2010

Computer-aided Semantic Signature Identification and Document Classification via Semantic Signatures

Uday Kiran Para
West Virginia University

Computer-aided Semantic Signature Identification and Document Classification via Semantic Signatures

Uday Kiran Para

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Electrical Engineering

Elaine M. Eschen, Ph.D., Chair
Alan V. Barnes, Ph.D.
Arun A. Ross, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2010

Keywords: Text Mining, Document Vectors, Semantic Signatures, Clustering, Keywords, Document Retrieval, Document Clustering, RCV1, Reuters

Abstract

Computer-aided Semantic Signature Identification and Document Classification via Semantic Signatures

Uday Kiran Para

In this era of textual data explosion on the World Wide Web, it may be very hard to find documents that are similar to the documents that are of interest to us. To overcome this problem we have developed a type of *semantic signature* that captures the semantics of target content (text). Semantic signatures from a text/document of interest are derived using the software package *semantic signature mining tool (SSMinT)*. This software package has been developed as a part of this thesis work in collaboration with Sri Ramya Peddada. These semantic signatures are used to search and retrieve documents with similar semantic patterns. Effects of different representations of semantic signatures on the document classification outcomes are illustrated. Retrieved document classification accuracies of Euclidean and Spherical K-means clustering algorithms are compared. A Chi-square test is presented to prove that the observed and expected numbers of documents retrieved (from a corpus) are not significantly different. From this Chi-square test it is proved that the semantic signature concept is capable of retrieving documents of interest with high probability. Our findings indicate that this concept has potential for use in commercial text/document searching applications.

Dedication

To My Family

To My Advisor Dr. Elaine Eschen and Dr. Alan Barnes

Acknowledgments

I would like to first express my truest and sincerest thanks to Dr. Elaine Eschen and Dr. Alan Barnes. Over the past three years working together, Dr. Eschen and Dr. Barnes provided me with the guidance and support necessary for completing the project and grow as a student, researcher and a wonderful programmer.

I would like to thank WV EPSCoR for providing the financial support through the Information Fusion Networks for Intelligence and Security (InfoNets) project led by Dr. Arun Ross (Associate Professor, Lane Department of Computer Science and Electrical Engineering, WVU). Dr. Ross has been a great inspiration during the whole time, which helped a lot.

The InfoNets meetings every semester have helped a lot by providing insights and exposure into a lot of research work that ultimately helped in coming up with new ideas for the research.

I would like to thank Dr. Robert Duval (Associate Professor, Political Sciences at West Virginia University) and his PhD student Kyle Christensen for providing the Discrete Algorithms Research Team (DART) with the huge corpus of web pages for research purposes. Also I would like to thank the National Institute of Standards and Technology for providing us with a large collection structured newswire articles knows as “Reuters Corpus, Volume 1” or RCV1.

Table of Contents

List of Figures

List of Tables

List of Symbols / Nomenclature

1. KDF – Keyword Descriptor File
2. SSD – Semantic Signature Descriptor
3. ARFF – Attribute Relation File Format
4. KT – Keyword Tool
5. LT – Learner Tool
6. DAT – Data Analysis Tool
7. CR – Cluster Representation
8. TOI – Topic of Interest
9. HTML – Hyper Text Markup Language
10. XML – Extensible Markup Language
11. RCV1 – Reuters Corpus Volume I
12. GSCI – Science and Technology
13. GHEA – General Health

Chapter 1: Introduction

Today's modern societies are built are dependent on information. Computers coupled with the Internet can make information available quickly to anyone looking for it. More importantly, computers can process that information more quickly than humans. They can also provide information enabling us to make better decisions that normally would have been made previously by a human being with imperfect knowledge built on their individual education and experience but not necessarily the best information. Computers can thus aid us in making the right decisions at the right moment using the best information available. This thesis deals with helping to refine the way computers decide which information is most pertinent and make, or help their human users make, decisions based upon it.

Our basic approach to mining text data aims at capturing the semantic structures in the text. Semantic structure depends on the correlations between keywords and locality of keyword groups. The traditional bag-of-words or keyword frequency approaches fall short of modeling these attributes. Our approach models not only keyword frequency, but also the distance between keywords and their relative ordering in the text. To this end, we derive high-dimensional vectors that store quantified relationships between keywords in a text document. In order to capture the locality of semantic structures, we generate many vectors per document. The content of these vectors is similar to the document vector (one per document) used by Zhang et al. in [1, 2]. However, unlike Zhang et al., we do not use these vectors directly to classify documents. Vectors generated from known content (learning) documents are used to develop *semantic signatures* that model the semantic structure of the target content. Multiple Semantic Signatures can be used to model various nuances of single target content. Semantic Signatures drawn from a library are then used to classify documents of unknown content. Our new

approach has proven to be a remarkably sensitive tool for differentiating semantic content in text data.

We all know that manually searching for documents that are similar to the documents you already have can be a very knowledge and time intensive task. This thesis helps to alleviate this problem by developing the concept of as *semantic signatures*. These *semantic signatures* help us find documents with similar semantic content by capturing it from the documents that are of interest to us. The information in these semantic signatures is stored in a compact format (in semantic signature Descriptor) that can be used in the present or future to retrieve similar documents from large collections of documents.

Semantic signatures from a document have information related to the interactions between keywords derived from this particular document. These keywords are manually selected by an analyst with the help of the graphical user interface (GUI) application named *Keyword Tool that* is developed as part of this thesis work. Now using these keywords, document vectors are generated from the training document(s) and then the noisy document vectors are filtered out with the help of GUI application named *Learner Tool*. Then the information from the selected document vectors after noise removal are stored in condensed form (in semantic signature Descriptor) and used in another application named *Data Analysis Tool* to identify documents that are semantically similar to the documents from which these vectors are generated.

Chapter 2: Text Processing, Clustering Techniques and Latent Semantic Analysis

Chapter 2 provides background information on concepts of text processing, clustering in the context of text processing and Latent Semantic Analysis (LSA). Section 2.1 describes some of the concepts used in text processing and how these concepts are applied in the context of semantic signature analysis (SSA). Section 2.2 gives an introduction to clustering techniques and distance measures used for finding the distance or similarity between vectors. Section 2.3 gives an introduction to Latent Semantic Analysis.

2.1 Text Processing Concepts

2.1.1 Stemming

Stemming is the process of reducing words to their stem, base or root form [3]. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. The process of stemming, often called *conflation*, is useful in search engines for query expansion or indexing and other natural language processing problems.

Stemming programs are commonly referred to as *stemming algorithms* or *stemmers*. There are various types of stemming algorithms which differ in respect to accuracy and performance and how certain stemming obstacles are overcome. Types of stemmers are listed in table 2.1. For the experiments in this thesis a standard suffix stripping algorithm called Porter Stemmer [4] is employed.

S.NO	Types of Stemmers
1	Brute Force Algorithms
2	The Production Technique
3	Suffix Stripping Algorithms
4	Additional Algorithm Criteria
5	Lemmatization Algorithms
6	Stochastic Algorithms
7	N-gram Analysis
8	Hybrid Approaches
9	Affix Stemmers
10	Matching Algorithms

Table 2.1: Types of stemmers

2.1.2 Stop Words

Stop words are the words that are deemed as noise and filtered out in text processing for certain types of applications. Hans P. Luhn, one of the pioneers in the field of Information Retrieval, is credited with coining the phrase and using the concept in his design [5]. In this thesis stop word removal is controlled by human input and is not automated. Stop words differ from language to language and application to application. Different sets of stop words can be used for different languages and applications. Not all text processing applications use stop words lists [6]. Some applications avoid using them to support the features like phrase searching. For some languages such as English, Dutch, German, Polish etc., stop word lists have already been developed and are readily available. There are many languages for which stop word lists have to be developed. The list of English stop words used in this thesis is given in Appendix A.

2.1.3 HTML Parsing

As all of us know, most data on the World Wide Web is stored in the HTML (Hyper Text Markup Language) format. We must parse the HTML and extract the textual data that is required to analyze textual data from web pages using text mining or information retrieval concepts. This process of extracting text from the HMTL web pages is called *HTML parsing*. HTML parsing is comparatively difficult compared to XML parsing, resulting from the fact that HTML progressed through different versions; sometimes the HTML can be malformed and sources of HTML cannot be controlled. Different versions of HTML have slightly different formatting rules. The things that make HTML parsing difficult are listed below.

- HTML doesn't require end tags.
- HTML attribute values are not necessarily fully quoted with either single or double quotes.
- It is not a necessity for HTML tags to be properly nested.
- HTML tag names are not case sensitive.
- HTML allows duplicate attributes.
- Empty attributes are allowed in HTML.

HTML parsing is used in one of the experiments to parse the web pages from a huge corpus of 80,000 web pages. This corpus is collected from domestic extremist websites by Dr. Robert Duval and Kyle Christensen from social sciences department at West Virginia University. HTML parser programmed by Chudnovsky [7] was used in this thesis for conducting experiments.

2.1.4 XML Parsing

XML (Extensible Markup Language) is similar to HTML parsing, but easier to parse. XML parsing is easier because XML doesn't have the irregular structure compared to HTML. The following rules make XML parsing less complicated as opposed to the rules of HTML parsing listed above.

- XML require end tags.

- XML attribute values are fully quoted with either single or double quotes.
- It is necessary for XTML tags to be properly nested.
- XML tag names are case sensitive.
- XML does not allow duplicate attributes.
- Empty attributes are not allowed in XML.

XML parsing is used in this thesis to parse documents from Reuters Corpus Volume I (RCV1) [8]. This is a huge corpus of over 800,000 manually categorized newswire articles. XML parsing of these documents is done with the help of the XML parser class from the .NET System.XML namespace.

2.2 K-means Clustering

The K-means clustering algorithm, also known as Lloyd's Algorithm, is used to cluster/group a given set of vectors/observations into K clusters/groups. The input to this algorithm is the number of clusters K and the vectors/observations.

In this clustering algorithm, initially K vectors are chosen at random from the given vectors. These vectors act as the centroids for the starting iteration. During the first iteration the vectors are grouped with the centroid they are close to (depending on the distance measure used). When the *Euclidean distance measure* is used, a vector is said to be closest to a particular centroid if distance between them in Euclidean space is less than that of the distance between the vector and other centroids. If the *cosine similarity measure* is used, a vector is said to be closest to a particular centroid if the value of the cosine distance between them is greater than that of the cosine distance between the vector and other centroids. When Euclidean distance measure is used K-means clustering algorithm is known as Euclidean K-means, similarly it is known as Spherical K-means when cosine similarity measure is used [9].

After the first iteration the centroid of a group will be the mean of the group's vectors. After computing the new centroids the process is repeated. We can stop the iterations when the clusters seem to be stable or whenever the limit for the number of iterations is reached. It is necessary to number of iterations because sometimes the cluster may not be stable even after a large number of iterations and may loop forever. This is basic K-means clustering. There are some drawbacks

to the K-means clustering in that it does not yield the same result each time the algorithm is run on the same data. The numbers of clusters are not automatically determined and have to be provided to it in advance. It can be very sensitive to initial selection of the seed for cluster centroids [10].

The algorithm for K-means clustering is as follows [11]:

1. *Distribute all the vectors among the k bins.*
2. *Compute the mean vector for each bin.*
3. *Compare the vector of each vector to the bin means and note the mean vector that is most similar.*
4. *Move each vector to its most similar bin.*
5. *If no vector has been moved to a new bin, then stop; else go to step 2.*

Figure 2.1 K-means clustering algorithm.

2.2.1 Euclidean Distance Measure

The Euclidean distance measure/metric is used to find the distance between two points in Euclidean space. This distance is same as the one measured with a ruler.

If $X = (x_1, x_2, x_3 \ldots x_n)$ and $Y = (y_1, y_2, y_3 \ldots y_n)$ are two points in Euclidean n-dimensional space, then the Euclidean distance between X and Y is given by:

$$d(X,Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2 + \ldots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

In this thesis the Euclidean distance measure is used to compute the distance between two semantic feature vectors or document vectors. This gives a measure of how far the vectors are from each other in n-dimensional Euclidean space.

2.2.2 Cosine Similarity Measure

The cosine similarity gives the measure of similarity between two vectors in n-dimensions by finding the cosine of the angle between them. The cosine similarity measure for a pair of vectors A and B is given by:

$$CS(X,Y) = \cos(\theta) = \frac{A.B}{\|A\|.\|B\|}$$

where θ is the angle between vectors A and B.

Let us assume that we have three n-dimensional vectors X, Y and Z. Vector X is said to be more similar to Y than Z if the cosine similarity measure between X and Y is greater than the cosine similarity measure between X and Z, i.e. $CS(X,Y) > CS(X,Z)$.

In this thesis the cosine similarity measure is used to compute the similarity between two semantic feature vectors or document vectors. This gives a measure of how similar the vectors are to each other in n-dimensional space.

2.3 Latent Semantic Analysis

Latent Semantic Analysis is a technique used in text data mining and information retrieval [12, 13] for retrieving or categorizing documents based on their semantic content. Latent Semantic Analysis decomposes a matrix called a term-document matrix into three matrices U, Σ and V^T using singular value decomposition [14]. The columns (semantic feature vectors) of the V^T matrix represent individual documents. So these semantic feature vectors from V^T matrix are used in clustering the documents in this thesis.

2.3.1 Term- Document Matrix

A *term-document* matrix is a two-dimensional matrix/table containing the terms in one of the dimensions and documents in the other. The term-document matrix is also known as the *occurrence* matrix. In this thesis, terms are taken as rows and documents are taken as columns, when the singular value decomposition is performed on this matrix. The ordering of terms and documents can be the other way around. If the document-term matrix is decomposed using

singular value decomposition, matrix U corresponds to documents and matrix V corresponds to terms and vice-versa if the term-document matrix is used.

Example:

D1 = " New seeds may lead to strain of superbugs "

D2 = " New seeds may not lead to strain of superbugs"

	D1	**D2**
New	1	1
Seeds	1	1
May	1	1
Lead	1	1
To	1	1
Strain	1	1
of	1	1
Superbugs	1	1
not	0	1

Table 2.2: Document-Term Matrix

These term frequencies can be weighted using the term frequency – inverse document frequency (tf-idf) technique.

2.3.2 The Concept Space

LSA decomposes the term-document matrix (X) into three matrices U, Σ and V^T.

$$\boldsymbol{X = U\Sigma V^{T}}$$

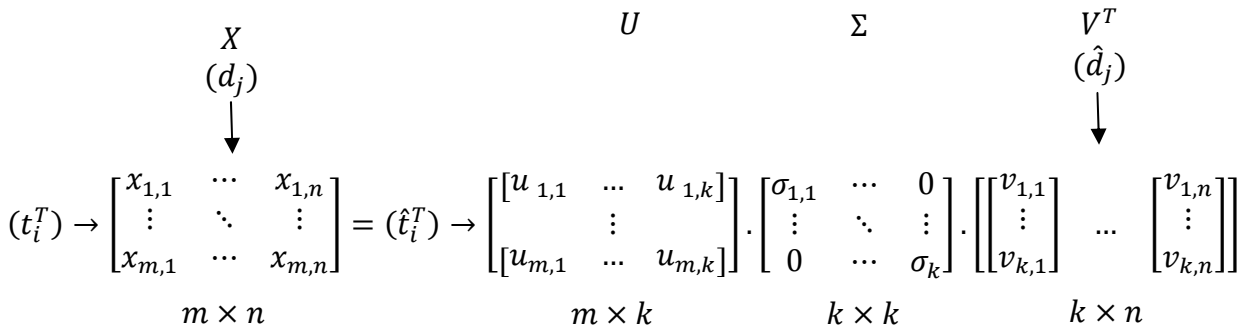

Figure 2.2: Singular Value Decomposition of X into three matrices namely U, Σ and V^T.

In figure 2.2, $\boldsymbol{\sigma_1, \sigma_2 \ldots \sigma_k}$ are called the singular values, $\boldsymbol{u_1, u_2 \ldots u_k}$ are called the left singular vectors and $\boldsymbol{v_1, v_2 \ldots v_k}$ are called the right singular vectors.

When k largest singular values and their corresponding singular vectors from U and V matrices are selected this gives us the *rank-k approximation* of the matrix X.

$$\boldsymbol{X_k = U_k \Sigma_k V_k^T}$$

After the rank-k approximation the term vectors $\hat{t}_j$ (row vectors) from the U matrix and the semantic feature vectors $\hat{d}_j$ (column vectors) from the V matrix give us a relation between the terms and concepts, documents and concepts, respectively. Here the vector $\hat{d}_j$ gives the relation between document *j* and each concept. Similarly $\hat{t}_j$ gives us the relation between term j and each of the concepts. These semantic feature vectors $\hat{d}_j$ are used for clustering documents in the concept space. In this thesis when clustering the semantic feature vectors $\hat{d}_j$ using K-means clustering technique, cosine similarity measure is used as the similarity metric.

2.4 Keywords

Keywords are words that occur rather frequently in the text of a document(s). In this thesis keywords and the relationships between them are used to represent a particular topic or document using semantic signatures.

Chapter 3: Introduction to Semantic Signatures and Tools

Chapter 3 provides introduction to the notion of semantic signatures and the theoretical concepts and tools developed as a part of this thesis. Section 3.1 gives an introduction to the Keyword Tool. Section 3.2 gives an introduction to the Learner Tool. Section 3.3 gives an introduction to the Data Analysis Tool. Sections 3.4 through 3.10 describe the techniques developed and used in this thesis.

The three tools are together called named as semantic signature mining tool (SSMinT) package. This software package was developed by me in conjunction with Sri Ramya Peddada [15].

3.1 Introduction to the Keyword Tool

The Keyword Tool has been developed as a part of this thesis to aid us in the keyword selection process. Keywords that are generated as a result of keyword selection process are used in the generation of semantic signatures. Here a semantic signature is a set of the document vector space that represents a specific concept. Document vectors are explained in section 3.7.

The goal of the keyword selection process is to identify the keywords that are semantically correlated and frequently used in the text that contains the targeted concept/topic. For example, keywords might be designed to capture violent intent, incitation, precipitation of fear, protectiveness (for a group), paranoia, etc. figure 3.1 shows the inputs, functions and outputs of the Keyword Tool. This tool was developed using Microsoft Visual Studio IDE and C#.

3.2 Introduction to the Learner Tool

The Learner Tool is used in this thesis to help us identify good semantic signatures from a document. The semantic signatures are then saved in a file (in xml format) with .SSD extension. The SSD files are then grouped into a library called *semantic signature descriptor library*. This

tool can be used to develop a semantic signature descriptor library for particular textual topic(s). This semantic signature descriptor library is used in the Data Analysis Tool for identifying documents that are similar to the topics captured by SSDs. Figure 3.2 shows the inputs, functions and outputs of the Learner Tool.

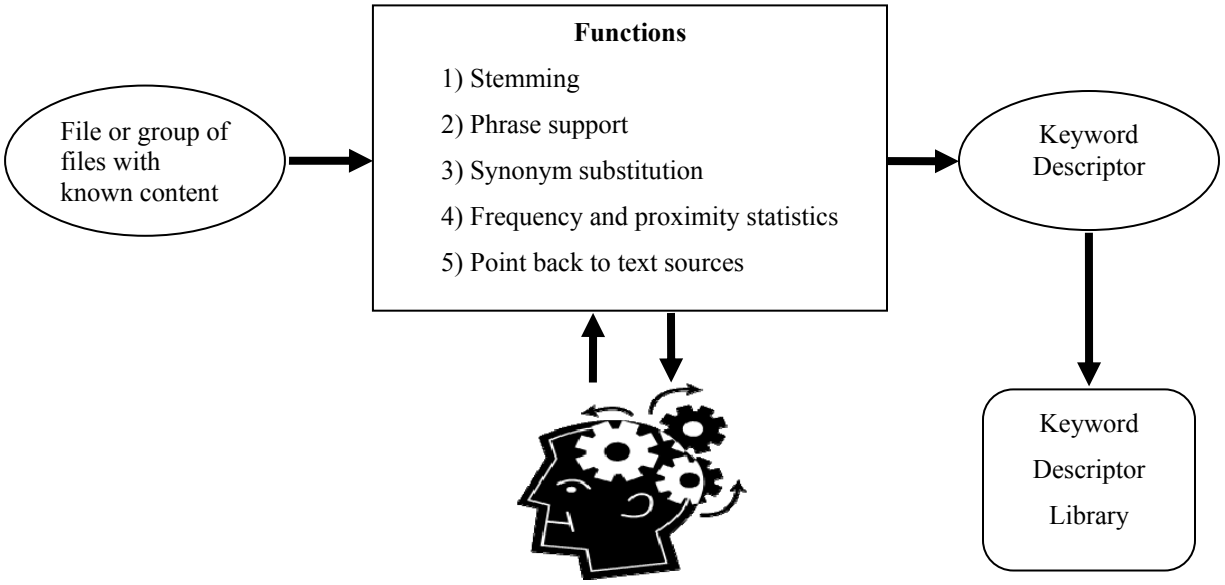

Figure 3.1: Human interaction with the Keyword Tool. Inputs, Outputs and Functions of the Keyword Tool.

