component No  596 for the mathematical sciences expand their
Now finding the associated documents
Finding connect component No  597 for two summer months The recipient
Now finding the associated documents
finished writing all the connect components in the file Total Unique connect components = 18
Total size is = 54920

C:\cs298_project_latest>
```

Similarly, the screen shots of the test run with 5, 4, 3, and 2 keyword sets is as follows.

Total number of 5-keyword sets = 597.
Total number of 4-keyword sets = 4050 – 597 = 3453
Total number of 3-keyword sets = 12974 – 4050 = 8924

Total number of 2-keyword sets = 54920 – 12974 = 41946

The total number of connected components (***concepts)*** = 18.

| S.No | Keyword sets | Connected components |
|---|---|---|
| 1 | 5 keyword set, 4 keyword set | 153 |
| 2 | 5 keyword set, 4 keyword set, 3 keyword set | 104 |
| 3 | 5 keyword set, 4 keyword set, 3 keyword set, 2 keyword set | 18 |

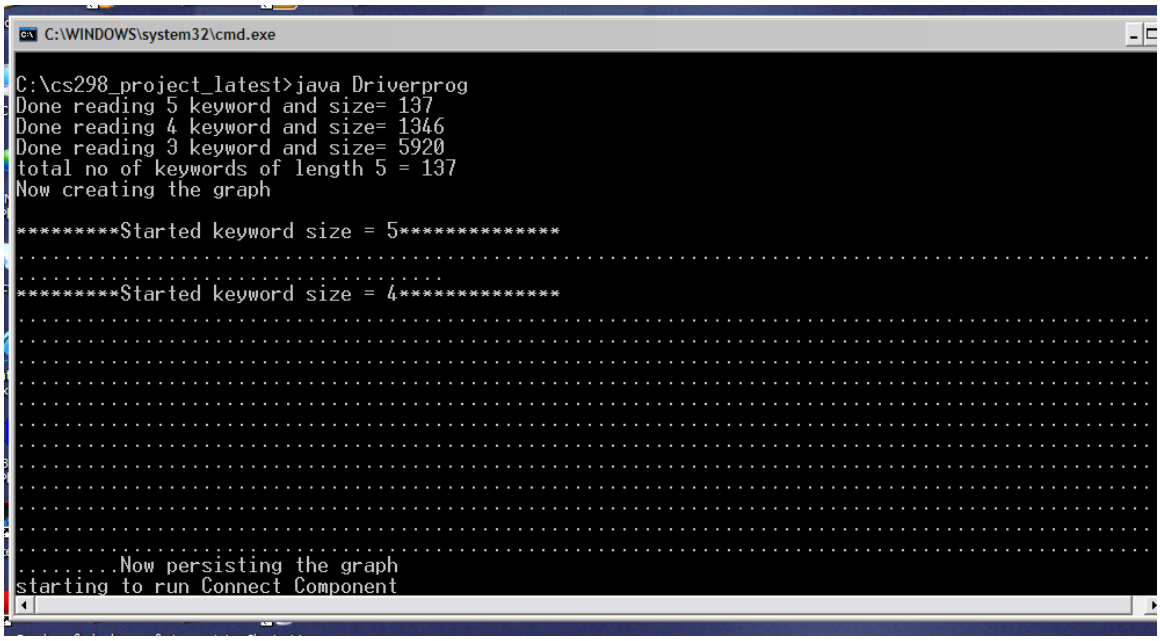
**Table 5.2**

Similarly in the summary table (Table 5.2), for the three test run scenarios, we see that as the number of keyword set group size (column name Keyword sets) increases the total number of connected components also decreases. This behavior is consistent with the observation in section 5.1 (Table 5.1) which says that as the keyword set group becomes bigger the resulting connected component also becomes bigger (i.e. represents more documents) there by reducing the total number of unique connected components. The bigger each connected component becomes less precisely it represents the concepts of all the referenced documents, conversely the smaller a connected component becomes more precisely it represents the concepts of the referenced documents. One additional inference that can be made by comparing the results in summary tables (Table 5.1 and Table 5.2) is that neglecting certain grammatical construct enforcing words leads to identification of more connected components or **concepts.** The correctness of these concepts will be ascertained by comparing it with the human notion of concepts.