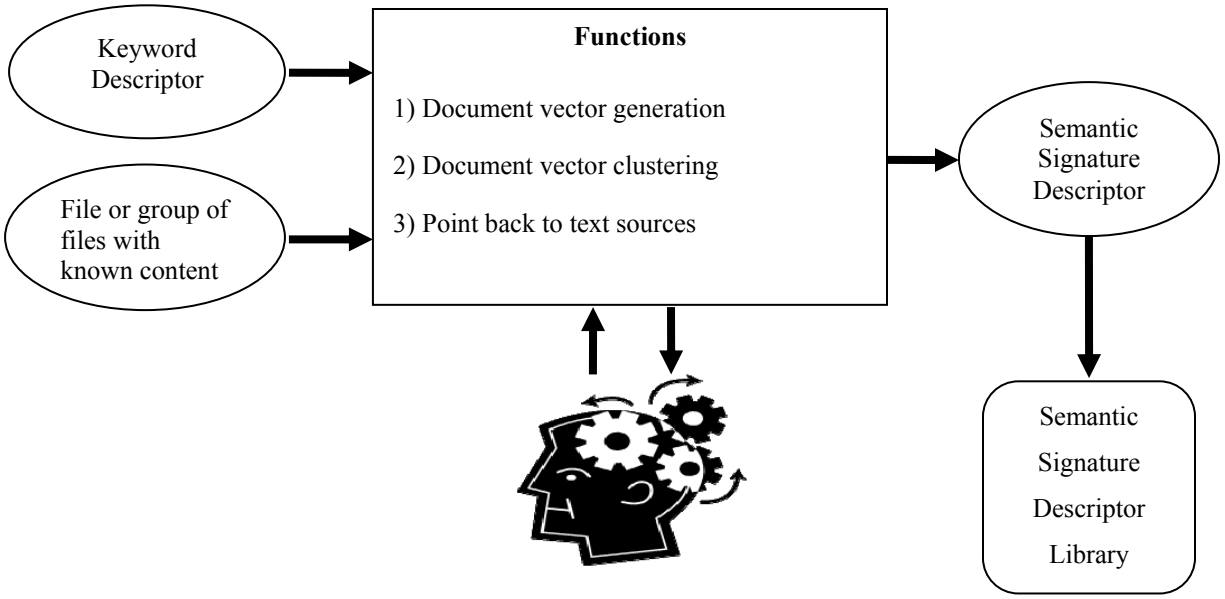

Figure 3.2: Human interaction with the Learner Tool. Inputs, Outputs and Functions of the Learner Tool.

3.3 Introduction to the Data Analysis Tool

The Data Analysis Tool is developed as a part of this thesis to aid us in the retrieval and classification of documents from a text corpus using the SSDs provided by the user. The output of the Data Analysis Tool is a matrix called *document analysis matrix*. This tool can be used to save the output in WEKA [16] file format known as the attribute relation file format. The Data Analysis Tool can also be used to cluster the rows of the document analysis matrix using K-means clustering. This tool was developed using Microsoft Visual Studio IDE and C#.

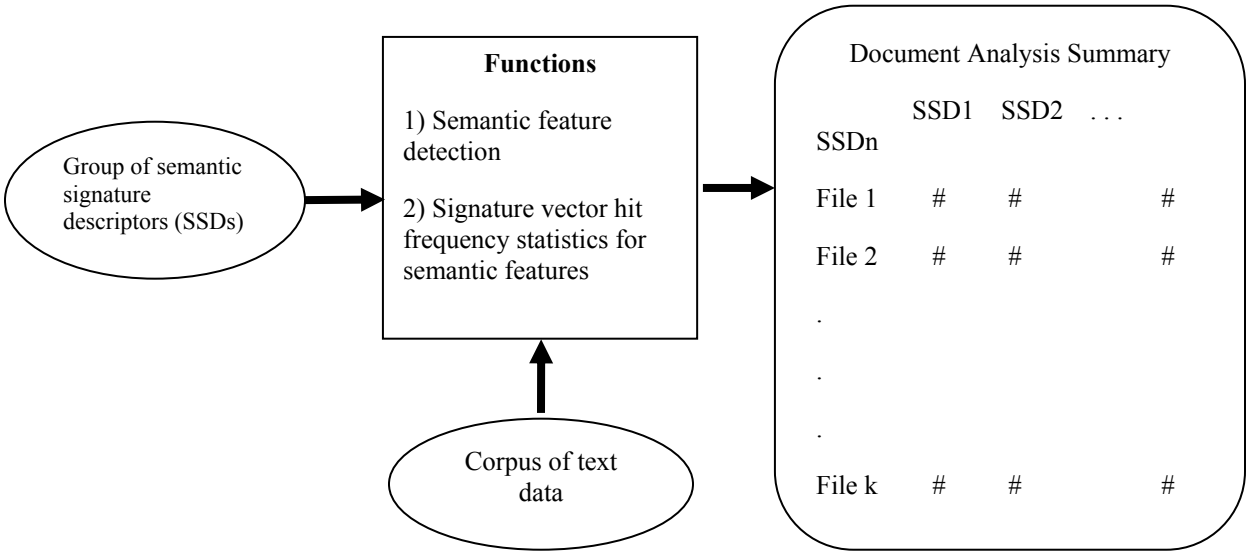

Figure 3.3: Human interaction with the Data Analysis Tool, Inputs, Outputs and Functions

3.4 Processing Document Analysis Matrix

Even though the semantic feature vectors (rows of document analysis matrix) can be clusterd in the Data Analysis Tool, it is limited to Euclidean and Spherical K-means clustering. For example, some users may want to cluster semantic feature vectors using other clustering techniques. We recommend using WEKA for clustering semantic feature vectors, visualizing the clustered semantic feature vectors and saving the clustering results.

3.5 Semantic Signature

The semantic signature is a way of representing a piece of text in a document by using some keywords and the distances between these keywords in this particular piece of text. Semantic signatures are analogous to meaningful signatures.

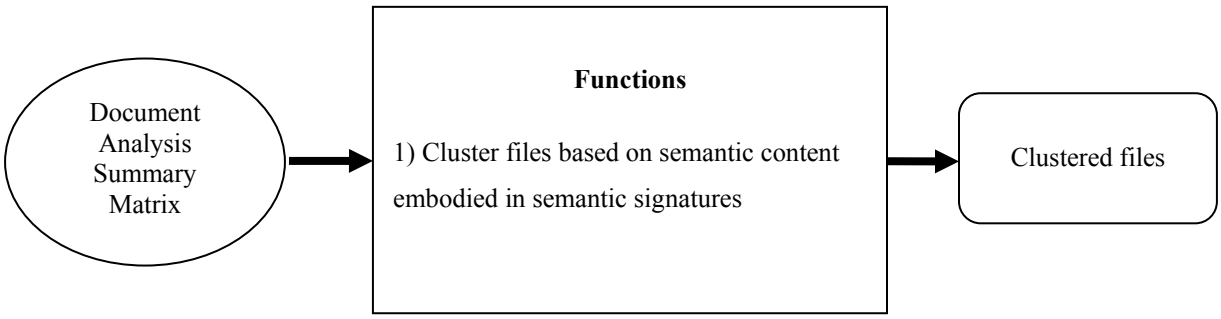

Figure 3.4: Overview of document analysis matrix processing.

Here, we are trying to apply semantic signatures in the field of information retrieval [12, 13] [17] and document clustering. All of us know that in information retrieval applications, we have a table known as term-document matrix. In this thesis we are replacing the terms with features known as semantic signatures. Semantic signature descriptors are files that contain semantic signatures derived from a document containing known content or content of interest to us. Semantic signature descriptors also contain other parameters essential for the process of detecting semantic signatures in other text files.

3.6 Window Weight Function

Weight function is a function that gives the weight between a given set of keywords with a distance (d) between them. Here distance (d) gives the number of words between the given set of keywords excluding the stop words. Throughout the document the term '*weights'* refers to the weights obtained from the following weight function:

$$W(d) = \sqrt{\frac{a^2}{a^2 + d^2}}$$

where ‘a’ is a user defined constant. By default its value is 5.0.

The following table has the values of window weight function at various values of ‘a’ and ‘d’

d	***W(d) with a* = 1.25**	***W(d) with a* = 2.5**	***W(d) with a* = 5**	***W(d) with a* = 10**
0	1.00	1.00	1.00	1.00
1	0.78	0.93	0.98	0.99
2	0.53	0.78	0.93	0.98
3	0.38	0.64	0.86	0.96
4	0.30	0.53	0.78	0.93
5	0.24	0.45	0.71	0.89
6	0.20	0.38	0.64	0.86
7	0.17	0.34	0.58	0.82
8	0.15	0.30	0.53	0.78
9	0.14	0.27	0.48	0.74
10	0.12	0.24	0.45	0.71
11	0.11	0.22	0.41	0.67
12	0.10	0.20	0.38	0.64
13	0.09	0.19	0.36	0.61
14	0.09	0.17	0.34	0.58
15	0.08	0.16	0.32	0.55
16	0.08	0.15	0.30	0.53
17	0.07	0.14	0.28	0.51
18	0.07	0.14	0.27	0.48
19	0.06	0.13	0.25	0.46
20	0.06	0.12	0.24	0.45

Table 3.1: Window weight function values for varying ‘a’ and ‘d’

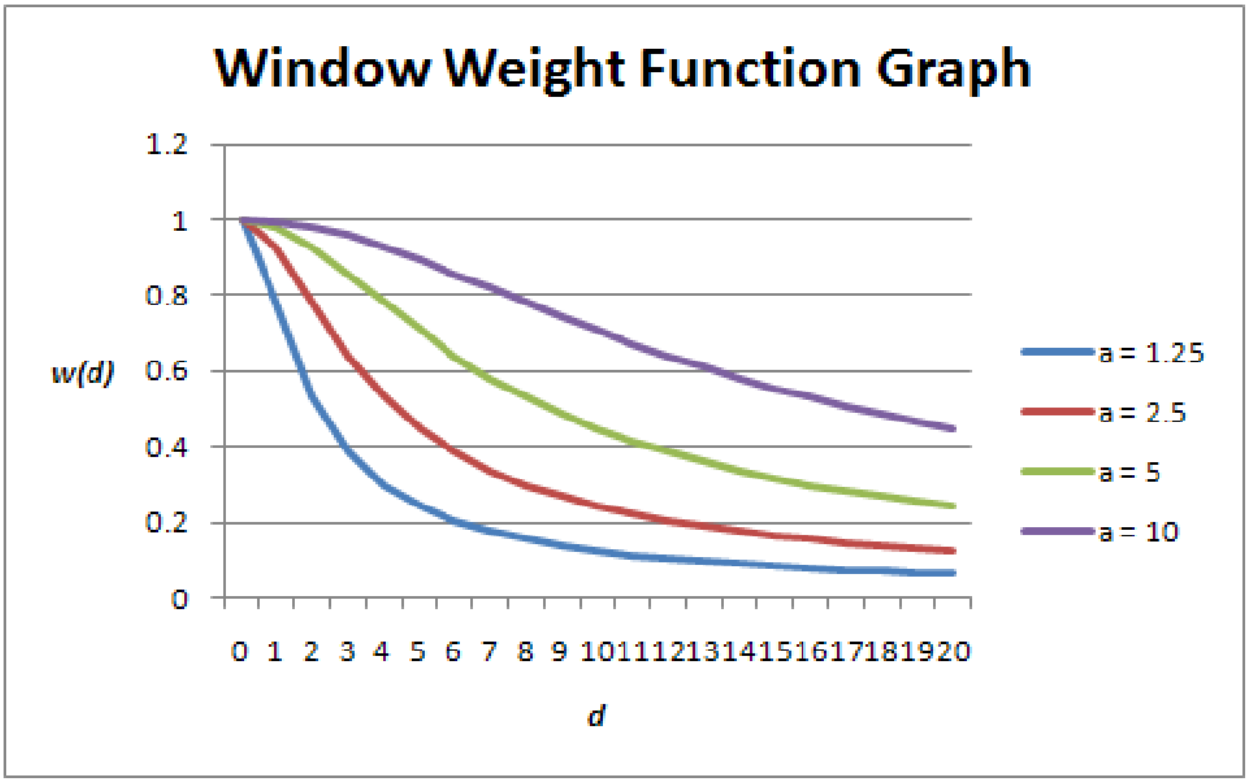

Figure 3.5: Graph of window weight function for various values of 'a'.

3.7 Document Vector (DV)

Document vectors represent metrics for correlated keywords in the given text data. We employ a keyword distance metric, such as w (d) = sqrt (a2 / (a^2 + d^2)), where 'a' is a constant and 'd' is the count of words between two keywords under consideration. To compute the *weight* between two keywords (both keywords can be identical), the distance metrics between occurrences of all the keywords are accumulated and normalized by frequency. All weights are computed forward in the file, so as to include each distance only once. For example, the weight for "KW0 followed by KW1" in text where the keywords appeared in the order:

$$\textbf{\textit{KW0 …← } d_1 \textbf{\textit{→… KW1 …← } } d_2 \textbf{\textit{→… KW0 …← } } d_3 \textbf{\textit{→… KW1}}$$

Figure 3.6: Keywords apperances in a piece of text.

Would be [w(d_1) + w(d_1+d_2+d_3) + w(d_3)]/3, whereas the weight for "KW1 followed by KW0" would be w(d_2). In figure 3.6, d_1, d_2 and d_3 are number of words between keyword pairs (KW0,

KW1), (KW1, KW0) and (KW0, KW1) respectively. Taking w[i, j] to be the weight for "KW_i followed by KW_j", these weights produce the vector (w[0,0], w[0,1], w[1,0], w[1,1]). This is a document vector of four dimensions. We generate document vector for a piece of text in a document beginning at a keyword and ending at a word, which is at a distance given by the user. The piece of text referred to earlier is called a *window* and the number of words in this window is called as *window length*. The concept of document vectors is flexible and can incorporate other statistical measures and quantified attributes, which would increase the dimension of the vector.

	KW0	***KW1***
KW0	W (d1+d2)	[W (d1)+W (d1+d2+d3)+W (d3)]/3
KW1	W (d2)	W (d2+d3)

Table 3.2: Sample Keyword weights table

The above table contains normalized weights between all the keyword combinations in the window from figure 3.6.

3.8 Weights Calculation in the Keyword Tool

In the Keyword Tool, weights are calculated in both forward and backward directions to capture how far other words are in the document from a given word on the average. This way we will be able to select keywords that form a meaningful sentence and capture the semantics.

3.9 Techniques Used in the Learner Tool

3.9.1 Weights Calculation in the Learner Tool

In Learner Tool the weights in each dimension of the document vectors are calculated in only one direction (forward) as opposed to both directions in the Keyword Tool. An example of how to calculate document vector is shown in this section.

Shuttle Atlantis rolled to launch pad.

The space shuttle Atlantis, fitted with new rockets, was shifted to its seaside launch pad on Tuesday, bringing astronaut Shannon Lucid one step closer to home. **Atlantis is being primed for a Sept. 12 launch** to pick up Lucid from the Russian Mir space station, where she has been working since March. The 53-year-old mother of three was due home earlier this month, but NASA delayed **Atlantis' flight by six weeks to replace two suspect rocket boosters. The shuttle** made the slow, 3.4 mile (5.5 km) journey from its assembly building to the **launch pad at the Kennedy Space Centre, riding atop a giant caterpillar-tracked transporter. Shuttle** managers ordered new solid rocket boosters for **Atlantis after dangerously hot gas singed crucial seals in the boosters used to launch** sistership Columbia in June.

Figure 3.7 Sample text showing keywords highlighted in colors and windows in bold.

The words in bold and color in figure 3.7 are the keywords selected using the Keyword Tool. Bold pieces of text in figure 3.7 are called as *Windows* in this thesis.

Index	Keyword	Position
0	2	0
1	0	1
2	1	4
3	2	8
4	0	9
5	1	19
6	0	32
7	1	40
8	0	74
9	2	86
10	1	103
11	2	117
12	0	125
13	1	138

Table 3.3: Base Table for the text in figure 3.7

Using text from figure 3.7 and keywords from the Keyword Tool a table called as *Base Table* is generated, which contains the positional information of keywords in text from figure 3.7. In table

3.3 keywords 0, 1 and 2 refer to 'Atlantis', 'Launch' and 'Shuttle' respectively. In table 3.3 position column refers to position of the words in the text from figure 3.7.

Now a table called as *Window Index Table* is generated from base table (table 3.3).

Window Starting Position in the Base Table	Window Ending Position in the Base Table
0	5
6	7
8	9
10	11
12	13

Table 3.4: Window Index Table

Using information in base table and window index table a matrix is generated. This matrix contains the averaged weighted distances between all keyword possible keyword occurrences in first window of figure 3.7. Weight function from section 3.6 is used for weighting the distances between keywords.

	Atlantis	Launch	Shuttle
Atlantis	0.53	0.53	0.58
Launch	0.71	0.32	0.78
Shuttle	0.82	0.48	0.53

Table 3.5: A matrix showing the weights calculated using keyword-to-keyword distances

This matrix is now represented in a vector form. This vector is the document vector that is explained in section 3.7. All the rows from the matrix in table 3.5 are appended one after the other to generate the document vector. The document vector generated from the matrix (in table 3.5) is shown below.

0.53	0.53	0.58	0.71	0.32	0.78	0.82	0.48	0.53

Table 3.6: Document vector

3.9.2 Document Vector Clustering Metrics

In our applications it is necessary to quantify document vector similarity. We accomplish this via clustering the document vectors in multidimensional space. We generate a cluster sphere C of signatures that are "close" to the seed signature S* (derived from text with the desired target content). Not all components of the document vector are created equal. Each component will

have some characteristic distribution of values for the input data space; some will have numerically broad distributions, while others will have narrow distributions. Document vector components may also be correlated. We use coordinate transformations to obtain an uncorrelated sphere about center S*. In some dimensions the distribution is likely have strong correlations and poorly behaved (ill-conditioned) directions. Iterative redefinition of the cluster around S* using the improved metric from the previous iteration will result in a more precise similarity measure. This process illuminates the significance of signature components, and thereby, can contribute to signature design.

3.9.3 Cluster Representations

The output clusters from the Learner Tool are being represented in three different ways. They are:

- Cluster Representation 1 (CR1)
- Cluster Representation 2 (CR2)
- Cluster Representation 3 (CR3)

3.9.3.1 Cluster Representation 1

In this representation a cluster is represented by its centroid and the distance between the centroid and the farthest vector in this particular cluster [18]. When semantic signatures are saved in an SSD using this cluster representation, the resulting SSD is designated as a CR1 SSD.

3.9.3.2 Cluster Representation 2

In this representation a cluster is represented by its centroid and the cosine distance between the centroid and the vector which is farthest away from the centroid in terms of angle in this particular cluster. When semantic signatures are saved in an SSD using this cluster representation, the resulting SSD is designated as a CR2 SSD.

3.9.3.3 Cluster Representation 3

In this representation a cluster is represented by all the vectors in the cluster. When semantic signatures are saved in an SSD using this cluster representation, the resulting SSD is designated as a CR3 SSD.

3.10 Techniques used in the Data Analysis Tool

3.10.1 Classifying Document Vectors in the Data Analysis Tool

In this thesis the phrase "document vectors" refer to the vectors generated from a document using the keywords and semantic signatures information from SSDs. Classification of document vectors differs for different cluster representations used in the Learner Tool.

When CR1 is used for representing a cluster of document vectors in the Learner Tool, a document vector is classified as belonging to the cluster (semantic signatures) represented by an SSD if the Euclidean distance between it and the centroid of the cluster is less than the radius of the cluster (assuming that the cluster is spherical). When a document vector is classified as belonging to a cluster represented by an SSD it is called as a *hit*. The hit count for an SSD is incremented for each document vector classified as belonging to the cluster represented by this SSD.

When CR2 is used for representing a cluster in the Learner Tool, a document vector is classified as belonging to the cluster (semantic signatures) represented by an SSD if the cosine distance between it and the centroid of the cluster is more than the cosine distance value from this SSD [19]. The definition of a hit is the same in the case of CR2 as in the case of CR1 described above.

When CR3 is used for representing a cluster in the Learner Tool, we do not have a hit or miss counter for the document vectors that fall or do not fall in the cluster as we did in cluster representations CR1 and CR2. In CR3 we use a continuous scoring mechanism. For each document vector found, the cosine distance between it and all the semantic signatures from the SSD are calculated. The maximum of the calculated cosine distances is used as a similarity score for that vector. The similarity scores for all the document vectors found in the analysis are averaged to get a similarity score for the document which takes the place of the hit frequency found in CR1 and CR2.

Chapter 4: Tools in Depth

Chapter 4 provides information about the tools developed as a part of this thesis.

In order to conduct the experiments in this thesis three tools have been developed. These tools as a whole are named as semantic signature mining tool (SSMinT).The names of these three tools are:

1. Keyword Tool
2. Learner Tool
3. Data Analysis Tool

Section 4.1, 4.2 and 4.3 describe concepts and functionalities of the Keyword Tool, Learner Tool and Data Analysis Tool respectively in detail.

4.1 Keyword Tool

As the name suggests the Keyword Tool is used for selecting keywords from the given textual data that better represents the given textual data. The selected textual data should be of known content. This tool provides word frequencies, proximity statistics and the ability to "pointback" to the text with the selected keywords highlighted in the given text and generates *keyword descriptor files* containing the analyst's data preprocessing choices, parameter choices and the selected keywords.

Our method is to start with preprocessing a given text to remove *stop words* (e.g., articles and prepositions, who, what, why, when, where, etc.) and optionally perform *word stemming* (e.g. convert hating and hated to hate). We then produce a frequency ordered list of words from the text source. The analyst selects a word KW0 of his choice from this list. In the second phase, we provide the analyst with a list of words that appear most frequently and closest to KW0 in the text. From this second list the analyst may choose one or more keywords. The process may be -

continued by selecting one keyword KW1 at this point and running a third phase in which we provide the analyst with a list of words that are highly correlated with KW1. Keyword design tools will also be able to utilize word class dictionaries, so that words with similar meaning are replaced by intensity ranked *meta-words* in the text and phrase grouping so that a phrase such as "black market" is treated as one word.