## 5.3 Comparison of Graph Theory and Geometrical Approach

A comparative study about the runtime behavior of graphical and geometrical approach was performed using the same machine and data. The result metrics that were measured are execution time, the amount of free memory in the Java Virtual Machine after finishing all the major computations, and the total amount of memory in the Java virtual machine after finishing all the major computations. The java methods used for free memory and total memory measurements are Runtime.freeMemory(), and Runtime.totalMemory() respectively. The exact details and description of these methods can be found at http://java.sun.com/j2se/1.4.2/docs/api/.

a) Measurements on the graph theory approach. To run type as below:
Program Folder > java Driverprog (hit Enter)



```
C:\WINDOWS\system32\cmd.exe
Now finding the associated documents
Finding connect component No  130 for Symbol Helvetica Chicago Times New
Now finding the associated documents
Finding connect component No  131 for Times New Roman Arial Courier
Now finding the associated documents
Finding connect component No  132 for ZapfDingbats Palatino Greek GenMath MathMeteor
Now finding the associated documents
Finding connect component No  133 for It doesn t matter what
Now finding the associated documents
************************************no of nodes =  46
Finding connect component No  134 for abstract It doesn t matter
Now finding the associated documents
Finding connect component No  135 for an abstract It doesn t
Now finding the associated documents
Finding connect component No  136 for any abstract you upload Oh
Now finding the associated documents
************************************no of nodes =  43
Finding connect component No  137 for take any abstract you upload
Now finding the associated documents
 finished writing all the connect components in the file Total Unique connect components = 55
 Total size is = 5920
The total execution time = 3828 MilliSeconds
Memory Usage => Free = 6413512 total = 28441088 Max = 1072879616

C:\cs298_project_latest>
```

In the end portion of the second screen shot we can see that values of the three important performance metrics are:

Execution time = 3828 milliseconds.

Free memory in JVM = 6413512 Bytes.

Total memory in JVM = 28441088 Bytes.

b) Measurements on the geometrical approach. To run type as below:
Program Folder > java Driverprog1 (hit Enter)

```
C:\WINDOWS\system32\cmd.exe

C:\cs298_project_latest>java Driverprog1
Done reading 5 keyword and size= 57
Done reading 4 keyword and size= 270
Done reading 3 keyword and size= 1800
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
```

```
C:\WINDOWS\system32\cmd.exe
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
**************************************connect Component is
 finished writing all the connect components in the file Total Unique connect components = 55
The total execution time = 4391 MilliSeconds
Memory Usage => Free = 7661432 total = 46987776 Max = 1072879616

C:\cs298_project_latest>
```

Similarly in the end portion of the second screen shot we can see that values of the three important performance metrics are:

Execution time = 4391 milliseconds.

Free memory in JVM = 7661432 Bytes.

Total memory in JVM = 46987776 Bytes.

The comparison of the above results (from section (a) and (b) ) reveals that:

1. The execution time for graph theory approach (3828 msecs) is less than execution time for geometrical approach (4391 msecs). This means that graph theory approach for this implementation runs faster.

2. The free memory for graph theory approach (6413512 Bytes) is less than free memory for geometrical approach (7661432 Bytes). According to java API documentation free memory is an approximation to the total amount of memory currently available for future allocated objects. The free memory readings show that geometrical approach uses less memory than the graph memory.

3. The total memory for graph theory approach (28441088 Bytes) is also less than total memory for geometrical approach (46987776 Bytes). According to java API documentation total memory is the total amount of memory currently available for current and future objects. The total memory readings also show that with graph approach less total memory is available in the JVM than geometrical approach. Hence geometrical approach is more memory efficient than graph approach, a point also complimented by free memory reading above.