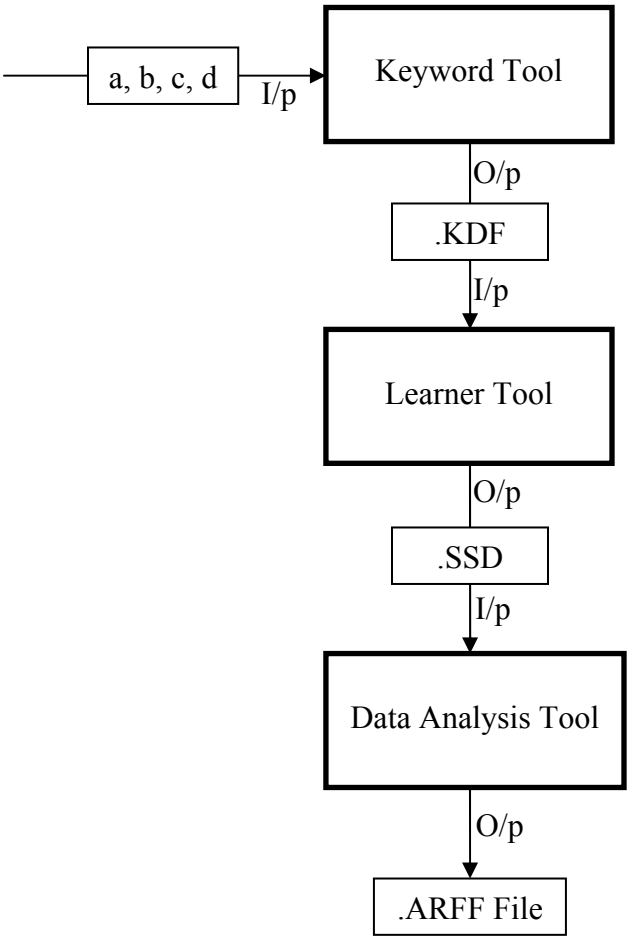

Figure 4.1: Data flow between the three tools.

a) File from which semantic signature has to be extracted
b) Window size (default value is 20)
c) Window function constant (default value is 5.0)
d) Stemming (default value is false)

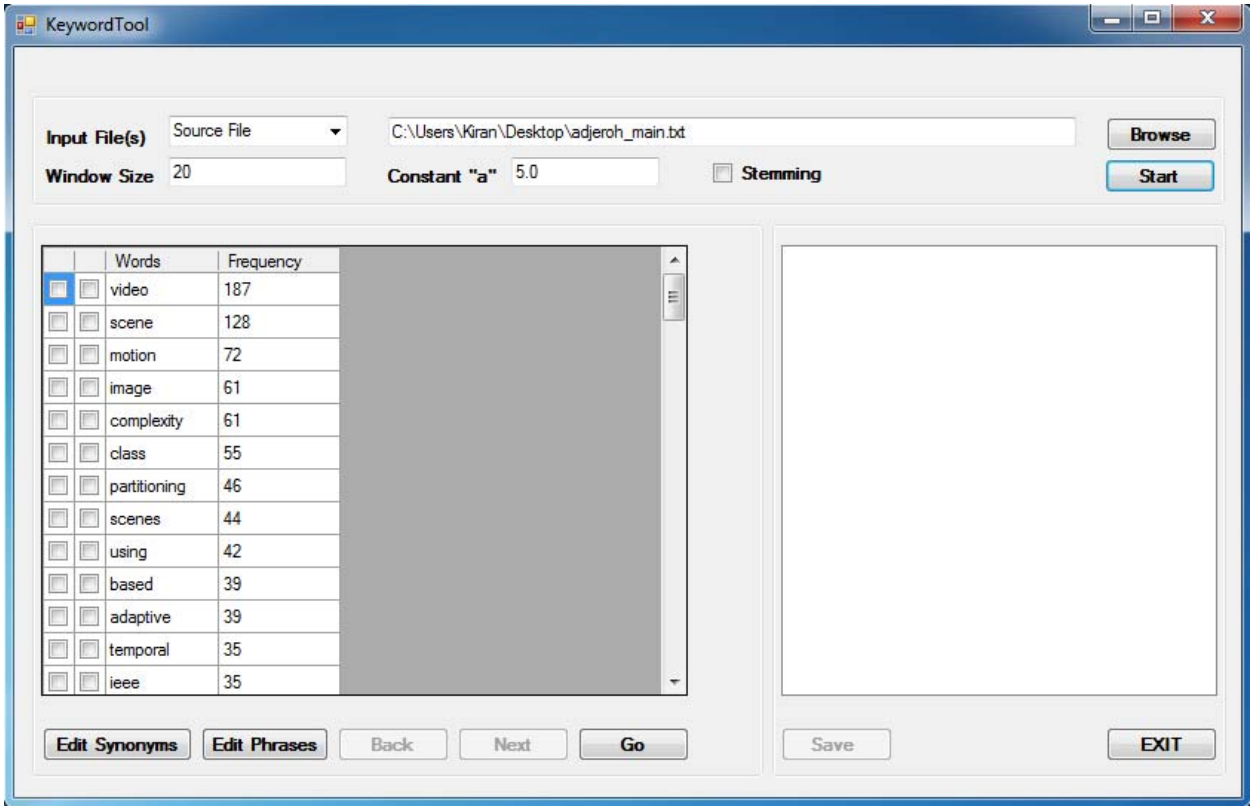

Figure 4.2: Keyword Tool displaying a list of words ordered by frequency of apperance in the given text document.

Phrases are sometimes used as keywords because single word keywords may not represent a topic well enough. For example take the phrase “black market”. When single word keywords are used some documents may have both the words black and market or just black or market. In some scenarios both the words might be present in the document but it may not at all be talking about a “black market”. The author may be talking about “black gram” and markets or “black umbrellas” and markets instead. In this case, document vectors will have the weights for black and market and they may be classified as a document belonging to the topic of black markets, even though the document might be talking about a different topic entirely. So, to avoid these kinds of situations, it is recommended to use phrases instead of single word keywords when we want to target a specific topic.

Selecting good keyword sets is the user’s hardest job. In the process of keyword selection, keywords are selected such that they form a structure related to a sentence. For keywords to form such a structure they should be from parts of speech like verb, adjective, noun etc. We should not

select all the keywords from a single part of speech. They should be from various parts of speech such that together they form a structure similar to a sentence.

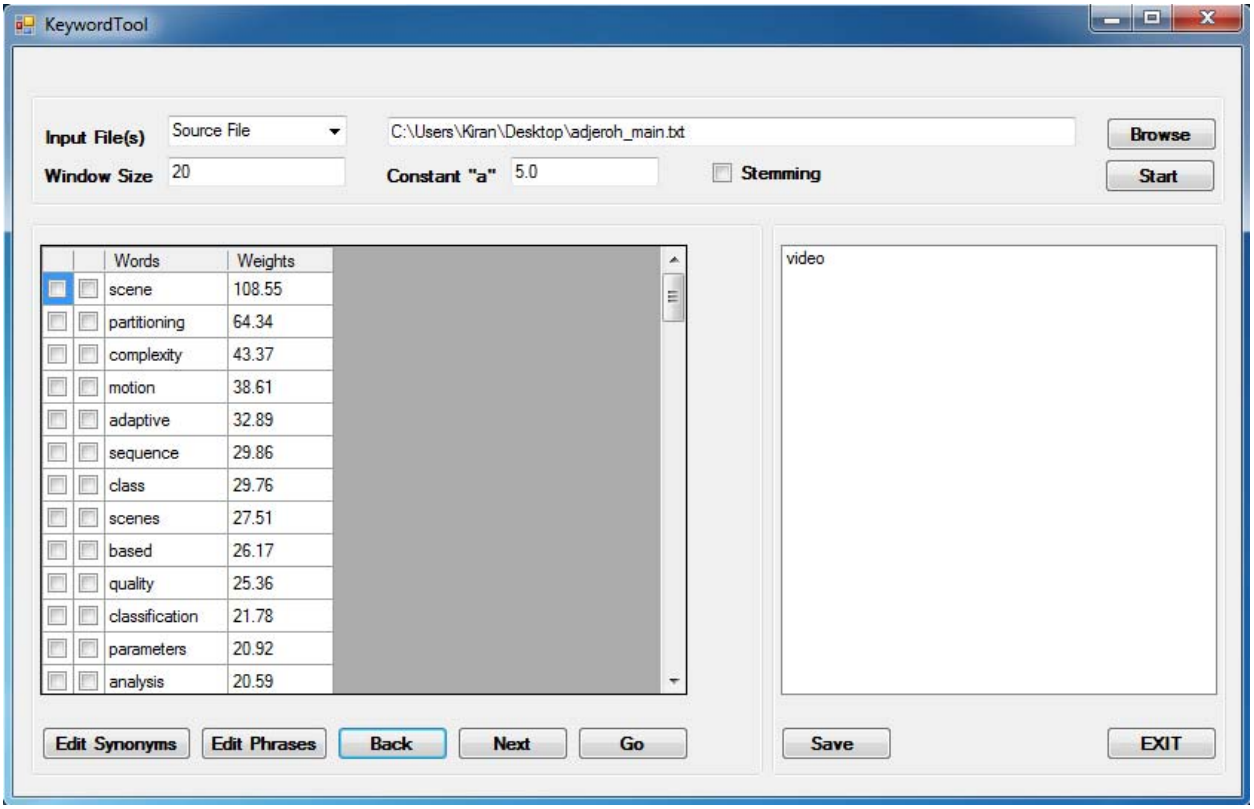

Figure 4.3: Keyword Tool displaying a list of words and their weights relative to the keyword video from the given text document.

4.1.1 Keyword Tool Input Data

The input for the Keyword Tool will be a file containing plain ASCII text from which meaningful keywords are to be extracted. The input file can be in any one of the following formats:

- .txt
- .html
- .htm
- Reuters corpus (RCV1) XML format

By default the window size is 20 and constant of the window function ‘a’ is 5. Window size by default is taken as 20 because on an average a sentence is made up of twenty words.

Additionally there is an option for stemming the words from the given file(s). The words are stemmed after the stop words are removed. Porter stemmer algorithm is used for stemming the words from the given file. Also stemming is optional. By default the words are not stemmed.

4.1.2 User Interaction with the Keyword Tool and Underlying Processes

This section describes about the user interaction process with the Keyword Tool. Keyword Tool is used for assisting the user with the keyword selection process.

4.1.2.1 ‘Start’ Button Click Event

After the user has given the required input he/she can start the keyword selection process by clicking the *Start* button as demonstrated in figure 4.2. When the user clicks the *Start* button, the tool checks the input provided by the user. The path of the file displayed in the input file text box (figure 4.2) is verified to see whether it exists or not. If it does not exist, a message is displayed asking the user to enter correct input. If the file exists, the preprocessing begins. The first step of the preprocessing stage is to determine the type of the file. If it is an html file, html tags are removed and only plain text is considered for further processing. If it is a Reuters xml file, only the headline and body of the document is considered for further processing.

Phrase replacement is performed if the user inputs any phrases using the *Edit Phrases* dialog box. The phrases in the document identified by the user are replaced with temporary phrase markers ($phr0, $phr1… $phrN). The entire document is then split into words, which are stored in a *word list*. Now the temporary phrase markers are replaced with the appropriate phrases.

Stemming is done if the check user checks the check box in the user interface before clicking the *Start* button. Otherwise it skips the stemming part of the code. Stemming algorithm in the tool is implemented with the help of porter stemmer algorithm.

Synonyms substitution is performed if synonyms are entered by the user through using the *Add Synonyms* dialog box. During synonyms substitution process, synonyms are replaced with their respective keywords defined by the user. This completes the preprocessing stage.

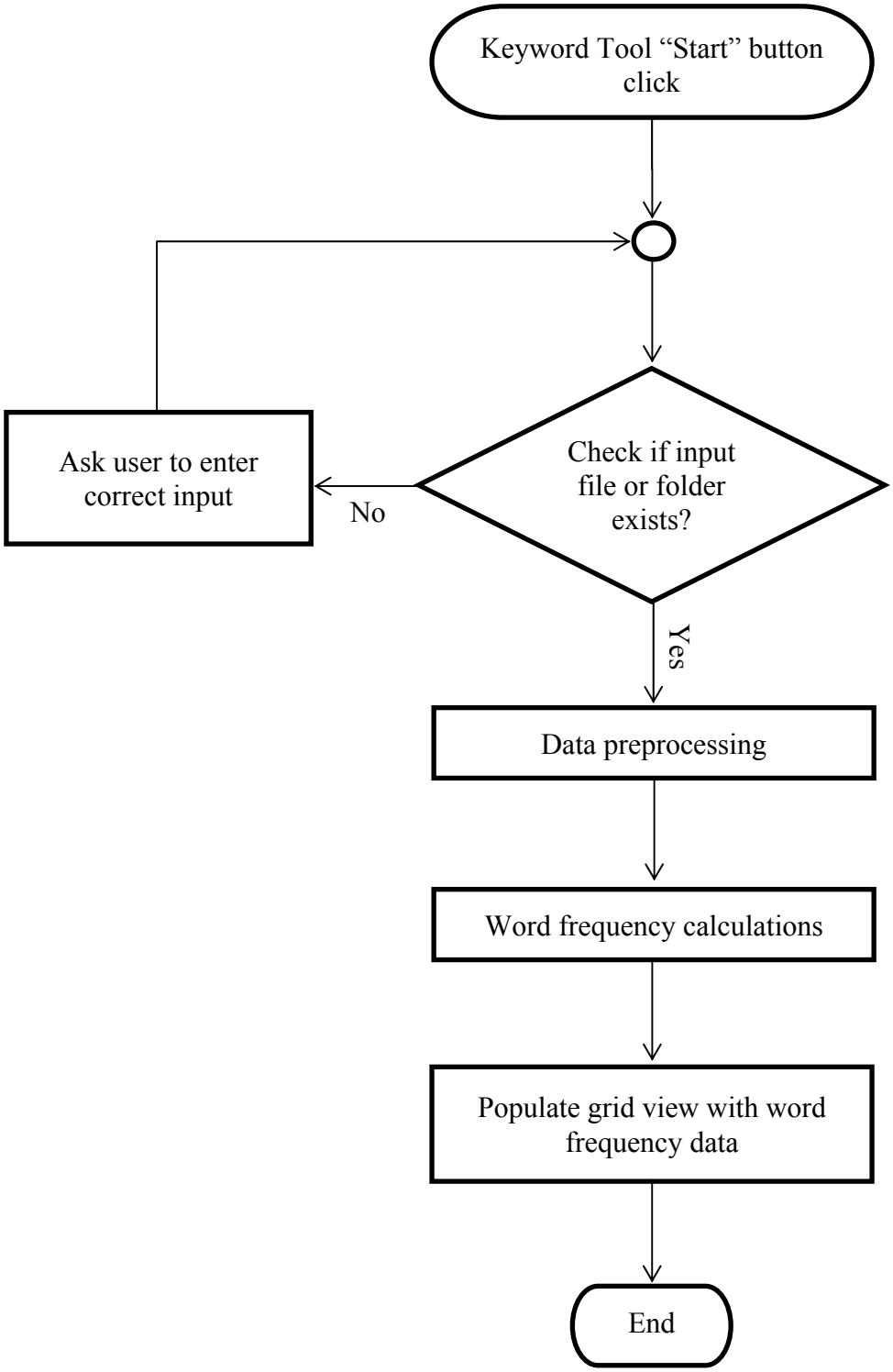

Figure 4.4: Flow chart showing the user interaction and processes after Start button click.

The frequency of each word in the *word list* is calculated and stored in a *word frequency list*. The next step is stop word removal. Stop word removal compares each word of the *word frequency list* with an existing list of stop words. If a word in the *word frequency list* matches a stop word, the word will be removed from the *word frequency list*. The 100 most frequent words are displayed in a *data grid* located in the middle left position of the Keyword Tool (figure 4.2). Two check boxes can be seen before every word. When a user clicks the first check box, a window pops up containing the input text from which the word has been extracted with the selected word and previously selected keywords (e.g. video keyword in figure 4.3) from the next list box (middle right in the Keyword Tool) highlighted. This feature is called the *Pointback* feature and helps the user with the selection of the appropriate keywords from the given file. The second check box is used to select the appropriate keyword. This ends the *Start* button click process.

4.1.2.2 'Go' Button Click Event

When a high frequency keyword is selected and *Go* button is clicked this keyword appears in the selected *keywords list box* (located in middle right position of the Keyword Tool). Now word proximity statistics are calculated for this keyword to other high frequency words based on the window weight function [w(d) from section 3.6] in both forward and backward directions from the selected keyword. This word proximity statistics are sorted and displayed along with their respective keywords in descending order in a data grid along with *pointback* capability.

If a pointback check box of a particular keyword is checked, the pointback dialog box opens the input file with the keywords in the keywords list box and the selected word from the data grid in different colors.

The analyst now examines to find whether there exists a semantic relationship between the highlighted words. If a semantic relationship exists, then the analyst clicks the *Go* button to add this keyword to the keywords list box. This process of keywords selection continues until the analyst comes up with a meaningful keyword set.

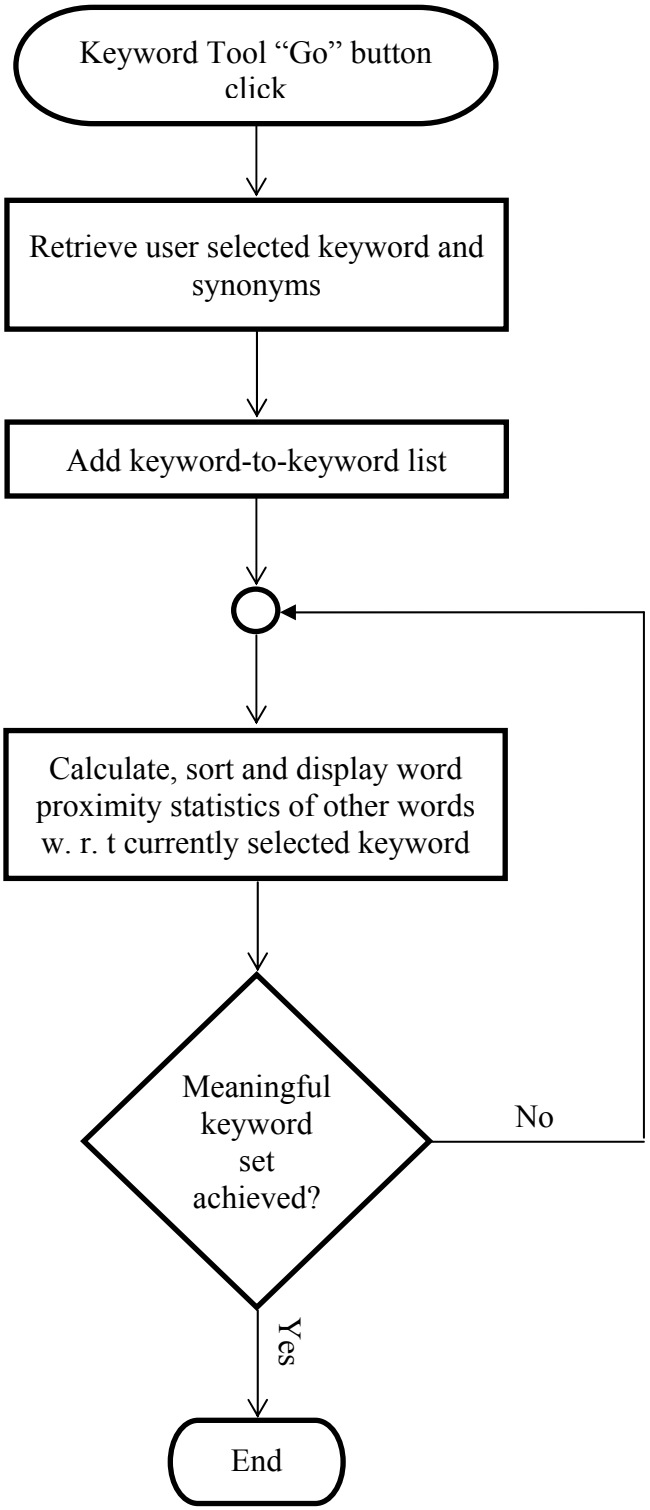

Figure 4.5: Flow chart showing the user interaction and processes after Go button click.

4.1.2.3 'Save' Button Click Event

When the *Save* button is clicked, a file name is automatically generated based on the selected keywords and is inserted into the file name field of the save dialog box. If the next button clicked by the user is cancel, the save dialog box will be closed. Otherwise if the user clicks save button the data entered by the user is written into a file in .KDF format. The file includes all the details about version, stemming, path of the file from which keywords are generated, window length, window weight function constant (a), keywords, synonyms and phrases.

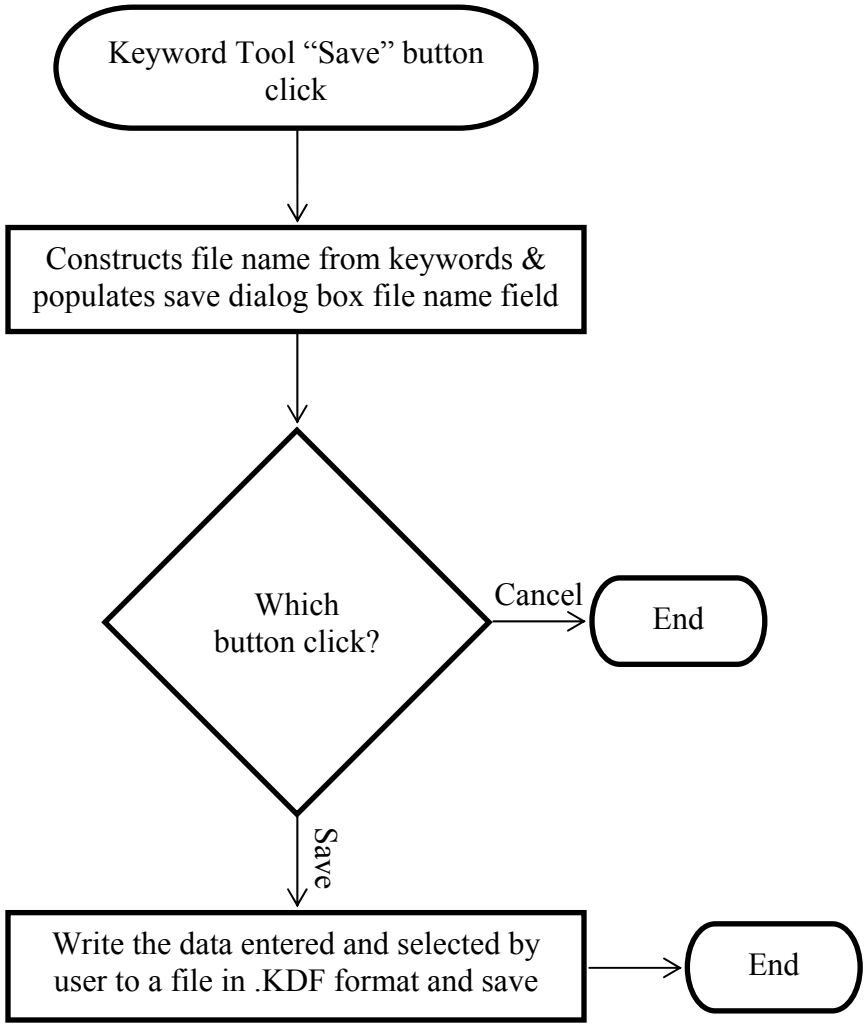

Figure 4.6: Flow chart showing the user interaction and processes after Save button click.

4.1.3 Keyword Tool Output

The output of the Keyword Tool is a Keyword Descriptor file, which contains the user-selected keywords, stemming information, synonyms, phrases and window length.

4.1.4 Keyword Tool Output File Storage Format

The output of the Keyword Tool is stored in XML format as an increasing amount of data is stored and transmitted using the XML format. Keeping this in mind the output of the Keyword Tool is stored in XML format. The extension of the output file is named as .KDF.

An example of .KDF format is shown below in figure 4.7.

```
<keywordTool version="1.1">
<stemming used="no" stemmer="porter"></stemming>
<source folder="no" url="no"
file="yes">C:\Users\Kiran\Desktop\adjeroh_main.txt</source>
<windowLength length="20"></windowLength>
<keywords>video,partitioning,adaptive</keywords>
<synonyms></synonyms>
<phrases></phrases>
</keywordTool>
```

Figure 4.7: .KDF XML format sample.

"KeywordTool" tag contains the attribute 'version', which contains the version information.

"Stemming" tag contains attributes 'used' and 'stemmer'. The former attribute contains "no" if stemming is not used and "yes" if stemming is used, and the later contains name of the stemming algorithm used.

"Source" tag contains the attributes 'folder', 'url' and 'file'. Attribute 'folder' contains "no" if the source text is from a webpage or file and 'yes' if path of a folder is given by the user containing multiple files. Attribute 'url' contains "no" if the source text is not from an online

webpage and “yes” if the source text is from an online webpage. Attribute ‘file’ contains ‘no’ if the input data is not from a single file and ‘yes’ if the data is from a single file. “Source” tag also contains path of the file or folder or URL given by the user.

“WindowLength” tag contains the attribute ‘length’, which contains the user given document vector window size or default window size of 20.

“Keywords” tag contains the comma-separated keywords selected by the user.

“Synonyms” tag contains the synonyms for the keywords given by the user.

“Phrases” tag contains the multi word keywords and these phrase keywords are even included in the keywords tag.

4.2 Learner Tool

Learner Tool operates on text of known content from which keywords have been extracted. It generates and clusters document vectors from this training file. It also provides point-back to the original text and highlights the text for the selected document vector so that the analyst can identify classes/clusters of vectors that embody the targeted semantic content. It finally generates *semantic signature descriptors*, which contains the information pertaining to the selected document vectors cluster and the keyword descriptors associated with the document vectors contained in the cluster.

4.2.1 Learner Tool Input Data

The Inputs to the Learner Tool are a KDF, some document(s) (in .txt or .html or RCV1 XML format), clustering algorithm and number of clusters document vectors should be clustered into.

4.2.2 User Interaction with the Learner Tool and Underlying Processes

This section describes about the user interaction process with the Learner Tool. This tool is used for assisting the user with the semantic signature generation for a document.

4.2.2.1 'Start' Button Click Event

When the user clicks the *Start* button, the inputs provided to the tool are checked for validity. There will be three inputs to the tool. The first input contains the path of the Keyword descriptor file or folder. The second input contains the path of the file from which the signature has to be extracted. The third input is the clustering algorithm and the number of clusters. The paths displayed in the text boxes are verified to see if they exist or not. If it does not exist, it displays a message to the user asking to enter correct input. If the file exists, the keyword descriptor file is parsed to extract the data from the XML tags which include version, stemming, path of the file from which keywords are generated, window length, function constant, keywords, synonyms and phrases. This data along with the training document is given as input to the preprocessor.

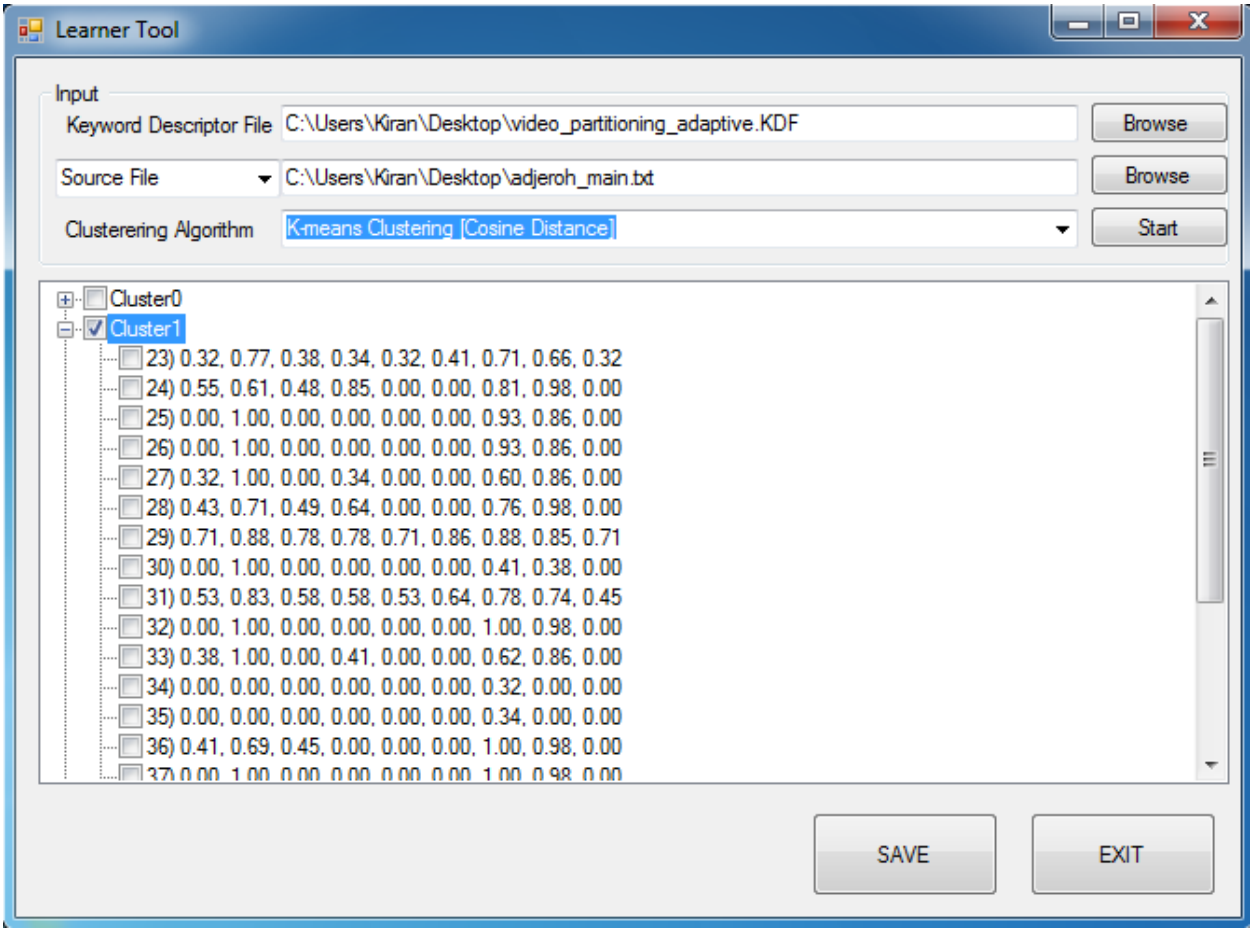

Figure 4.8: A document vectors cluster selected in the Learner Tool.

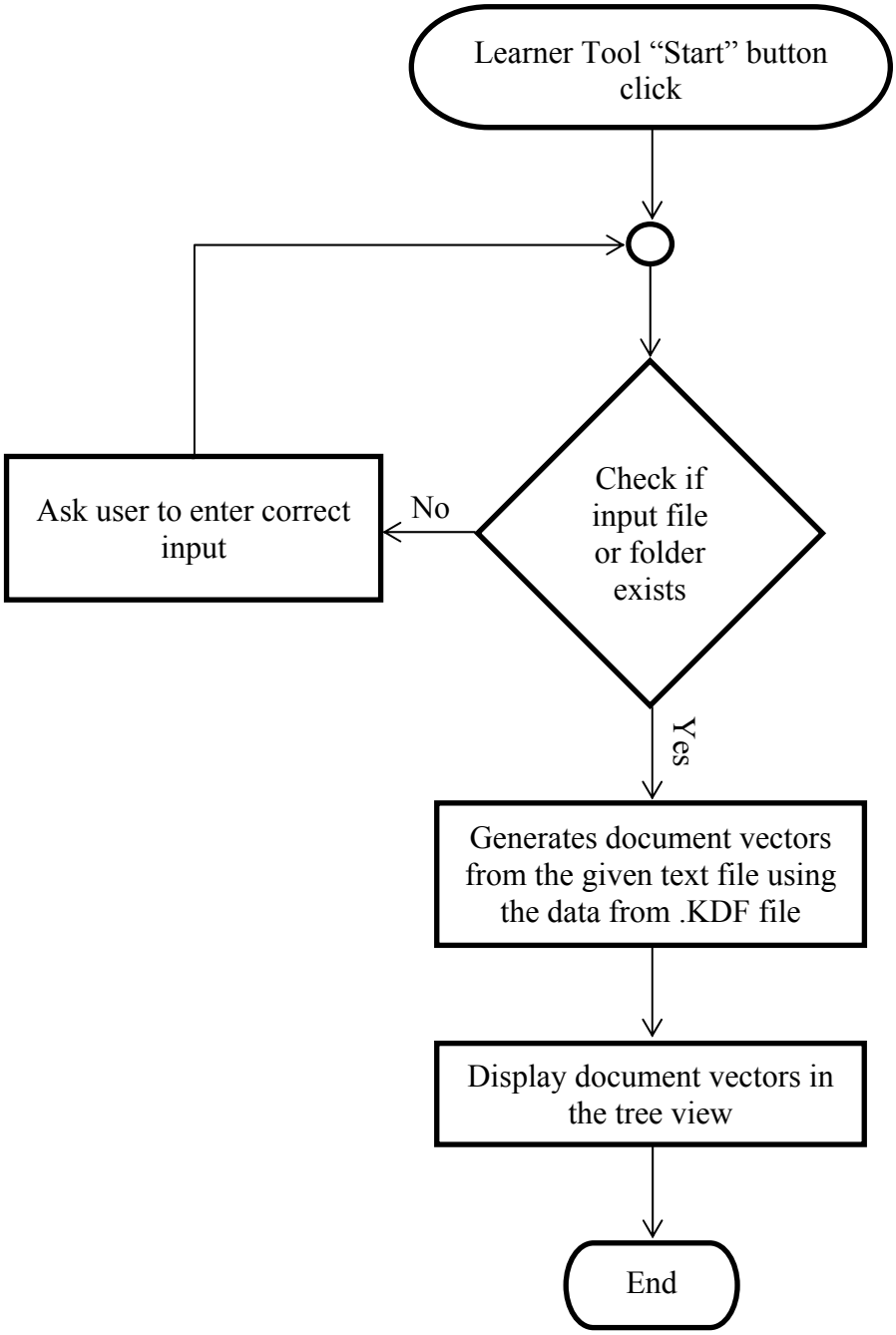

Figure 4.9: Flow chart showing the user interaction and processes after Start button click.

The first step of the preprocessing is to determine the type of the file. If it is an html file, html tags are removed and only text is considered for further processing. If it is an xml file, only the headline and body of the document is considered for further processing. Phrase replacement is

done if any phrases are extracted while parsing the keyword descriptor file. Otherwise it skips the phrase replacement part of the code.

The phrases in the document are replaced with temporary phrase markers. The entire document is then split into words that are stored in a *word list*. Now, the temporary phrase markers are replaced with their corresponding phrases.

Stemming is done if the stemming tag from the keyword descriptor file contains 'true'. Otherwise it skips the stemming part of the code. Stemming in the tool is done with the help of porter stemmer algorithm.

Synonyms replacement is done if any synonyms are extracted while parsing the keyword descriptor file. Otherwise it skips the synonym replacement part of the code. The synonyms in the document are replaced with its synonymous keyword. This completes the preprocessing stage.

Now the vectors are generated using the weight calculations as follows. The base table has two columns which stores the keywords and their respective positions. The window index table contains two columns that contain the starting and ending keyword positions in the base table for a particular window. Using the keyword position information from the base table and the window index table, weights are calculated for each keyword to all other keywords in the window only in forward direction. Using these weights document vectors are generated as described in section 3.9.1. The generated vectors are given to the clustering algorithm selected by the user specifying the number of clusters. The clusters along with the vectors in each cluster are displayed in a tree view along with pointback for the user to select the appropriate cluster. When the pointback check box is checked, a dialog box is displayed highlighting the text corresponding to the vector in the given document.

4.2.2.2 'Save' Button Click Event

When the *save* button is clicked, it checks whether a cluster has been selected by the user or not. If it is not selected, appropriate message is displayed to the user asking him/her to select a cluster. If a cluster is selected, it automatically generates a file name based on the KDF file and populates the file name field in the save dialog box. If the next button clicked by the user is

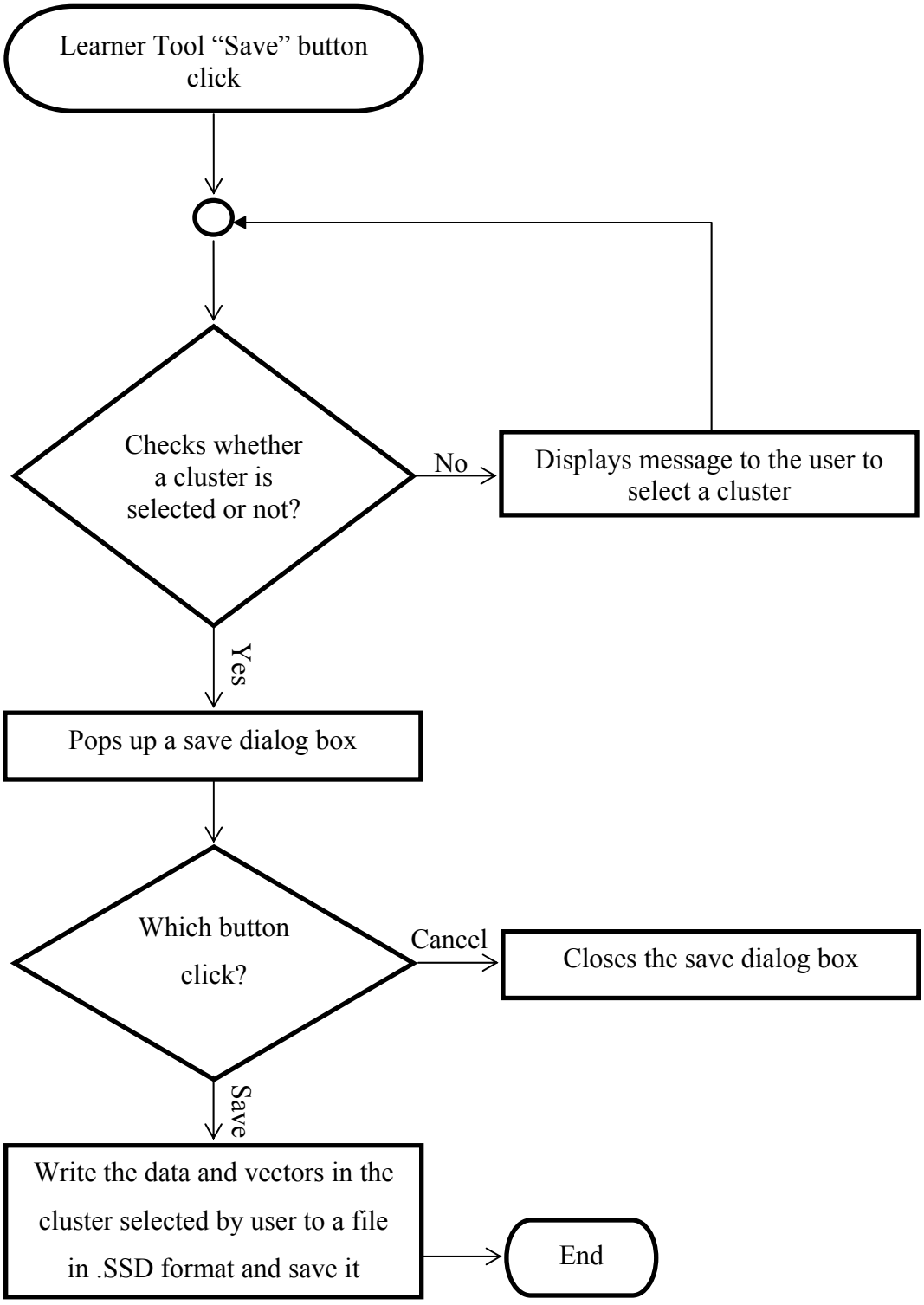

Figure 4.10: Flow chart showing the user interaction and processes after “Save” button click.

Cancel, the save dialog box will be closed. Otherwise, if the user clicks *Save* button the data entered by the user is written into a file in .SSD format, which includes all the details about Keyword Tool version, stemming, path of the file from which keywords are generated, window length, function constant, keywords, synonyms and phrases from the KDF file. Vectors from the selected cluster, centroid, distance measure, radius of the cluster, Learner Tool version, path of the document from which the document vectors are generated, path of the KDF given by the user, name of the clustering algorithm and the number of clusters are also saved in the .SSD file.

4.2.3 Learner Tool Output

The output of the Learner Tool is a semantic signature descriptor file, which contains the semantic signature(s) from the document and other information required for generating document vectors from other documents.

4.2.4 Learner Tool Output File Storage Format

The output of the Learner Tool is stored in the XML format. Extension of the output file is called as .SSD. Abbreviation for SSD is semantic signature descriptor. This output file format has been appended at the end with the XML tags from the input .KDF file. An example of the .SSD file format is shown in figure 4.10.

The descriptions of the XML tags used in the SSD file format are as follows:

“ClassificationTool” tag contains the attribute ‘version’, which contains the version information of the Learner Tool.

“KdfSource” tag contains the full path of the KDF file given as input to the tool by the user.

“Source” tag contains the attributes ‘folder’ and ‘file’. Attribute ‘folder’ contains “no” if the source text is from a file and ‘yes’ if path of a folder is given by the user containing multiple files. Attribute ‘file’ contains ‘no’ if the input data is not from a single file and ‘yes’ if the data is from a single file. In between the closing and ending tags it contains the path of the file or folder or URL containing data.