## 5.4 Comparison with Yahoo Desktop Search

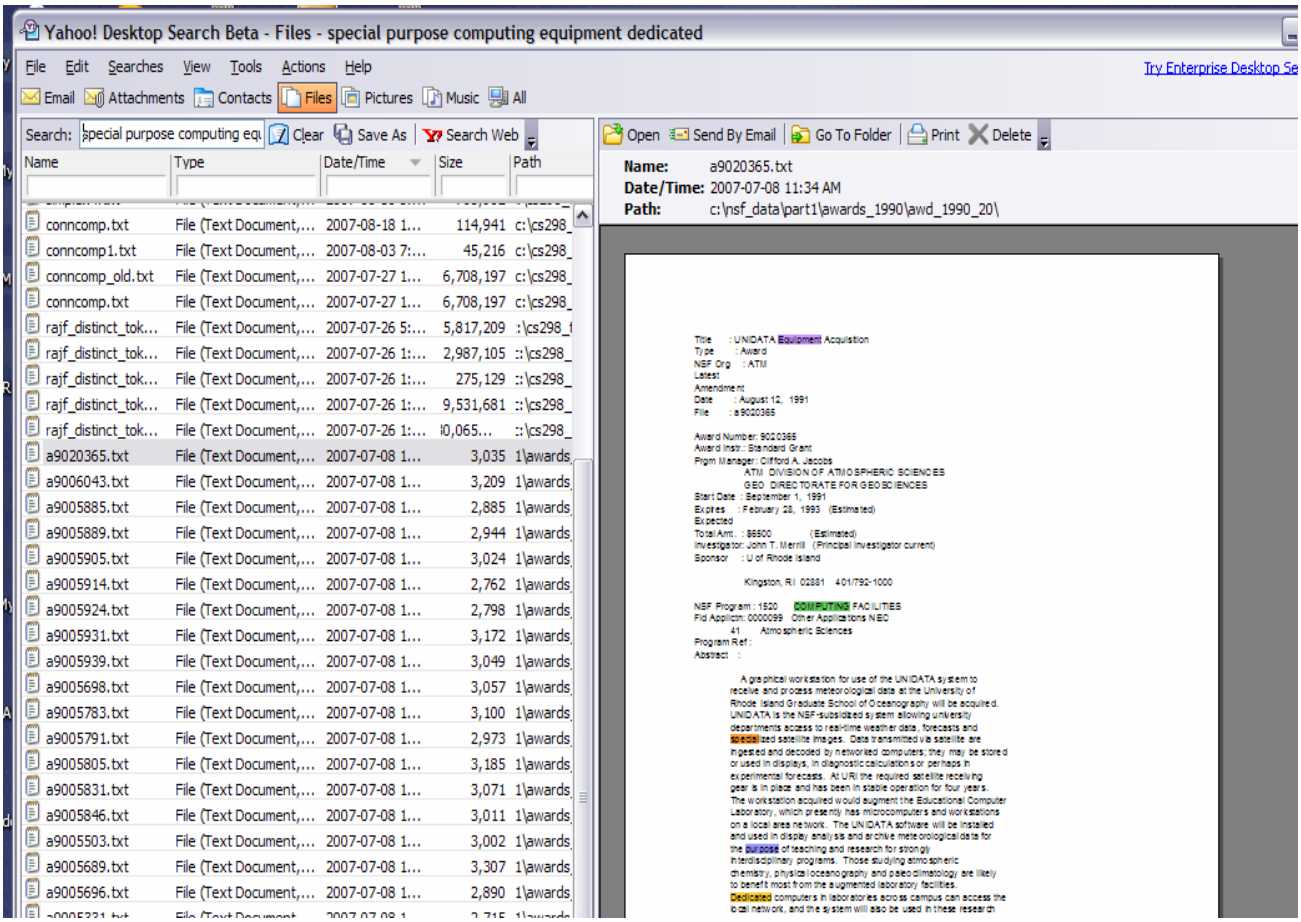
To gauge the accuracy of my project I have decided to compare it with a very popular text search engine namely Yahoo Desktop Search. Here are some of the observations.