“Clusterer” tag contains the attribute ’name’, which has the name of the clustering algorithm used for clustering the document vectors. In between the closing and ending tags it contains the value for the number of clusters.

```
<ClassificationTool version="1.1">
<kdfSource>C:\Users\Kiran\Desktop\video_partitioning_adaptive.KDF</kdfSource>
<source folder="no" file="yes">C:\Users\Kiran\Desktop\adjeroh_main.txt</source>
<clusterer name="kmeans">6</clusterer>
<centroid r="0.708296971721544" distanceMeasure="CD">0.0851, 0.7176, 0.0815, 0.0913, 0.0326,
0.0513, 0.3067, 0.3167, 0.0308</centroid>
<vectors>0.3162,0.7661,0.3846,0.3363,0.3162,0.4138,0.7082,0.6565,0.3162;0.5473,0.6073,0.4829,0.
8547,0,0,0.807,0.9806,0;0,1,0,0,0,0,0,0,0;0,1,0,0,0,0,0,0,0;0,1,0,0,0,0,0.9285,0.8575,0;0,1,0,0,0,0,0.9
285,0.8575,0;0.3162,1,0,0.3363,0,0,0.5981,0.8575,0;0.432,0.7069,0.4856,0.6402,0,0,0.765,0.9806,0;
0.7071,0.8801,0.7809,0.7809,0.7071,0.8575,0.8801,0.8475,0.7071;0,1,0,0,0,0,0.4138,0.3846,0;0,1,0,
0,0,0,0,0,0;0,1,0,0,0,0,0,0,0;0.53,0.8285,0.5812,0.5812,0.53,0.6402,0.7808,0.7409,0.4472;0,1,0,0,0,0,
1,0.9806,0;0,1,0,0,0,0,0,0,0;0,0.4856,0,0,0,0,0,0,0;0.3846,1,0,0.4138,0,0,0.6224,0.8575,0;0.4138,0.69
23,0.4472,0,0,0,1,0.9806,0;0,1,0,0,0,0,1,0.9806,0;0,1,0,0,0,0,0,0,0;0,1,0,0,0,0,0.9285,0.8575,0;0,1,0.2
676,0,0,0.2822,0,0,0;0,1,0,0,0,0,0.9285,0.8575,0;0,1,0,0,0,0,0,0,0;</vectors>
<keywordTool version="1.1">
<stemming used="False" stemmer="porter"></stemming>
<source folder="no" url="no" file="yes">C:\Users\Kiran\Desktop\adjeroh_main.txt</source>
<windowLength length="20"></windowLength>
<keywords>video,partitioning,adaptive</keywords>
<synonyms></synonyms>
<phrases></phrases>
</keywordTool>
</ClassificationTool>
```

Figure 4.11: .SSD XML format sample.

“Vectors” tag contains all the document vectors from the cluster selected by the user and these document vectors are separated by a semicolon.

"Centroid" tag contains the attributes 'r' and 'distanceMeasure'. Depending on the distance measure used attribute 'r' will contain radius or similarity measure. 'r' will contain radius if Euclidean distance measure is used and similarity measure if cosine similarity measure is used. Attribute 'distanceMeasure' contains the name of the distance measure used in the clustering algorithm. 'ED' represents Euclidean distance measure and 'CD' represents cosine similarity measure. In between the closing and ending tags it contains the centroid of the cluster selected by the user.

The remaining tags are from the KDF file given as input by the user to the Learner Tool.

4.3 Data Analysis Tool

The Data Analysis Tool (DAT) operates on a corpus of data (plain text, html, etc.) with unknown content (may include known content files as markers) along with a library of semantic signature descriptors (SSDs). The DAT detects semantic features by generating document vectors for the input documents and computing vector hit (within the semantic signature classes/clusters) frequencies for each file. The DAT also generates an output matrix known as *document analysis matrix*. The columns of this matrix will be the SSDs and the rows will be the input documents. Each row of this matrix corresponds to a document and will be referred to as *semantic feature vector* from this point on in the document.

4.3.1 Data Analysis Tool Input Data

Inputs to the Data Analysis Tool are semantic signature library, testing corpus and some options from the user.

4.3.2 User Interaction with the Data Analysis Tool and Underlying Processes

4.3.2.1 'Start' Button Click Event

Flow chart for *Start* button click is shown in figure 4.12. There will be two inputs to the tool. The first input contains the path of the semantic signature descriptor file or folder. The second input contains the path of the testing corpus. The Path of the files in the text boxes is verified to see whether they exist or not when a user clicks the *Start* button. If it does not exist, it displays a

message to the user asking to enter correct input. If the files exist, the semantic signature descriptor file is parsed to extract the data from the XML tags which include version of the classification tool, source of keyword descriptor files, path of the file from which semantic signatures are extracted, name of the clustering algorithm, number of clusters, centroid, radius, distance measure, vectors, Keyword Tool version, stemming, path of the file from which keywords are generated, window length, function constant, keywords, synonyms and phrases. This data along with document from the testing corpus is given as input to the preprocessor.

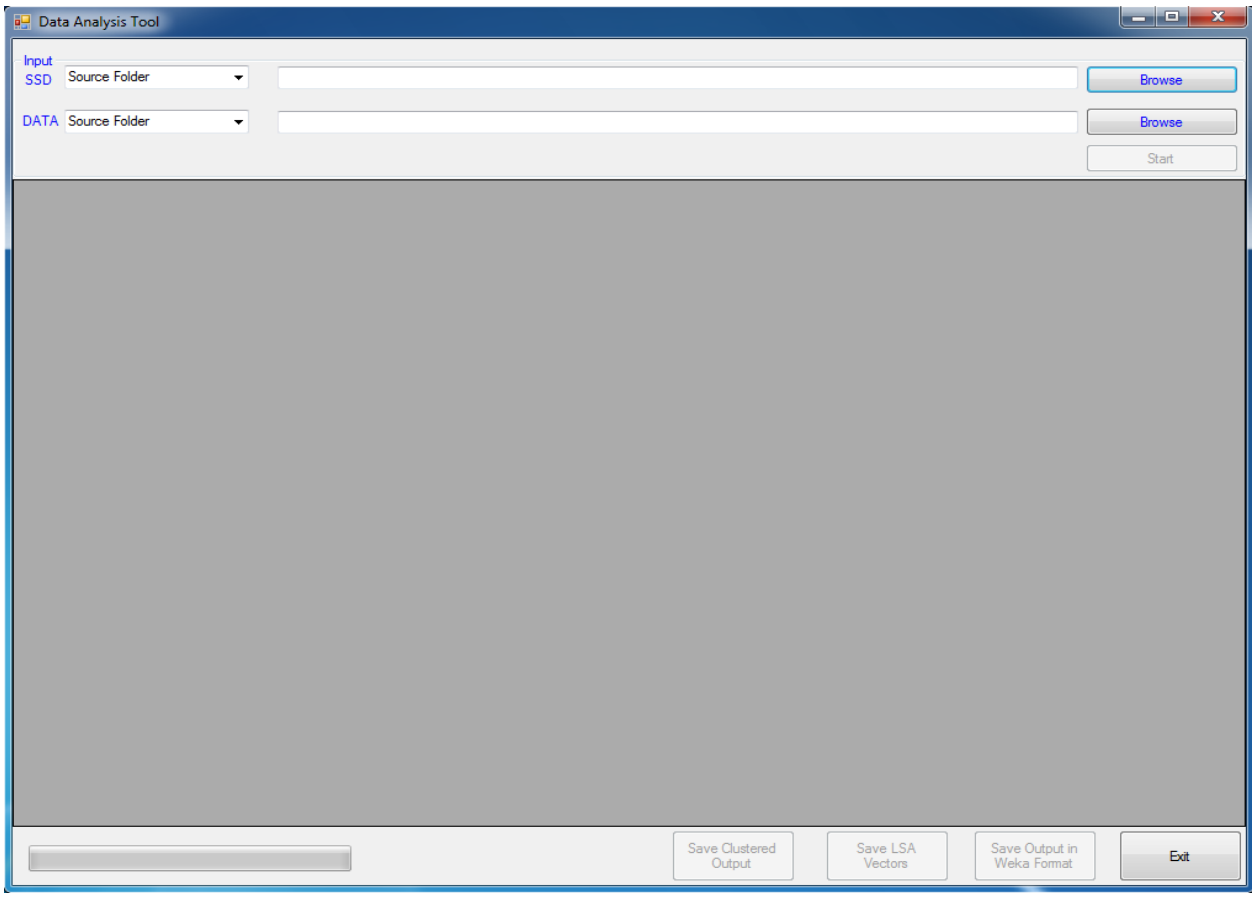

Figure 4.12: Screen shot of the Data Analysis Tool.

The first step of the preprocessing is to determine the type of the file. If it is an HTML file, HTML tags are removed and only text is considered for further processing. If it is an XML file, only the headline and body of the document is considered for further processing.

Phrase replacement is done if any phrases are extracted while parsing the semantic signature descriptor file. The phrases in the document are replaced with temporary phrase markers. The

entire document is then split into words that are stored in a *word list*. Now, the temporary phrase markers are replaced with the phrases.

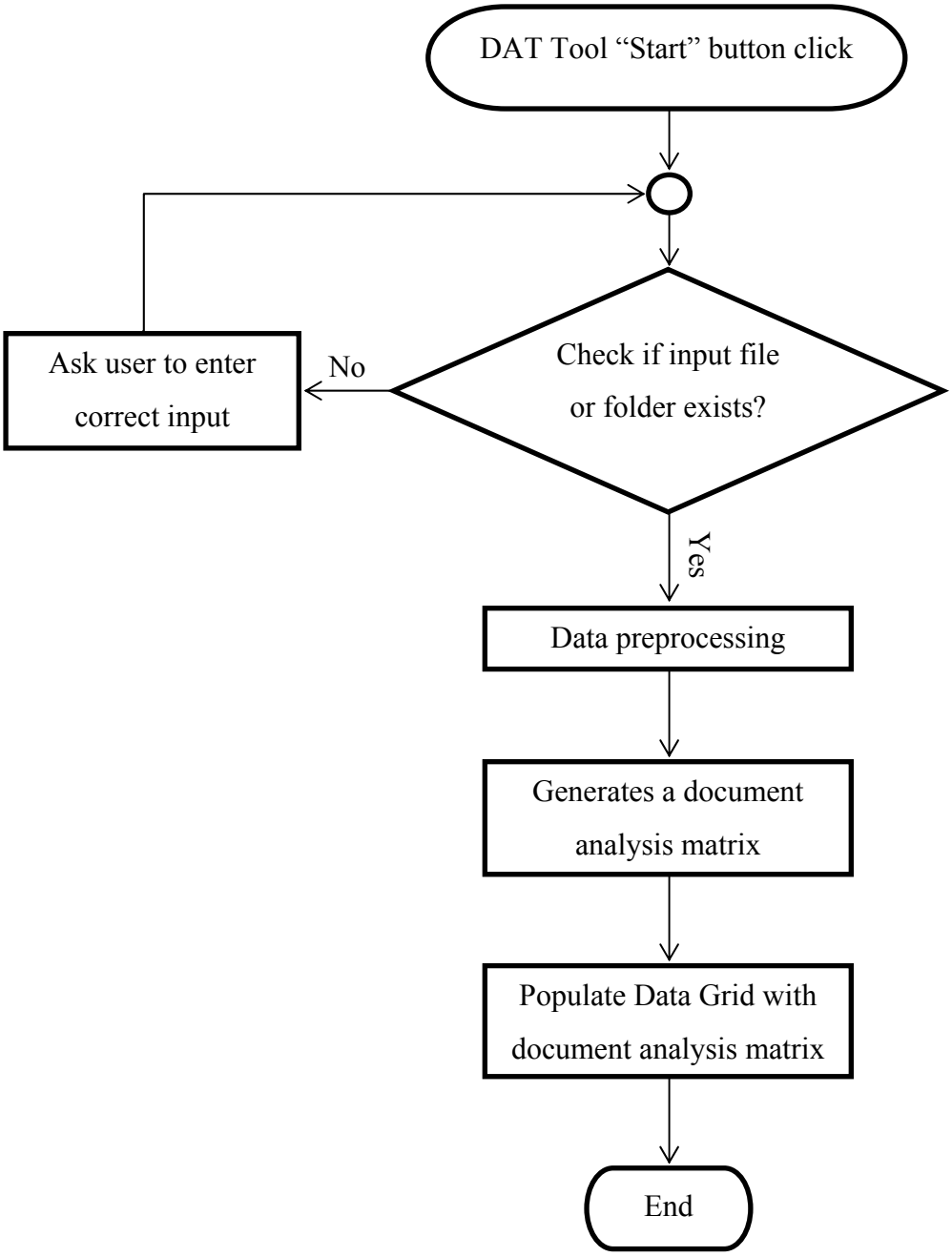

Figure 4.13: Flow chart showing the user interaction and processes after Start button click.

Stemming is done if the stemming tag from the semantic signature descriptor file contains 'true'. Otherwise it skips the stemming part of the code. Stemming in the tool is done with the help of porter stemmer algorithm.

Synonyms replacement is done if any synonyms are extracted while parsing the semantic signature descriptor file. Otherwise it skips the synonym replacement part of the code. The synonyms in the document are replaced with their respective keywords. This completes the preprocessing stage.

Now the vectors are generated using the weight calculations as follows. The base table has two columns which stores the keywords and their respective positions. The window index table contains two columns that contain the starting and ending keyword positions in the base table for a particular window. Using the keyword position information from the base table and the window index table, weights are calculated for each keyword to all other keywords in the window only in forward direction. Using these weights document vectors are generated as described in section 3.9.1.

The distance between the document vectors generated from the testing document and the centroid from the SSD file are calculated. If the distance is less than radius of the cluster and if all the keywords are present in the document vectors window, then it is calculated as a hit (i.e. falls into the cluster from SSD). This is how a *hit* is calculated in the case of CR1 SSDs. Then the resultant frequencies of hits are populated in a matrix named as *document analysis matrix*, whose columns headers are SSDs and the rows headers are documents from the testing corpus. In the case of CR2 SSDs as input, a document vector falls into a cluster from the SSD if the cosine similarity between itself and centroid is less than the similarity measure from SSD.

Similarity scores are computed for a document if CR3 SSDs are given as input to the Data Analysis Tool. Cosine distance between each document vector and all the semantic signatures from the SSD are calculated. Then the maximum of the previously calculated cosine distances corresponding to all the document vectors are averaged to get a similarity score for the document from which document vectors were generated. This is how similarity score is calculated for a document in the case of CR3 SSDs.

Using Latent Semantic Analysis semantic feature vectors are calculated as follows. The numbers of appearances of each distinct keyword from all the given SSD files are calculated for each document of the testing corpus. Using this frequency information, term-document matrix is generated. This term document matrix is given as input to the Singular Value Decomposition

function, which gives us three matrices namely U, V and Σ. The rows of the V matrix are the semantic feature vectors.

The document analysis matrix that is generated earlier is now populated in a data grid view for the user to view. Here, if a semantic feature vector contains all zeroes it is discarded from the data grid view.

4.3.2.2 'Save Clustered Output' Button Click Event

Flow chart for *Save Clustered Output* button click is shown in figure 4.13. The flow chart shows the user interaction process and the functions that take place in the background in response to the user interaction. When the *Save Clustered Output* button is clicked, the user is presented with a dialog box to enter the number of clusters, distance measure for the clustering algorithm and the cluster representation. The dialog box will be closed if the user clicks *Cancel* button. Otherwise if the user clicks *OK* button, a save dialog box is opened. If the user clicks *Cancel* button, the save dialog box will be closed. Otherwise if the user clicks *OK* button, the Data Analysis Tool clusters the rows (semantic feature vectors) of the document analysis matrix, writes the semantic feature vectors and cluster information to a file and saves it. Before clustering the semantic feature vectors from the Data Analysis Tool, all the semantic feature vectors whose elements are all zeros are removed. It also writes testing corpus document paths to an html file and saves it.

4.3.2.3 'Save Output in WEKA Format' Button Click Event

Flow chart for *Save Output in WEKA Format* button click is shown in figure 4.14. The flow chart shows the user interaction process and the functions that take place in the background in response to the user interaction. When the *Save Output in WEKA Format* button is clicked, it pops up a save dialog box asking the user to enter the name of the file. If the user clicks *Cancel* button then the save dialog box will be closed. Otherwise it writes the semantic feature vectors to a file in ARFF format and saves it. Here it discards the vectors whose elements are all zeroes. It also writes testing corpus document paths to an HTML file and saves it.

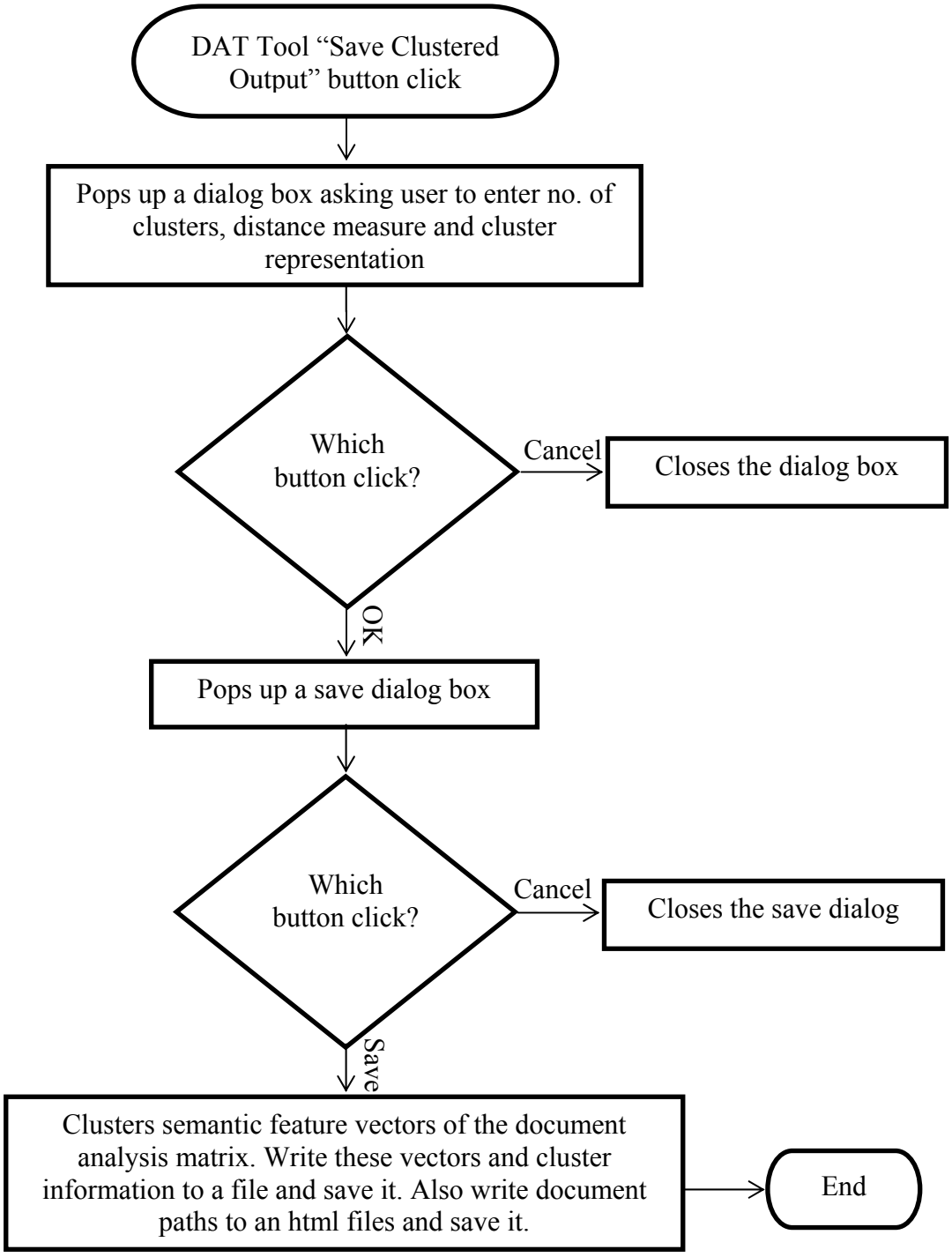

Figure 4.14: Flow chart showing the user interaction and processes after Save Clustered Output button click.

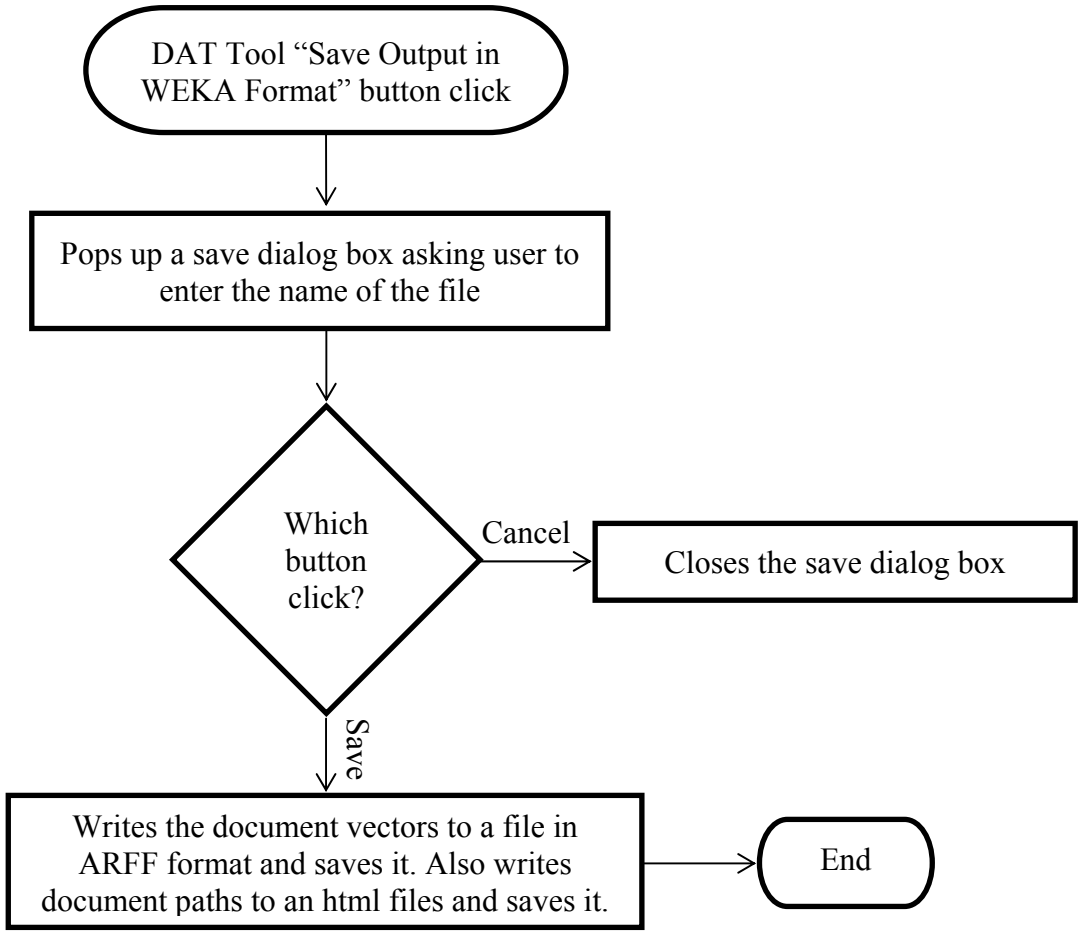

Figure 4.15: Flow chart showing the user interaction and processes after "Save Output in WEKA Format" button click.