a) The output of this project is saved in a text file (ConnComp.txt) which will contain the concepts. Each concept is composed of a collection of phrases plus some document names (from the original input document set). Taking a concept randomly from the output file and comparing it with yahoo desktop search is shown below.

Collection of phrase in that concept: "*special purpose computing equipment dedicated, purpose computing equipment dedicated, computing equipment dedicated, purpose equipment dedicated, purpose computing dedicated, purpose computing equipment, special computing equipment dedicated, special equipment dedicated, special computing dedicated, special computing equipment, special purpose equipment dedicated, special purpose dedicated, special purpose equipment, special purpose computing dedicated, special purpose computing, special purpose computing equipment, purchase special purpose computing, purchase purpose computing, purchase special computing, purchase special purpose*".

Document names: "a9003921.txt, a9005831.txt, a9005696.txt, a9005885.txt, a9005905.txt, a9005924.txt, a9003682.txt, a9216171.txt, a9003401.txt, a9005931.txt, a9004700.txt, a9005805.txt, a9004981.txt, a9005331.txt, a9005783.txt, a9005791.txt, a9004195.txt, a9005939.txt, a9005503.txt, a9006043.txt, a9005889.txt, a9260946.txt, a9001488.txt, a9005914.txt, a9005698.txt, a9003353.txt, a9004628.txt, a9005689.txt, a9005846.txt".

From the above collection of phrase I am taking the longest phrase "*special purpose computing equipment dedicated*" to perform a query in yahoo desktop search.
A screen shot of the yahoo desktop is also shown below.




The document names for the above search query "`special purpose computing equipment dedicated`" as given by yahoo desktop search are as below.

Yahoo Desktop Search Document Names: "`a9020365.txt, a9003921.txt, a9005831.txt, a9005696.txt, a9005885.txt, a9005905.txt, a9005924.txt, a9003682.txt, a9003401.txt, a9005931.txt, a9004700.txt, a9005805.txt, a9004981.txt, a9005331.txt, a9005783.txt, a9005791.txt, a9004195.txt, a9005939.txt, a9005503.txt, a9006043.txt, a9005889.txt, a9001488.txt, a9005914.txt, a9005698.txt, a9003353.txt, a9004628.txt, a9005689.txt, a9005846.txt`".

Comparison of the document names given by my project and yahoo desktop search reveals the following information.

1. The total number of documents returned by yahoo desktop search is 28.

The total number of documents returned by my project is 29.

The number of documents returned by both (common) is 27. There was discrepancy in the results of both the searches by 3 documents. One document that yahoo desktop search returned (a9020365.txt) was not returned by my project, on the other hand two

documents (a9216171.txt, and a9260946.txt) that were part of my document result set were not returned by yahoo desktop search. A human analysis of the document a9020365.txt can easily reveal that this document, which was picked by yahoo desktop search and discarded by my project, is not semantically close to other documents in the result set. This document was picked by yahoo desktop search simply on the basis of matching words **special**ized, **purpose**, and **dedicated.** To account for the documents a9216171.txt and a9260946.txt, which were returned by my project and discarded by yahoo, my human analysis finds them semantically closer to the other documents in the result set. The document a9216171.txt talks about video coding and image processing and hence can be easily accepted to be close to the query phrase "`special purpose computing equipment dedicated`". Similarly, document a9260946.txt talks about using some special purpose ground equipment in conjunction with GPS for aircraft landing system. We know by our common sense that quick computation is a must for highly skilled equipments like aircraft, missile systems etc. Therefore document a9260946.txt is also semantically close to the query string.

b) Comparing the results obtained by applying the association rule changes as mentioned in section 5.2 (discarding certain unimportant words). This time I decided to pick a concept that spans through higher number of files. As mentioned under section 5.4 part (a) above the concept which is a collection of phrases and document names is as follows.

Collection of phrase in that concept: "*National Earthquake Hazard Reduction Program, Earthquake Hazard Reduction Program, Hazard Reduction Program, Earthquake Reduction Program, Earthquake Hazard Program, Earthquake Hazard Reduction, National Hazard Reduction Program, National Reduction Program, National Hazard Program, National Hazard Reduction, National Earthquake Reduction Program, National Earthquake Program, National Earthquake Reduction, National Earthquake Hazard Program, National Earthquake Hazard, National Earthquake Hazard Reduction, National Earthquake Hazards Reduction Program, Earthquake Hazards Reduction Program, National Hazards Reduction Program, National Earthquake Hazards Program, National Earthquake Hazards Reduction, component National Earthquake Hazard, Hazards Reduction Program, Earthquake Hazards Program, Earthquake Hazards Reduction, National Hazards Program, National Hazards Reduction, National Earthquake Hazards, component Earthquake Hazard, component National Hazard, component National Earthquake, research component National Earthquake, research National Earthquake, research component Earthquake, research component National*".

Document names: "`a9204835.txt, a9416482.txt, a9417493.txt, a9319417.txt, a9004511.txt, a9011452.txt, a9206565.txt, a9117800.txt, a9011783.txt, a9416470.txt, a9111877.txt, a9118025.txt, a9405552.txt, a9003598.txt, a9105050.txt, a9224945.txt, a9405490.txt, a9218652.txt, a9119335.txt, a9112749.txt, a9404762.txt, a9104158.txt, a9105500.txt, a9408506.txt, a9116722.txt, a9096302.txt, a9002704.txt, a9014456.txt, a9011441.txt, a9416223.txt, a9003575.txt, a9416499.txt, a9409013.txt, a9218704.txt, a9018166.txt, a9018487.txt, a9105322.txt, a9117319.txt, a9412802.txt, a9415738.txt, a9118090.txt, a9104448.txt, a9416120.txt, a9416546.txt, a9304110.txt, a9018848.txt, a9416271.txt, a9213236.txt,`

a9200768.txt, a9416425.txt, a9205235.txt, a9004381.txt, a9004556.txt, a9011325.txt, a9305180.txt, a9406378.txt, a9406781.txt, a9405498.txt, a9416196.txt, a9011322.txt, a9022121.txt, a9205369.txt, a9105069.txt, a9023166.txt, a9304549.txt, a9117699.txt, a9219922.txt, a9416190.txt, a9011456.txt, a9003646.txt, a9011294.txt, a9416342.txt, a9011332.txt, a9114967.txt, a9316457.txt, a9017358.txt, a9019003.txt, a9011919.txt, a9018690.txt, a9011845.txt, a9017657.txt, a9303796.txt, a9121566.txt, a9416339.txt, a9416416.txt, a9315055.txt, a9096281.txt, a9416183.txt, a9416144.txt, a9205448.txt, a9105467.txt, a9117768.txt, a9105515.txt, a9104199.txt, a9304657.txt, a9118525.txt, a9115056.txt, a9005302.txt, a9117464.txt, a9117834.txt, a9219856.txt, a9304587.txt, a9118332.txt, a9017661.txt, a9418465.txt, a9416314.txt, a9316528.txt, a9206815.txt, a9416228.txt, a9105152.txt, a9418643.txt, a9018356.txt, a9405533.txt, a9219676.txt, a9005594.txt, a9019185.txt, a9219529.txt, a9316337.txt, a9011121.txt, a9405519.txt, a9205257.txt, a9004428.txt, a9116397.txt, a9416340.txt, a9418922.txt, a9316150.txt, a9004350.txt, a9118201.txt, a9416758.txt, a9019193.txt, a9005092.txt, a9205669.txt, a9117730.txt, a9416213.txt, a9416335.txt, a9009444.txt, a9304232.txt, a9219187.txt, a9004207.txt, a9011