4.3.2.4 'Save Clustered LSA Vectors' Button Click Event

Flow chart for *Save Clustered LSA Vectors* button click is shown in figure 4.15. The flow chart shows the user interaction process and the functions that take place in the background in response to the user interaction. When the *Save Clustered LSA Vectors* button is clicked, it displays a dialog box asking the user to enter the number of clusters, distance measure for the clustering algorithm and the cluster representation. If the user clicks *Cancel* button then the dialog box will be closed. Otherwise it opens up a save dialog box. If the user clicks *Cancel* button then the save dialog box will be closed. Otherwise it clusters the LSA semantic feature

vectors then writes these semantic feature vectors and cluster information and to a file and saves it. Here it discards the semantic feature vectors whose elements are all zeroes.

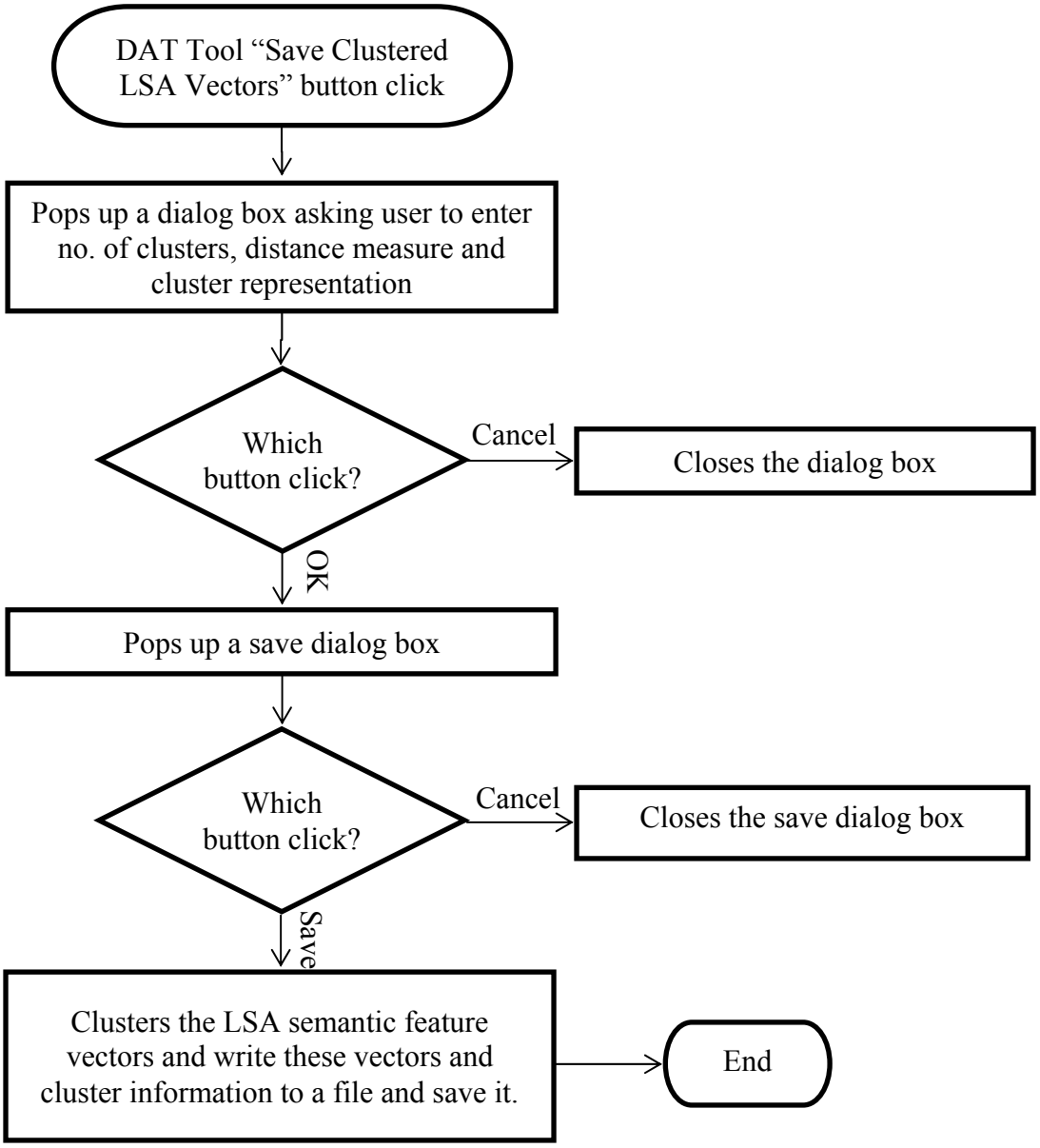

Figure 4.16: Flow chart showing the user interaction and processes after "Save Clustered LSA Vectors" button click.

4.3.3 DAT Output

The output of the Data Analysis Tool is document analysis matrix. Each row of this matrix is a semantic feature vector. Semantic feature vectors differ for different types of cluster representations.

4.3.4 DAT Output File Storage Format

The output of the Data Analysis Tool is stored in attribute relation file format (ARFF), which is the file format used by WEKA data mining tool. The header of the ARFF file consists of the name of relation, a list of attributes (SSDs in this case) and their data types. Anything after % sign are treated as comments and this is a single line comment. The rows in the below figure after @data are semantic feature vectors. An example of the .ARRF format is shown in figure 4.17.

```
@relation 'Document Clustering'
% comments
@attribute 'adj-adaptive-video-partitioning-ws20' numeric
@attribute 'cu-software-testing-ws20' numeric
@attribute 'ross-iris-synthesis-ws20' numeric
@data
33,0,0
4,0,0
0,40,0
0,4,0
```

Figure 4.17: Example of attribute relation file format.

Chapter 5: Experimental Setup

Chapter 5 describes the experimental approach used to benchmark the semantic signature concept. Three experiments are conducted in this thesis and they are described in sections 5.1, 5.2 and 5.3. Section 5.1 explains the approach used to compare the three cluster representations: CR1, CR2 and CR3. Section 5.2 describes the approach used to retrieve documents related to two different categories and classify them. Section 5.3 explains the approach employed in estimating CR1's information retrieval capabilities using a statistical test.

5.1 Experiment for Evaluating Cluster Representations (CRs)

5.1.1 Objectives

The objective of this experiment is to determine the best document vectors cluster representation among the existing three document vectors cluster representations (CR1, CR2 and CR3) and the best distance measure for k-means clustering of the semantic feature vectors. Here we are determining how much better these representations are at retrieval and sub-classification of closely related documents into sub-categories.

5.1.2 Corpus

The training data set for this experiment is a collection of nine research papers (also referred to as main papers in this thesis) written by nine different authors. The testing data set consists of a collection of 5 reference research papers cited in the papers of each of the 9 different authors and training data set itself. The total number of research papers in the testing data set is 54. All the authors are from the computer science and electrical engineering department at West Virginia University.

5.1.3 Procedure

From each main research paper 4 keyword descriptor files (KDFs) have been collected. So the total number of KDFs in the KDF library is 36. Then the Learner Tool is used to generate SSDs for all those 36 KDFs using the same text documents from which KDFs are generated. In this experiment two groups of SSDs are generated. In the first group of 36 SSDs (SSD-KE), document vectors are clustered using the Euclidean K-means clustering algorithm. In the second group of 36 SSDs (SSD-KC), document vectors are clustered using Spherical K-means clustering algorithm.

In case CR1, the input to the Data Analysis Tool is the SSD-KE group of 36 SSDs. In both cases CR2 and CR3, the input to the Data Analysis Tool is the SSD-KC group of 36 SSDs. These 36 SSDs (SSD-KE or SSD-KC) and the testing corpus are given as input to the Data Analysis Tool, which generates an N by M matrix whose columns are SSDs and rows are the documents from the testing corpus. Here 'M' is the number of SSDs and 'N' is the number of documents in the testing corpus. This matrix is called *document analysis matrix*. Let us call each row of this matrix a semantic feature vector. Each element of this matrix consists of the frequency of document vector hits in the corresponding documents in the case of CR1 and CR2. In the case of CR3, each element of document analysis matrix consists of similarity scores. How these scores are calculated is described in chapter 3.

At this point a different type of clustering is performed to assign the various documents analyzed into clusters based on the semantic feature vectors described above. The semantic feature vectors are clustered using Euclidean or Spherical K-means clustering algorithms. In the next step, these document clusters are analyzed manually to determine the quality of documents being retrieved and clustered. The results are presented in chapter 6.

5.2 Multi-category Retrieval and Classification of Documents from a Huge Corpus

5.2.1 Objectives

The objective of this experiment is to find how different clustering protocols applied at the SSD level (CR1, CR2 and CR3) and the document matrix level affect our tools ability to retrieve documents of interest from a large corpus and properly classify the retrieved documents into subclasses. For these experiments we use the Reuters Corpus Volume 1 (RCV1) [8]. This set of documents contains approximately one year of Reuters wire service articles to which a set of tags has been manually added to indicate the type of content or category of each article. In this study we used articles with the category tags GSCI and GHEA. We chose to retrieve documents relating to space science and general health. We then studied how the tools performed using various SSD cluster protocols (CR1, CR2, and CR3) and different document clustering protocols.

5.2.2 Corpus

The testing and training corpora for this experiment were taken from Reuters Corpus (RCV1. Documents belonging to two different categories were selected for training phase of the experiment. And those two categories were:

1. Space topic, which is a subtopic of science and technology (GSCI) and
2. Subtopics of general health (GHEA).

5.2.3 Procedure

In this experiment, 51 files are randomly collected from Reuters corpus belonging to GSCI and GHEA category. These 51 files act as a training corpus and were not a subset of testing corpus. The testing corpus was also taken from Reuters corpus and consists of 67,952 newswire articles for the month of November 1996. From each training document KDFs and SSDs are generated using the Keyword Tool and Learner Tool respectively. In all 91 semantic signature descriptor files (SSDs) are derived from 51 training documents using the Keyword and Learner Tools. Of

these 91 SSDs, 46 SSDs are derived from space topic related documents and the other 45 SSDs from subtopics of general health related documents. These SSDs and the testing corpus are given as input to the Data Analysis Tool for generating the document analysis matrix. Each row of document analysis matrix corresponds to a retrieved document.

	Relevant	**Non Relevant**
Retrieved	True Positives (tp)	False Positives (fp)
Not Retrieved	False Negatives (fn)	True Negatives (tn)

Table 5.1: Confusion matrix

$$Precision\ (P) = \frac{tp}{tp + fp} \quad (5.1)$$

$$Recall\ (R) = \frac{tp}{tp + fn} \quad (5.2)$$

$$F1 - measure = \frac{2PR}{P + R} \quad (5.3)$$

The document retrieval results are evaluated using precision, recall and F-measure [20]. Precision is the percentage of retrieved documents that are relevant. Recall is the percentage of relevant documents that are retrieved.

After evaluation of the document retrieval results, document classification accuracies are calculated. These accuracies are calculated for classification results of Euclidean and Spherical K-means clustering with 2, 3 and 4 clusters.

5.3 Chi-Square Test for Document Retrieval Experiment

5.3.1 Objectives

The objective of this experiment is to employ a statistical hypothesis to measure the document retrieval rates. By grouping sets of SSDs together to form 4 different retrieval filters and

assuming they are statistically independent we can measure the true positive and false positive rates for each of the 4 retrieval filters. Thus this experiment is to demonstrate the information retrieval [13, 21] capabilities of the semantic signature concept using a Chi-square minimization technique.

5.3.2 Corpus

The testing and training corpora for this experiment are taken from Reuters Corpus Volume 1 (RCV1) [8]. From RCV1, documents belonging to space topics are selected for training phase of the experiment. Space topics are a sub topic of science and technology (GSCI) in RCV1.

5.3.3 Procedure

This experiment is designed and executed to analyze the document retrieval capabilities of semantic signatures. The reliability of semantic signatures in retrieving documents is tested using Chi-square test.

5.3.3.1 Assumptions and Theory behind Chi-square Test

Let us assume there are 'D' documents in the testing corpus. Here testing and training corpora are subsets of RCV1 corpus. The intersection of documents between training and testing corpora is an empty set. Out of these '*D*' documents let us assume that there are '*N*' documents of interest and '*(D - N)*' documents of no interest. To conduct this chi-square test SSD files are generated from documents in the training corpus which contain text that is of interest to us. The process of generating SSD files is explained in chapters 3 and 4. These SSD files are now divided into four groups randomly. Let these SSD file groups

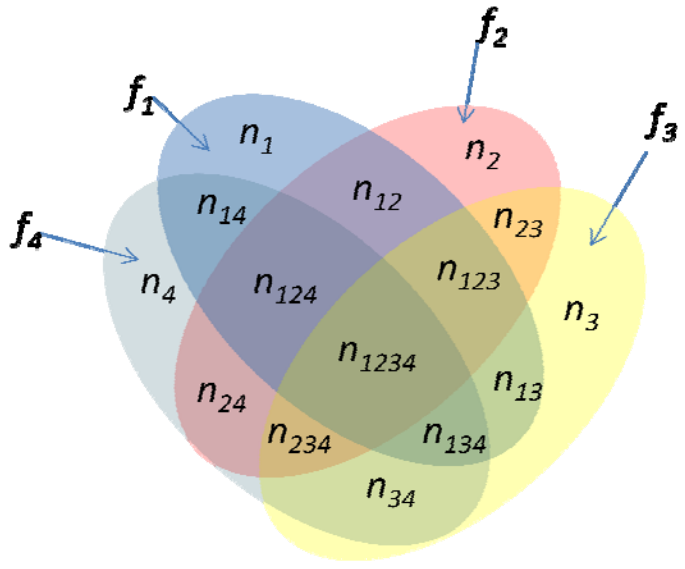

Figure 5.1: Venn diagram showing the number of documents retrieved by different combinations of filters

be g_1, g_2, g_3 and g_4. These four SSD file groups are used to retrieve documents from the testing corpus individually, one at a time and are also used in implementing filters f_1, f_2, f_3 and f_4. These filters retrieve documents that are similar to those used in the generation of SSD files. Let n_1, n_2, n_3 and n_4 be the number of documents retrieved by the filters f_1, f_2, f_3 and f_4, respectively. Let us assume that there is some intersection between the documents filtered by these four filters as illustrated in Fig. 5.1 namely, n_{12}, n_{13}, n_{14}, n_{23}, n_{24}, n_{34}, n_{123}, n_{124}, n_{134}, n_{234}, n_{1234}. Where n_{12} would be the number of documents retrieved by filters f_1 and f_2 and so on for the other n's. Let p_1, p_2, p_3 and p_4 be the probabilities of retrieving the documents of interest by the filters f_1, f_2, f_3 and f_4, respectively. Similarly, let q_1, q_2, q_3 and q_4 be the probabilities of retrieving the documents that are of no interest by the filters f_1, f_2, f_3 and f_4, respectively. As a part of chi-square test we are making the assumption that all the four filters are statistically independent.

From the above assumptions we can derive 15 equations for the expected values of number of documents retrieved by all the combinations of filters f_1, f_2, f_3 and f_4. They are as follows:

$$\bar{n}_1 = Np_1 + Mq_1 \quad – \quad \text{(a1)} \quad \text{where M = (D - N)}$$

$$\bar{n}_2 = Np_2 + Mq_2 \quad – \quad \text{(a2)}$$

$$\bar{n}_3 = Np_3 + Mq_3 \quad – \quad \text{(a3)}$$

$$\bar{n}_4 = Np_4 + Mq_4 \quad – \quad \text{(a4)}$$

$$\bar{n}_{12} = Np_1p_2 + Mq_1q_2 \quad – \quad \text{(a5)}$$

$$\bar{n}_{13} = Np_1p_3 + Mq_1q_3 \quad – \quad \text{(a6)}$$

$$\bar{n}_{14} = Np_1p_4 + Mq_1q_4 \quad – \quad \text{(a7)}$$

$$\bar{n}_{23} = Np_2p_3 + Mq_2q_3 \quad – \quad \text{(a8)}$$

$$\bar{n}_{24} = Np_2p_4 + Mq_2q_4 \quad – \quad \text{(a9)}$$

$$\bar{n}_{34} = Np_3p_4 + Mq_3q_4 \quad – \quad \text{(a10)}$$

$$\bar{n}_{123} = Np_1p_2p_3 + Mq_1q_2q_3 \quad – \quad \text{(a11)}$$

$$\bar{n}_{124} = Np_1p_2p_4 + Mq_1q_2q_4 \quad – \quad \text{(a12)}$$

$$\bar{n}_{134} = Np_1p_3p_4 + Mq_1q_3q_4 \quad – \quad \text{(a13)}$$

$$\bar{n}_{234} = Np_2p_3p_4 + Mq_2q_3q_4 \quad – \quad \text{(a14)}$$

$$\bar{n}_{1234} = Np_1p_2p_3p_4 + Mq_1q_2q_3q_4 \quad – \quad \text{(a15)}$$

In the above equations the values to the left are the expected values of the number of documents retrieved by all the combinations of filters f_1, f_2, f_3 and f_4.

Given a particular experimental measurement, we can find its contribution to Chi-square as

$$\delta^2 = \frac{(E-O)^2}{\sigma^2} \quad (5.1)$$

The equation for Chi-square is

$$\chi^2 = \sum \delta^2 \quad (5.2)$$

The above equation can be written as:

$$\chi^2 = \sum_{i=1}^{4} \delta_i^2 + \sum_{i,j=1;i\neq j}^{4} \delta_{ij}^2 + \sum_{i,j,k=1;i\neq j\neq k}^{4} \delta_{ijk}^2 + \sum_{i,j,k,l=1;i\neq j\neq k\neq l}^{4} \delta_{ijkl}^2 \quad (5.3)$$

The two outcome nature of the filter follows a binomial distribution. So the variances from the expected values from the equations (a1) through (a15) are as follows:

$$\sigma_1^2 = Np_1(1-p_1) + Mq_1(1-q_1) \quad – \quad \text{(b1)}$$

Where M = (D-N)

$$\sigma_2^2 = Np_2(1-p_2) + Mq_2(1-q_2) \quad – \quad \text{(b2)}$$

$$\sigma_3^2 = Np_3(1-p_3) + Mq_3(1-q_3) \quad – \quad \text{(b3)}$$

$$\sigma_4^2 = Np_4(1-p_4) + Mq_4(1-q_4) \quad – \quad \text{(b4)}$$

$$\sigma_{12}^2 = Np_1p_2(1-p_1p_2) + Mq_1q_2(1-q_1q_2) \quad – \quad \text{(b5)}$$

$$\sigma_{13}^2 = Np_1p_3(1 - p_1p_3) + Mq_1q_3(1 - q_1q_3) \quad – \quad \text{(b6)}$$

$$\sigma_{14}^2 = Np_1p_4(1 - p_1p_4) + Mq_1q_4(1 - q_1q_4) \quad – \quad \text{(b7)}$$

$$\sigma_{23}^2 = Np_2p_3(1 - p_2p_3) + Mq_2q_3(1 - q_2q_3) \quad – \quad \text{(b8)}$$

$$\sigma_{24}^2 = Np_2p_4(1 - p_2p_4) + Mq_2q_4(1 - q_2q_4) \quad – \quad \text{(b9)}$$

$$\sigma_{34}^2 = Np_3p_4(1 - p_3p_4) + Mq_3q_4(1 - q_3q_4) \quad – \quad \text{(b10)}$$

$$\sigma_{123}^2 = Np_1p_2p_3(1 - p_1p_2p_3) + Mq_1q_2q_3(1 - q_1q_2q_3) \quad – \quad \text{(b11)}$$

$$\sigma_{124}^2 = Np_1p_2p_4(1 - p_1p_2p_4) + Mq_1q_2q_4(1 - q_1q_2q_4) \quad – \quad \text{(b12)}$$

$$\sigma_{134}^2 = Np_1p_3p_4(1 - p_1p_3p_4) + Mq_1q_3q_4(1 - q_1q_3q_4) \quad – \quad \text{(b13)}$$

$$\sigma_{234}^2 = Np_2p_3p_4(1 - p_2p_3p_4) + Mq_2q_3q_4(1 - q_2q_3q_4) \quad – \quad \text{(b14)}$$

$$\sigma_{1234}^2 = Np_1p_2p_3p_4(1 - p_1p_2p_3p_4) + Mq_1q_2q_3q_4(1 - q_1q_2q_3q_4) \quad – \quad \text{(b15)}$$

So the Chi-square equation from (3) can be re-written as follows:

$$X^2 = \sum_{i=1}^{4} \frac{(\bar{n}_i - n_i)^2}{\sigma_i^2} + \sum_{i,j=1;i\neq j}^{4} \frac{\left(\bar{n}_{ij} - n_{ij}\right)^2}{\sigma_{ij}^2} + \sum_{i,j,k=1;i\neq j\neq k}^{4} \frac{\left(\bar{n}_{ijk} - n_{ijk}\right)^2}{\sigma_{ijk}^2} + \sum_{i,j,k,l=1;i\neq j\neq k\neq l}^{4} \frac{\left(\bar{n}_{ijkl} - n_{ijkl}\right)^2}{\sigma_{ijkl}^2} \tag{5.4}$$

The number of degrees of freedom of the Chi-square test in this experiment is the difference between the number of elements in the summation of Chi-square and number of variables in the Chi-square equation. The number of equations is 15 and the number of variables is 9. So the number of degrees of freedom for this experiment is 6.