819.txt, a9017767.txt, a9296125.txt, a9405870.txt, a9405767.txt, a9206545.txt, a9117811.txt, a9417700.txt, a9005300.txt, a9105970.txt, a9316344.txt, a9118445.txt, a9415728.txt, a9416148.txt, a9206473.txt, a9011784.txt, a9205830.txt, a9416219.txt, a9104735.txt, a9204748.txt, a9204643.txt, a9304652.txt, a9004220.txt, a9418754.txt, a9304949.txt, a9410264.txt, a9105575.txt, a9011449.txt, a9205777.txt, a9305172.txt, a9011102.txt, a9416320.txt, a9011458.txt, a9411759.txt, a9201406.txt, a9105733.txt, a9220104.txt, a9014787.txt, a9416119.txt, a9223453.txt, a9100673.txt, a9316871.txt, a9011319.txt, a9316513.txt, a9004375.txt, a9416458.txt, a9011041.txt, a9418942.txt, a9417389.txt, a9103493.txt, a9416214.txt, a9208838.txt, a9219361.txt, a9116254.txt, a9196115.txt, a9405547.txt, a9003678.txt, a9205591.txt, a9118038.txt, a9316581.txt, a9304952.txt, a9418482.txt, a9416181.txt, a9118401.txt, a9416277.txt, a9116736.txt, a9118430.txt, a9412260.txt, a9215158.txt, a9206092.txt, a9404962.txt, a9022389.txt, a9004177.txt, a9105229.txt, a9317461.txt, a9416336.txt, a9017569.txt, a9305081.txt, a9011330.txt, a9118086.txt, a9418905.txt, a9117705.txt, a9415721.txt, a9304555.txt, a9304560.txt, a9315976.txt, a9207181.txt, a9216637.txt".

For query string comprising of the longest phrase from the concept's phrase collection "*National Earthquake Hazard Reduction Program*" yahoo desktop search returned a total of 237 documents. This is too big of a number to go over each document in detail so I will mention few documents that were picked up by yahoo desktop search engine but not by my project. Majority of documents returned by both the searches talk about research done by earthquake hazard reduction program whereas document a9001494.txt (returned only by yahoo desktop search) pertains to study of walls or buildings made of RC and document (returned only by yahoo desktop search) a9001256.txt pertains to structural control research for seismic and wind resistant design. The two documents a9001494.txt and a9001256.txt are definitely not semantically closer to the majority of documents returned by this search query. I am sure a further probe can yield some more such discrepancies in the results returned by Yahoo desktop search. By giving the examples of a9001494.txt and a9001256.txt documents one thing is certain that Yahoo desktop search is not very smart in terms of semantic search.

c) One more concept comparison from the same result as section 5.4 part (b) yields interesting results as shown below.

Collection of phrase in that concept: "`vessels specifically dedicated oceanographic research, specifically dedicated oceanographic research, dedicated oceanographic research, specifically oceanographic research, specifically dedicated research, specifically dedicated oceanographic, vessels dedicated oceanographic research, vessels oceanographic research, vessels dedicated research, vessels dedicated oceanographic, vessels specifically oceanographic research, vessels specifically research, vessels specifically oceanographic, vessels specifically dedicated research, vessels specifically dedicated, vessels specifically dedicated oceanographic`".

Document names: "`a9000251.txt, a9300636.txt, a9000246.txt, a9314910.txt, a9000393.txt, a9000158.txt, a9302587.txt, a9001169.txt, a9300825.txt, a9000312.txt, a9000463.txt, a9300411.txt, a9000048.txt, a9303344.txt, a9000343.txt, a9000049.txt, a9301213.txt, a9000130.txt, a9000046.txt, a9300503.txt, a9106232.txt, a9302254.txt`".

The query for the longest phrase from the above collection "`vessels specifically dedicated oceanographic research`" in yahoo desktop search yields a total of 22 documents. The interesting observation here is that the 22 documents returned by yahoo desktop search were same as mentioned above (which are returned by my project). So both the search techniques yielded similar results for this query.

# 6 Conclusion

In order to help search engines give more meaningful results to a user's query based on the semantics rather than just the textual match, I propose a novel approach of concept analysis so that documents can be clustered into groups such that the documents in each group are semantically similar. The principle idea behind the work is that a document can be seen as a collection of keywords where each keyword represents some human thought [2]. The interaction of these keywords leads to some concept formation, in other words capture the semantics. The semantics of a collection of documents can be structured into a simplicial complex [2]. One unique aspect of this work lies in the fact that ordering of keywords within a document is preserved which is not the case with most of the search engine implementations because they are based on single value decomposition (0-simplex). According to my hypothesis ordering of words (keywords) becomes very important with respect to semantics when discussing keyword interactions. For example the keyword pair "wall street" and "street wall" is semantically very different. The concept analysis algorithm identifies the concepts (a collection of phrases) plus the document names for each of the important concepts within a document collection. These concepts can be indexed (indexing not implemented) to answer semantics based search queries.

The two techniques used for concept analysis were graph theory approach and geometrical approach. In graph theory terms the concepts were represented by connected components after reducing simplicial complex to a graph structure. Graph theory approach is universally applicable where as geometrical approach can only be used in case of closed simplexes [3][6]. The test run results show that graph theory approach runs faster but uses more memory than geometrical approach.

A random comparison of the test run results with yahoo desktop search shows more precise results. In 3 search result comparison concept analysis yielded better results on two occasions than yahoo desktop search whereas on one occasion both yielded the same result. While performing the comparison study I found that all yahoo desktop search does is look for documents that contain any or all of the keyword from the query phrase, it also employs stemming if the match is not exact but doesn't look for ordering of keywords at all as compared to my approach which looks for ordered keyword sets that are near by.

Finally, the most important aspect of this concept analysis is to decide the association rules for keywords. After trying with several values I decided to use 5 as the maximum distance between two keywords to be considered nearby and the number of documents, for any keyword set to be important, greater than 20. After running the simplex generation algorithm and applying the graph theory or geometrical approach I found that concepts that involved the maximum sized keyword pairs were most precise in clustering the documents semantically. Another important point which is worth noting is that smaller the range of keyword pairs (i.e. from maximum sized keyword set to the minimum sized keyword set) more precise is the semantic space of the cluster formed by that connected component, a point explained in section 5.1 and 5.2 above.

# 7 List of Literature References

[1] – Clara Yu, John Cuadrado, Maciej Ceglowski, and J. Scott Payne. Patterns in Unstructured Data. Discovery, Aggregation, and Visualization. (http://www.hirank.com/semantic-indexing-project/lsi/cover_page.htm)
[2] – Lin T. Y., Chiang I-J, and Hu Xiaohua. Semantic Based Clustering of Web Documents.
[3] - Lin T. Y., Chiang I-J. A simplicial complex, a hypergraph structure in the latent semantic space of document clustering (International Journal of approximate reasoning Jan/2005).
[4] - Michael W. Berry, Susan T. Dumais, Gavin W. O'Brien. Using Linear Algebra for Intelligent Information Retrieval, Department of Computer Science, University of Tennessee, Knoxville, Dec. 1994.
[5] - S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by Latent semantic analysis, Journal of the American Society for Information Science, page 1, 1990.
[6] – Margaret H. Dunham. Association Rules (Basic Algorithms), Data Mining Introductory and Advanced Topics, page 169, 2004.
[7] – Rakesh Agarwal, Tomasz Imielinski, Arun Swami. Mining Association Rules between Sets of Items in Large Databases (Sigmod 1993).