We minimize the equation (5.4) to get the minimum possible Chi-square. And the values associated with minimum of equation (5.4) are taken as the values of the variables p_1, p_2, p_3, p_4, q_1, q_2, q_3, q_4 and N.

Finally, errors in the values of probabilities and *N* for one standard deviation are calculated. It is the same as calculating the value of variable for a unit change in the minimum value of the Chi-square equation while minimizing the equation and keeping the variable constant.

Chapter 6: Experiment Results

This chapter contains the results for the experiments described in chapter 5 (sections 5.1, 5.2 and 5.3). Section 6.1 describes the results of the experiment described in 5.1 for assessing the effects of cluster representations and clustering algorithms on the output of the tools. Section 6.2 describes the results of the experiment for multi-category retrieval and classification of documents from a huge corpus. Section 6.3 describes the results of the statistical determination of document retrieval rates in huge corpora.

6.1 Experiment for Finding Better Cluster Representations (CRs)

6.1.1 Results

Author Name	Abbreviation
Dr. Donald Adjeroh	AD
Dr. Bojan Cukic	CU
Dr. Hany Ammar	HA
Dr. Katerina Goseva-Popstojanova	KA
Dr. Natalia Schmid	NA
Dr. Daryl Reynolds	RE
Dr. Arun Ross	RO
Dr. Tim Menzies	TM
Dr. Matthew Valenti	VA

Table 6.1: Abbreviations for professor's names whose papers are used in this experiment

The columns in the tables 6.2 through 6.7 represent authors whose research papers and reference papers are used as training and testing documents. Rows represent the clusters into which the

papers are clustered and are numbered 0 through 8 (C0, C1… C8). In the following tables 6.2 through 6.7, numbers 0 to 53 represent the 54 testing documents. The 9 authors from whom papers are collected are represented in the following tables with a two letter abbreviation. The abbreviations are given in table 6.1.

6.1.1.1 Semantic Feature Vectors Generated Using CR1 SSDs and Clustered Using Euclidean K-means

In this case we use the SSD cluster characterization CR1 (Euclidean k-means) and the clustering of the documents using Euclidean k-means on the semantic feature vectors (the rows in the document analysis matrix).

Analysis of individual clusters

In the tables 6.2 - 6.7, C0 through C8 represent 9 clusters.

	AD	CU	HA	KA	NA	RE	RO	TM	VA
C0		6,9,10	15						
C1	1,3,4,5	7	16	19	25,28	31,32,33	37,40	44,45,47	50,51,52,53
C2					24				
C3		8	12,14	18,20,21				42,43	
C4	0,2		13,17						
C5						30,34,35			48,49
C6		11		22,23					
C7							36,38,39,41		
C8					29				

Table 6.2: Semantic feature vector clustering results with Euclidean K-means when CR1 SSDs are given as input to DAT

Cluster0: In this cluster some of the Cukic and Ammar papers (or their references) are grouped together. The papers 6, 9, 10 and 15 are talking about ‘software, testing’ (47-74), ‘software, reliability’ (5-19), ‘software, engineering’ (9-25) and ‘software, states’ (3-23). After manual

analysis it was found that these papers are really talking about similar topic. Thus we can say that this cluster has papers related to software topics.

Cluster1: All the papers with in this cluster have their semantic feature vectors closer to origin with values mostly in the range 1-10. This is the reason that these papers are grouped together. And this is not a good cluster because this cluster has papers related to different topics. We view this as a weakness in the Euclidean k-means clustering as a method of assigning papers to catagories.

Cluster2: In this cluster there is only one paper and it is Schmid's main paper (24). The reason that this paper fell in this cluster is because this paper has large number of hits (39) for a SSD with keywords 'information rate' and 'empirical'. 39 hits in this semantic feature vector are throwing it away from the other semantic feature vectors (in Euclidean space) that may be somewhat similar to it. Again this is one of the drawbacks of K-means clustering with Euclidean distance measure.

Cluster3: This cluster has papers by Cukic, Ammar, Katrina and Menzies or their references. In this cluster the papers have hits from SSDs with keywords 'software, testing' (11-26), 'software, reliability' (4-30), 'software, engineering' (3-10), 'software, states' (6-16). Thus we can say that this cluster has papers related to software topics.

Cluster4: This cluster has papers by Adjeroh and Ammar or their references. In this cluster the papers have hits for SSDs with keywords 'motion, complexity, scene' (26-66), 'video, scene, complexity' (26-31). After manually analyzing the papers I found that the papers from Adjeroh (0, 2) and Ammar (13, 17) actually have one thing in common, 'complexity'. Adjeroh's papers dealt with complexities related to image processing topic and Ammar's papers dealt with complexities related to software topic. Even though papers are related to different topics they have something in common.

Cluster5: This cluster has papers by Reynolds and Valenti or their references. In this cluster the papers have hits for SSDs with keywords 'power, relays' (1-28), 'diversity, cooperative, wireless' (1-7), 'relay, nodes' (15-41). After manually analyzing the papers I found that these papers are all related to wireless topics.

Cluster6: This cluster has papers by Cukic and Katrina or their references. In this cluster the papers have the hits from SSDs with keywords ‘software, testing’ (45-150), ‘software, reliability’ (29-76), ‘software, engineering’ (11-30), ‘software, states’ (13-30). After manual analysis of these papers it was found that they are all related to software topics.

Cluster7: This cluster has papers by Ross or his references. In this cluster the papers have hits from SSDs with keywords ‘iris, synthesis’ (34-41), ‘iris, synthetic, real’ (37-56). After manual analysis of these papers it was found that they are all related to biometrics topics.

Cluster8: This cluster has papers by Schmid or her references. In this cluster the paper has the majority of hits for SSDs with keywords ‘pca, encoded, data’ (6). After manual analysis of these papers it was found that they are all related to communications and biometrics topic.

Overall, the cluster representation CR1 with papers clustered using Euclidean K-means on the rows of the document analysis matrix is great except for cluster1 where all the papers whose semantic feature vectors are close to origin are grouped together.

6.1.1.2 Semantic Feature Vectors Generated Using CR1 SSDs and Clustered Using Spherical K-means

In this case we use the SSD cluster characterization CR1 (Euclidean k-means) and the clustering of the documents using Spherical k-means on the semantic feature vectors (the rows in the document analysis matrix).

Analysis of individual clusters

Cluster0: No papers in this cluster.

Cluster1: No papers in this cluster.

Cluster2: No papers in this cluster.

Cluster3: This cluster has papers by Adjeroh, Reynolds and Valenti or their references. In this cluster the papers have hits for SSDs with keywords ‘motion, complexity, scene’ (11-66), ‘temporal, class, video’ (10-49), ‘model, sensitivity, parameters’ (2-5). After manual analysis of these papers it was found that, papers 0, 2, 4, and 5 are related to image processing topics and 32,

49, 50 are related to wireless topics. Majority of the papers in this cluster belong to image processing topics.

Cluster4: This cluster has majority of the papers by Ross or his references. In this cluster the papers have hits from SSDs with keywords ‘iris, synthesis’ (13-48), ‘iris, synthetic, real’ (20-56). After manual analysis of these papers it was found that majority of the papers are related to biometrics topics.

	AD	CU	HA	KA	NA	RE	RO	TM	VA
C0									
C1									
C2									
C3	0,2,4,5					32			49,50
C4	3				25		36,37,38,39, 40,41	44	
C5						30,31,33,34, 35			48,51,52,53
C6		6,7,8,9,10,11	12,13,14,15, 16,17	18,19,20,21, 22,23				42,43,45,47	
C7									
C8	1				24,28,29				

Table 6.3: Semantic feature vector clustering results with Spherical K-means when CR1 SSDs are given as input to DAT

Cluster5: This cluster has papers by Reynolds and Valenti or their references. In this cluster the papers have hits for SSDs with keywords ‘power, relays’ (1-28), ‘diversity, cooperative, wireless’ (1-7), ‘relay, nodes’ (15-41). After manual analysis of these papers it was found that these papers are all related to wireless topics.

Cluster6: This cluster has papers by Cukic, Ammar, Katrina and Menzies or their references. In this cluster the papers have hits from SSDs with keywords ‘software, testing’ (1-150), ‘software, reliability’ (3-76), ‘software, engineering’ (3-30), ‘software, states’ (3-30). After manual analysis it has been found that all but one paper belonging to software topics are grouped in this cluster.

Cluster7: No papers in this cluster.

Cluster8: This cluster has papers by Schmid or her references. In this cluster the papers have the majority of hits for SSDs with keywords ‘schur, convex, function, concave’ (1-7), ‘pca, encoded, data’ (6-10). After manual analysis of these papers it was found that all the papers except one are related to communications and biometrics topics.

After analyzing the results of CR1 Spherical K-means clustering, we can see that the semantic feature vectors of the papers pointing in the same direction are clustered together. In the case of CR1 Euclidean K-means, the semantic feature vectors which are in similar directions in this multi-dimensional Euclidean space are clustered together.

6.1.1.3 Semantic Feature Vectors Generated Using CR2 SSDs and Clustered Using Euclidean K-means

In this case we use the SSD cluster characterization CR2 (Euclidean k-means) and the clustering of the documents using Euclidean k-means on the semantic feature vectors (the rows in the document analysis matrix).

Analysis of individual clusters

Cluster0: This cluster has a paper by Ross. In this cluster the paper has hits from SSDs with keywords ‘furrows, radial, concentric’ (12), ‘impostor, distributions, images’ (4), ‘iris, synthesis’ (12), ‘iris, synthetic, real’ (11). According to manual analysis this paper belongs to the biometrics topic.

Cluster1: This cluster has a paper by Adjeroh and one of his reference papers. In this cluster the papers have hits from SSDs with keywords ‘adaptive, video, partitioning’ (73, 43), ‘motion, complexity, scene’ (28, 3). According to manual analysis these papers belong to image processing topics.

Cluster2: This cluster has a paper referenced by Cukic. In this cluster the paper has hits from SSDs with keywords ‘failure, subdomain, testing’ (48), ‘software, testing’ (150), ‘testing, partition, random’ (4), ‘software, reliability’ (18), ‘estimate, reliability’ (2), ‘reliability, software, architecture’ (21), ‘software, engineering’ (11). According to manual analysis this paper belongs to a software topic.

Cluster3: This cluster has papers by Cukic, Ammar and Menzies or their references. In this cluster the papers have hits from SSDs with keywords 'software, testing' (12-26), 'software, engineering' (3-11). After manual analysis it has been found that all papers belong to software topics.

	AD	CU	HA	KA	NA	RE	RO	TM	VA
C0							36		
C1	0,4								
C2		11							
C3		8	13,14					42,43	
C4									49
C5					24				
C6	1,2,3,5	7	16	19,21	25,27,29	30,31,32,33, 34,35	37,38,39,41	44,45,47	48,50,52,53
C7		6,9,10	15	18,20					
C8			12,17	22,23					

Table 6.4: Semantic feature vector clustering results with Euclidean K-means when CR2 SSDs are given as input to DAT

Cluster4: This cluster has a paper referenced by Valenti. In this cluster the paper has hits for SSDs with keywords 'ad hoc, wireless, networks' (80), 'user, channels, inter' (9), 'power, uplink, cooperative' (2). After manual analysis of this paper it was found that it is related to wireless topic.

Cluster5: In this cluster there is only one paper and it is Schmid's main paper (24). The reason that this paper fell in this cluster is because this paper has large number of hits for SSDs with keywords 'information rate' and 'empirical' (38), 'pca, encoded, data' (11), 'recognition, capacity, empirical' (62), 'templates, object' (23). This paper belongs to communications and biometrics topics. There are other papers which are similar to this one but they fell in cluster6 because they have fewer hits for the same SSDs compared to it.

Cluster6: All the papers with in this cluster have their semantic feature vectors closer to origin with values mostly in the range 1-10 except for the paper by Reynolds. This is the reason that

these papers are grouped together. And this is not a good clustering outcome because this cluster has papers related to different topics.

Cluster7: This cluster has papers by Cukic, Ammar and Katrina or their references. In this cluster the papers have hits from SSDs with keywords ‘software, testing’ (16-74), ‘software, reliability’ (3-27), ‘reliability, software, architecture’ (3-28), ‘software, engineering’ (6-26). After manual analysis it has been found that all papers in this cluster belong to software topics.

Cluster8: This cluster has papers by Ammar and Katrina or their references. In this cluster the papers have hits from SSDs with keywords ‘software, testing’ (14-46), ‘software, reliability’ (11-63), ‘component, reliability’ (5-16), ‘estimate, reliability’ (1-17), ‘reliability, software, architecture’ (20-71), ‘software, engineering’ (3-17). After manual analysis it has been found that all papers in this cluster belong to software topics.

Overall the CR2 Euclidean K-means clustering is good except for cluster6 where all the papers whose semantic feature vectors are close to origin are grouped together.

6.1.1.4 Semantic Feature Vectors Generated Using CR2 SSDs and Clustered Using Spherical K-means

In this case we use the SSD cluster characterization CR2 (Spherical k-means) and the clustering of the documents using Spherical k-means on the semantic feature vectors (the rows in the document analysis matrix).

Analysis of individual clusters

Cluster0: This cluster has papers by Adjeroh and all his references. In this cluster the papers have hits for SSDs with keywords ‘adaptive, video, partitioning’ (1-73), ‘motion, complexity, scene’ (1-28). After manual analysis of these papers it was found that they are all related to image processing topics.

Cluster1: This cluster has papers by Cukic and Ammar or their references. In this cluster the papers have hits from SSDs with keywords ‘failure, subdomain, testing’ (1-48), ‘software, testing’ (17-150), ‘software, reliability’ (3-18), ‘reliability, software, architecture’ (3-21),

'software, engineering' (3-26). After manual analysis it has been found that all papers in this cluster belong to software topics.

	AD	CU	HA	KA	NA	RE	RO	TM	VA
C0	0,1,2,3,4,5								
C1		6,8,9,10,11	14,15						
C2						30,31,32,34, 35			48,49,50,53
C3						33	37,39	42,43,44,45, 47	52
C4		7		18,19,20,21, 22,23					
C5					24,25,27,29		38		
C6			12,13,16,17						
C7									
C8							36,41		

Table 6.5: Semantic feature vector clustering results with Spherical K-means when CR2 SSDs are given as input to DAT

Cluster2: This cluster has papers by Reynolds and Valenti or their references. In this cluster the papers have hits for SSDs with keywords 'ad hoc, wireless, networks' (3-80), 'relay, powers' (3-30), 'user, channels, inter' (1-10), 'diversity, cooperative, wireless' (1-10), 'relay, nodes' (1-12). After manual analysis of these papers it was found that they are all related to wireless topics.

Cluster3: This cluster has a majority of the papers by Menzies or his references. In this cluster the papers have hits from SSDs with keywords 'software, testing' (1-22), 'space, search' (1-20). After manual analysis of these papers it was found that they are all related to software topics.

Cluster4: This cluster has papers by Cukic and Katrina or their references. In this cluster the papers have hits from SSDs with keywords 'failure, subdomain, testing' (1), 'software, testing' (5-46), 'software, reliability' (5-63), 'estimate, reliability' (1-23), 'reliability, software, architecture' (5-71), 'software, engineering' (4-32). After manual analysis it has been found that all papers in this cluster belong to software topics.

Cluster5: This cluster has papers by Schmid and Ross or their references. In this cluster the papers have hits from SSDs with keywords 'pca, encoded, data' (2-11), 'recognition, capacity,

empirical' (2-62), 'templates, object' (1-23). After manual analysis it has been found that all the papers except for one belong to communications and biometrics topics.

Cluster6: This cluster has papers by Ammar or his references. In this cluster the papers have hits from SSDs with keywords 'motion, complexity, scene' (1-31), 'software, testing' (7-23), 'components, risk, factors' (3-68), 'dynamic, coupling, complexity' (3-22), 'software, reliability' (8-17), 'reliability, software, architecture' (1-25), 'software, engineering' (3-12). After manual analysis it has been found that all the papers in this cluster belong to software topics.

Cluster7: No papers in this cluster.

Cluster8: This cluster has paper by Ross or his references. In this cluster the papers have hits from SSDs with keywords 'iris, synthesis' (12-15), 'iris, synthetic, real' (0-11). After manual analysis of these papers it was found that all of them are related to biometrics topics.

After analyzing the results we can see that cluster representation CR2 with Spherical K-means document clustering is almost perfect. We can also see that all but one of the clusters in table 6.5 have only one main paper. All professors' papers and their references fell into separate clusters with high accuracy.

6.1.1.5 Semantic Feature Vectors Generated Using CR3 SSDs and Clustered Using Euclidean K-means

In this case we use the SSD cluster characterization CR3 (all the vectors of the cluster) and the clustering of the documents using Euclidean k-means on the semantic feature vectors (the rows in the document analysis matrix).

Analysis of individual clusters

Cluster0: This cluster has papers by Schmid or her references. In this cluster the papers have the high scores for SSDs with keywords 'information, rate, empirical' (0-0.9), 'schur, convex, function, concave' (0.58-0.58), 'pca, encoded, data' (0.2-0.5), 'recognition, capacity, empirical' (0.54-0.64), 'templates, object' (0-0.87). After manual analysis of these papers it was found that all the papers are related to communications and biometrics topics.

Cluster1: This cluster has papers by Adjeroh and his reference papers. In this cluster the papers have high scores for SSDs with keywords ‘adaptive, video, partitioning’ (0.3-0.5), ‘motion, complexity, scene’ (0.53-0.67). According to manual analysis these papers belongs to image processing topics.

	AD	CU	HA	KA	NA	RE	RO	TM	VA
C0					24,25				
C1	0,2,4								
C2				23				42,43	
C3		10	14,15					47	
C4		6,8,9,11		19,20					
C5		7	12,13,16,17	18,21,22					
C6	1,3,5				26,27,28,29		36,38,40	46	
C7						30,31,32,33, 34,35			48,49,50,51, 52,53
C8							37,39,41	44,45	

Table 6.6: Semantic feature vector clustering results with Euclidean K-means when CR3 SSDs are given as input to DAT

Cluster2: This cluster has papers by Katrina and Menzies or their references. In this cluster the papers have high scores for SSDs with keywords ‘failure, subdomain, testing’ (0.43-0.53), ‘software, testing’ (0.98-1.0), ‘software, reliability’ (0.69-0.88), ‘reliability, software, architecture’ (0.33-0.66), ‘search, random’ (0.35-0.81), ‘software, engineering’ (0.80-0.84), ‘space, search’ (0.53-0.65), ‘states, software’ (0.65-0.73). After manual analysis it has been found that all the papers belong to software topics.

Cluster3: This cluster has papers by Cukic, Ammar and Katrina or their references. In this cluster the papers have high scores for SSDs with keywords ‘failure, subdomain, testing’ (0.52-0.53), ‘software, testing’ (0.95-1.0), ‘software, reliability’ (0.69-0.77), ‘reliability, software, architecture’ (0.34-0.44), ‘software, engineering’ (0.53-0.84), ‘states, software’ (0.07-0.73). After manual analysis it has been found that all the papers belong to software topics.

Cluster4: This cluster has papers by Cukic and Katrina or their references. In this cluster the papers have high scores for SSDs with keywords ‘failure, subdomain’ (0.5-0.66), ‘software, testing’ (0.64-0.96), ‘testing, partition, random’ (0.55-0.79), ‘software, reliability’ (0.0-0.93), ‘component, reliability’ (0.63-0.64), ‘software, engineering’ (0.74-1.0), ‘states, software’ (0.0-0.73). After manual analysis it has been found that all the papers belong to software topics.

Cluster5: This cluster has papers by Cukic, Ammar and Katrina or their references. In this cluster the papers have high scores for SSDs with keywords ‘failure, subdomain, testing’ (0.44-0.5), ‘software, testing’ (0.96-1.0), ‘components, risk, factors’ (0.15-0.56), ‘software, reliability’ (0.65-0.89), ‘component, reliability’ (0.38-0.74), ‘estimate, reliability’ (0.49-0.73), ‘reliability, software, architecture’ (0.42-0.69), ‘software, engineering’ (0.72-0.88), ‘states, software’ (0.64-0.73). After manual analysis it has been found that all the papers belong to software topics.

Cluster6: This cluster has papers by Adjeroh, Schmid, Ross and Menzies or their references. In this cluster papers have high scores for SSDs with keywords ‘adaptive, video, partitioning’ (0-0.33), ‘pca, encoded, data’ (0-0.54), ‘recognition, capacity, empirical’ (0-0.54), ‘impostor, distributions, images’ (0-0.53), ‘iris, synthesis’ (0-0.73), ‘iris, synthetic, real’ (0-0.76). After manual analysis it has been found that the papers in this cluster belong to image processing, communications, biometrics and software topics.

Cluster7: This cluster has the main papers by Reynolds, Valenti and all their references. In this cluster papers have high scores for SSDs with keywords ‘power, uplink, cooperative’ (0.23-0.56), ‘’relay, powers’ (0-0.84), ‘user, channels, inter’ (0-0.69), ‘ad hoc, wireless, networks’ (0.38-0.74), ‘diversity, cooperative, wireless’ (0.39-0.7), ‘relay, nodes’ (0.0-1.0). After manual analysis it has been found that all the papers in this cluster belong to wireless topics. Also all the papers on wireless topics in this testing corpus are grouped into this cluster.

Cluster8: This cluster has papers by Ross and Menzies or their references. In this cluster papers have high scores for SSDs with keywords ‘iris, synthetic, real’ (0.0-0.5), ‘search, random’ (0.0-0.36), ‘space, search’ (0.53-1.0). After manual analysis it has been found that this cluster has papers belonging to biometrics and software topics.

Overall there are 7 pure clusters when CR3 cluster representation was used in combination with Euclidean K-means for clustering semantic feature vectors.

6.1.1.6 Semantic Feature Vectors Generated Using CR3 SSDs and Clustered Using Spherical K-means

In this case we use the SSD cluster characterization CR3 (all the vectors of the cluster) and the clustering of the documents using Spherical k-means on the semantic feature vectors (the rows in the document analysis matrix).

Analysis of individual clusters

Cluster0: This cluster has a paper referred to by Adjeroh. In this cluster the paper has high scores for SSDs with keywords 'adaptive, video, partitioning' (0.4), 'motion, complexity, scene' (0.53), 'impostor, distributions, images' (0.54). According to manual analysis this paper belongs to an image processing topic.

	AD	CU	HA	KA	NA	RE	RO	TM	VA
C0	3								
C1					26,27,29				
C2							37,39	44,45	
C3					28				
C4	0,2,4,5								
C5							36,38,40,41	46	
C6						30,31,32,33, 34,35			48,49,50,51, 52,53
C7	1								
C8		6,7,8,9,10,11	12,13,14,15, 16,17	18,19,20,21, 22,23	24,25			42,43,47	

Table 6.7: Semantic feature vector clustering results with Spherical K-means when CR3 SSDs are given as input to DAT

Cluster1: This cluster has papers by Schmid or her references. In this cluster the papers have high scores for SSDs with keywords 'pca, encoded, data' (0.46-0.54), 'recognition, capacity, empirical' (0.41-0.54), 'templates, object' (0.55-0.8). After manual analysis of these papers it was found that all the papers are related to communications and biometrics topics.

Cluster2: This cluster has papers by Ross and Menzies or their references. In this cluster the papers have high scores for SSDs with keywords ‘iris, synthetic, real’ (0.32-0.5), ‘search, random’ (0-0.36), ‘space, search’ (0.53-1). After manual analysis of these papers it was found that they are all related to software topics.

Cluster3: This cluster has a paper referred to by Schmid. In this cluster the paper has high scores for SSDs with keywords ‘failure, subdomain, testing’ (0.53), ‘software, testing’ (0.57), ‘testing, partition, random’ (0.48), ‘recognition, capacity, empirical’ (0.41). This paper belongs to a communications and biometrics topic.

Cluster4: This cluster has papers by Adjeroh and his references. In this cluster the papers have high scores for SSDs with keywords ‘adaptive, video, partitioning’ (0.3-0.51), ‘motion, complexity, scene’ (0.43-0.68), ‘video, scene, complexity’ (0.46-0.77), ‘schur, convex, function, concave’ (0-0.58). These papers belong to communications and image processing topics.

Cluster5: This cluster has papers by Ross, Menzies and their references. In this cluster the papers have high scores for SSDs with keywords ‘recognition, capacity, empirical’ (0-0.45), ‘impostor, distributions, images’ (0-0.53), ‘iris, synthesis’ (0.63-0.75), ‘iris, synthetic, real’ (0.49-0.76). After manual analysis of these papers it was found that all but one are related to image processing topics.

Cluster6: This cluster has papers by Reynolds, Valenti and their references. The papers have high scores for SSDs with keywords ‘power, uplink, cooperative’ (0.23-0.56), ‘relay, powers’ (0-0.84), ‘user, channels, inter’ (0-0.69), ‘adhoc, wireless, networks’ (0.38-0.74), ‘diversity, cooperative, wireless’ (0.39-0.7), ‘relay, nodes’ (0-1). After manual analysis of these papers it was found that all the papers are related to wireless communications topics.

Cluster7: This cluster has a paper referred to by Adjeroh. The paper has high scores for SSDs with keywords ‘adaptive, video, partitioning’ (0.33), ‘schur, convex, function, concave’ (0.58), ‘power, allocation, optimal’ (0.62), ‘impostor, distributions, images’ (0.41). This paper belongs to communications and biometrics topic.

Cluster8: This cluster has papers by Cukic, Ammar, Katrina, Menzies and their references. The papers have high scores for SSDs with keywords ‘failure, subdomain, testing’ (0-0.62), ‘schur,

convex, function, concave' (0-0.58), 'software, testing' (0.57-1.0), 'testing, partition, random' (0-0.78), 'components, risk, factors' (0-0.56), 'dynamic, coupling, complexity' (0-0.89), 'risk, factors, scenarios' (0-0.64), 'software, reliability' (0-0.89), 'component, reliability' (0-0.74), 'estimate, reliability' (0-0.87), 'reliability, software, architecture' (0-0.69), 'software, engineering' (0-1), 'states, software' (0-0.73). After manual analysis it has been found that all but two papers belong to software topics.

Overall there are 5 pure clusters when CR3 cluster representation was used in combination with Spherical K-means for clustering semantic feature vectors.

6.1.2 Analysis and Conclusions

In this experiment two comparisons are done. The first comparison is to see how Euclidean K-means performs relative to Spherical K-means in clustering the rows of the document analysis matrix to group documents together. Second is to see how the clustering representations CR1, CR2 and CR3 affect the document grouping. From the results of this experiment it was found that the performance of CR1, CR2 and CR3 cluster representations with Spherical K-means clustering was better than the CR1, CR2 and CR3 cluster representations with Euclidean K-means. So we can say that if there are several dimensions or features, Spherical K-means performs better than Euclidean K-means. Among all the three cluster representations it was found that CR2 with Spherical K-means seems to do a very fine clustering of papers. Also, CR1 and CR3 clustering results with Spherical K-means were similar to each other. CR1 and CR3 Spherical K-means exhibited a good generic clustering capability.

6.2 Multi-category Retrieval and Classification of Documents from a Huge Corpus

6.2.1 Results

The number of documents in the testing and training corpus is 67,952 and 51 respectively. The testing set are the documents that have been collected from the RCV1 corpus for the month of November 1996. The Reuters category tags indicating the articles content were ignored during the processing of the data but they were used in analyzing the results. The testing files and SSDs

that have been generated from the training files are given as input to the Data Analysis Tool to generate the document analysis matrix. Each row of document analysis matrix corresponds to a retrieved document.

Retrieved documents are manually categorized to find the true positive, false positive and false negative rates. To get an estimate of the true negative rate (a manual reading is impractical) the Reuters category tags were used. It was assumed that the only articles not retrieved which might be of interest would have the Reuters category tags general health (GHEA) or general science (GSCI). The performance evaluation scores namely precision, recall and F1-measure for this document retrieval experiment are found out to be 0.99, 0.85 and 0.92 respectively [20].

	Actually Space/General Health		
Predicted as Space/General Health		Related	Not Related
	Related	374 (true positive)	3 (false positive)
	Not Related	68 (false negative)	67,507 (true negative)

Table 6.8: Confusion Matrix

After retrieving the documents and measuring the performance evaluation scores, semantic feature vectors from this document analysis matrix are clustered using Euclidean and Spherical K-means with 2, 3 and 4 clusters to see how well the documents are clustered (here clustering is performed to classify retrieved documents). When semantic feature vectors are clustered using K-means four iterations are made for each clustering and a stable clustering outcome was selected. The tables 6.9 through 6.14 contain the results of Euclidean and Spherical K-means clustering with 2, 3 and 4 clusters.

	Garbage	General Health	Space
Cluster0	3	206	133
Cluster1	0	0	35

Table 6.9: Clustering results for Euclidean K-means with two clusters

	Garbage	General Health	Space
Cluster0	1	0	142
Cluster1	2	206	26

Table 6.10: Clustering results for Spherical K-means with two clusters

	Garbage	General Health	Space
Cluster0	0	0	37
Cluster1	0	0	41
Cluster2	3	206	90

Table 6.11: Clustering results for Euclidean K-means with three clusters

	Garbage	General Health	Space
Cluster0	0	4	0
Cluster1	1	4	132
Cluster2	2	198	36

Table 6.12: Clustering results for Spherical K-means with three clusters

	Garbage	General Health	Space
Cluster0	3	124	105
Cluster1	0	0	25
Cluster2	0	82	2
Cluster3	0	0	36

Table 6.13: Clustering results for Euclidean K-means with four clusters

	Garbage	General Health	Space
Cluster0	0	23	0
Cluster1	0	51	0
Cluster2	1	4	132
Cluster3	2	128	36

Table 6.14: Clustering results for Spherical K-means with four clusters

	K=2	K=3	K=4
Euclidean K-means	0.64	0.75	0.71
Spherical K-means	0.92	0.88	0.89

Table 6.15: Correct classification rate of documents retrieved with Euclidean and Spherical K-means clustering algorithm

In the above table, we find that the correct document classification rates for Spherical K-means are larger compared to Euclidean K-means for 2, 3 and 4 clusters. The average correct document classification rate for Euclidean and Spherical K-means with 2, 3and 4 clusters is found out to be 0.7 and 0.9 respectively.

6.2.2 Analysis and Conclusions

From the document classification and retrieval results of this experiment we can say that semantic signatures technique performs well within the field of information retrieval with a high precision and recall rates. From the results of this experiment it can be seen that Spherical K-means clustering algorithm performs better than Euclidean K-means clustering algorithm.

6.3 Chi-Square Test for Document Retrieval Experiment

6.3.1 Results

The following are the results for the experiment described in section 5.3:

Let the names of the testing corpora be TestingCorpus1 (TEC1) and TestingCorpus2 (TEC2). TEC1 and TEC2 are the subsets of RCV1 corpus and these two corpora are randomly selected. TEC1 and TEC2 consist of 67,952 and 65,607 documents respectively. TEC1 consists of all the newswire articles for the month of November 1996 and TEC2 consists of all the newswire articles for the month of December 1996. The number of documents, ‘D’ in TEC1 is 67,952 and 65,607 in TEC2. In all 60 SSD files are generated from the documents in training corpus. These training documents are a subset of RCV1. Intersection of training and testing corpora is a null

set. Two groups of SSDs are used in the experiments, one group containing 60 SSDs and the other containing 44 SSDs. The group consisting of 44 SSDs is a subset of the group containing 60 SSDs.

	TEC1 with 60 SSDs	***TEC2 with 60 SSDs***	***TEC1 with 44 SSDs***	***TEC2 with 44 SSDs***
n_1	302	277	84	71
n_2	164	137	201	144
n_3	186	145	132	89
n_4	138	96	169	156
n_{12}	86	63	61	47
n_{13}	124	118	41	37
n_{14}	68	50	68	50
n_{23}	72	60	99	56
n_{24}	46	36	119	111
n_{34}	107	58	58	46
n_{123}	70	54	33	28
n_{124}	45	34	61	44
n_{134}	54	43	38	32
n_{234}	39	30	48	39
n_{1234}	38	29	33	27

Table 6.16: Observed values of number of documents retrieved from the testing corpora

There are 4 sub-experiments with in this experiment and they are:

1. TEC1 with 60 SSDs: In which 60 SSDs are used in generating semantic feature vectors from the documents in TEC1 using the Data Analysis Tool.

2. TEC2 with 60 SSDs: In which 60 SSDs are used in generating semantic feature vectors from the documents in TEC2 using the Data Analysis Tool.
3. TEC1 with 44 SSDs: In which 44 SSDs are used in generating semantic feature vectors from the documents in TEC1 using the Data Analysis Tool.
4. TEC2 with 44 SSDs: In which 44 SSDs are used in generating semantic feature vectors from the documents in TEC2 using Data Analysis Tool.

The observed values (the number of documents) retrieved by individual and combinations of filters in the four experiments described in chapter 5 are given in table 6.16. The number of degrees of freedom of this Chi-square test is the difference between numbers of observed values and variables. There are 9 variables and 15 observed values. So there are 6 degrees of freedom in this Chi-square variable.

The following table has the Chi-square values for all of the above mentioned four sub-experiments.

	Chi-square Value with 60 SSD's	**Chi-square Value with 44 SSD's**
TEC1	22.9628	22.0876
TEC2	7.2258	16.2920

Table 6.17: Chi-square values obtained for the four sub-experiments

The errors in parameters $p_1, p_2, p_3, p_4, q_1, q_2, q_3, q_4$ were determined by finding the fixed value of the parameter of interest, p_1 for example, which changes the Chi-squared by 1 unit when all the other parameters are adjusted to minimize the Chi-square. This corresponds to interpreting the Chi-square function as the logarithm of a probability function and finding how much the parameter can be changed before the resulting probability changes by an amount corresponding to one standard deviation. The results of this tedious procedure are given in tables 6.18, 6.19, 6.22 and 6.23 for the four sub-experiments described previously. The middle line of the table has the most probable value of the parameter, while the first and last lines have the one standard deviation limits on the parameter.

Chi-square	p_1	p_2	p_3	p_4	q_1	q_2	q_3	q_4
23.9628	0.6806	0.4641	0.7568	0.4750	2.4379E-3	10.7231E-4	4.0276E-4	6.3148E-4
22.9628	0.7299	0.4959	0.8142	0.5086	0.0022	8.5839E-4	1.7917E-4	4.3344E-4
23.9628	0.7804	0.5284	0.8734	0.5427	1.8767E-3	6.4624E-4	-	2.3572E-4

Table 6.18: Error in Probabilities for one standard deviation change in Chi-square value on either side of its minima for TEC1 with 60 SSDs

Chi-square	p_1	p_2	p_3	p_4	q_1	q_2	q_3	q_4
8.2258	0.7999	0.4681	0.8061	0.4068	1.9922E-3	7.5719E-4	0.2247E-4	3.0762E-4
7.2258	0.8697	0.5011	0.8755	0.4362	2.2479E-3	9.49862E-4	2.1398E-4	4.7146E-4
8.2258	0.9416	0.5333	0.9510	0.4652	2.5035E-3	11.4706E-4	3.9212E-4	6.3942E-4

Table 6.19: Error in Probabilities for one standard deviation change in Chi-square value on either side of its minima for TEC2 with 60 SSDs

	Probability of finding documents that are of interest
TEC1	0.9876
TEC2	0.9954

Table 6.20: Probabilities for retrieving documents of interest from testing corpus using all the four filters with 60 SSDs

	Probability of finding documents that are of no interest
TEC1	0.0037
TEC2	0.0039

Table 6.21: Probabilities for retrieving documents of no interest from testing corpus using all the four filters with 60 SSDs

The probability of retrieving documents of interest using 60 SSDs distributed among all the four filters f_1, f_2, f_3 and f_4 is 0.9876 for TEC1 and 0.9954 for TEC2. These probabilities can be seen in table 6.20.

Chi-square	p_1	p_2	p_3	p_4	q_1	q_2	q_3	q_4
23.0876	0.4252	0.8181	0.4780	0.6806	-	3.3512E-4	3.6265E-4	2.8665E-4
22.0876	0.4533	0.8726	0.5081	0.7242	8.2688E-11	5.4254E-4	5.3663E-4	4.8235E-4
23.0876	0.4860	0.9309	0.5394	0.7700	11.0785E-5	7.5028E-4	7.1741E-4	6.7971E-4

Table 6.22: Error in Probabilities for one standard deviation change in Chi-square value on either side of its minima for TEC1 with 44 SSDs

Chi-square	p_1	p_2	p_3	p_4	q_1	q_2	q_3	q_4
17.2920	0.4218	0.7790	0.4236	0.7613	-	0.9548E-4	1.7371E-4	3.1608E-4
16.2920	0.4531	0.8506	0.4552	0.8299	6.6E-5	2.8769E-4	3.3626E-4	5.1755E-4
17.2920	0.4844	0.9261	0.4866	0.9019	21.614E-5	4.6982E-4	5.01E-4	7.1093E-4

Table 6.23: Error in Probabilities for one standard deviation change in Chi-square value on either side of its minima for TEC2 with 44 SSDs

	Probability of finding documents that are of interest
TEC1	0.9905
TEC2	0.9924

Table 6.24: Probabilities for retrieving documents of interest from testing corpus using all the four filters with 44 SSDs

	Probability of finding documents that are of no interest
TEC1	0.0016
TEC2	0.0012

Table 6.25: Probabilities for retrieving documents of interest from testing corpus using all the four filters with 44 SSDs

The probability of retrieving documents of interest from testing corpus using all the four filters can also be thought of as *Recall*. The probability of retrieving documents of interest using 44 SSDs distributed among all the four filters f_1, f_2, f_3 and f_4 is 0.9905 for both TEC1 and TEC2. These probabilities can be seen in table 6.24.

6.3.2 Analysis and Conclusions

From the results of sub-experiments 1 and 2 we can say that 66 SSDs which have been used in these experiments are able to retrieve documents of interest with high recall (0.9876 and 0.9954 respectively for sub-experiments 1 and 2). Also the Chi-square values for sub-experiments 1 and 2 are within the accepted range.

From the results of sub-experiments 3 and 4 we can see that Chi-square values are within the acceptable range. So we can say that the distributions of observed values and expected values for number of documents (that are of interest) retrieved are consistent.

After manually classifying 4000 documents from the TEC1, it was found that documents are retrieved by the Data Analysis Tool with high precision and recall. For finding the Chi-square values and the errors in the probabilities, Matlab was used and everything else was done in Microsoft Visual Studio and C#.NET.

Chapter 7: Conclusions and Future Work

Chapter 7 begins with Section 7.1 which is a broad overview of the previous content of this document. Section 7.2 describes conclusions reached based on the results of the experiments described previously in this document. Section 7.3 proposes additional open research questions which have been brought to light by the work and experimental results of this thesis. Section 7.4 proposes a possible list of applications where the concepts that are developed as a part of this thesis can be used.

7.1 Overview

In this thesis the broad field of text mining has been reviewed. Three tools as developed as a part of this thesis to conduct experiments and they are Keyword Tool, Learner Tool and Data Analysis Tool. The Keyword Tool was used to assist the analyst in the keyword selection process. The Learner Tool was used to generate semantic signatures for a document(s). The Data Analysis Tool was used to generate and classify semantic feature vectors for the testing documents using the given SSDs. The concepts developed as a part of this thesis are demonstrated using these tools by conducting three experiments.

In the first experiment three cluster representations were compared to find the best cluster representation among them. Also Euclidean K-means was compared with Spherical K-means to find the best clustering algorithm for classifying documents. From the first experiment we found that CR2 cluster representation with Spherical K-means performed well at classifying documents when compared to the others.

In the second experiment documents were retrieved from a subset of the Reuters corpus volume 1 to test the document retrieval capabilities of the concepts developed as a part of this thesis. Furthermore, documents are clustered into 2, 3 and 4 clusters to find the document classification

accuracies for Euclidean and Spherical K-means clustering. The effectiveness of the semantic signature concept at document retrieval is evaluated using performance evaluation measures namely precision, recall and F1-measure. Large values for performance measures indicate that semantic signature concept is highly effective in the field of document retrieval.

In the third experiment, where documents were retrieved from the testing corpus used in experiment 2, a Chi-square minimization was used to measure the recall. The recall of experiment 2 is significantly lower than the recall calculated determined in experiment 3. A manual review of the classification of the data from experiment 2 holds the answer contradiction. The Reuters Corpus contains some articles that are extremely short and have very little data for the programs to analyze. These account for the majority of the false negatives in the manually analyzed sample in experiment 2. This seems to imply that there is a document size issue that needs to be accounted for in the modeling of the efficiency of the filters.

Conclusions and possible future directions in which the research can be continued are discussed in the coming sections.

7.2 Conclusions

From the results of experiment 1 we found that cluster representation 2 with Spherical K-means performed extremely well at sub-classification of academic papers of different authors into separate clusters. We can say that cluster representation 2 clearly outperformed cluster representation 1 and cluster representation 3. In the case of document clustering the cosine similarity measure clearly outperformed Euclidean distance measure when used with K-means clustering.

The large values of precision (0.99), recall (0.85), and F1-measure (0.92) for the document retrieval experiment in section 6.2 clearly suggests that the concept of semantic signatures is extremely well suited for single or multi category document retrieval applications. The comparison between Spherical and Euclidean k-means at clustering documents for the purpose of classification clearly shows that Spherical K-means performs better at classifying documents than Euclidean K-means.

I think the clustering results of the experiments in sections 6.1 and 6.2 of this thesis can be further improved by using feature selection methods so that unnecessary features (SSDs) can be removed. The removal of these features will reduce the dimensions of the semantic feature vectors thus leading to better clustering results.

Experiment 2 and 3 complement each other. The manual analysis of experiment 2 shows that the program has trouble with extremely short files. Experiment 3 shows that for the files large enough to be properly handled by these programs the tools developed here are extremely precise.

After manually analyzing some of the SSDs, I found they are not retrieving some of the documents that are similar in semantic content. I found this situation arises when SSDs are retrieved from documents that are small and which by themselves are not able to provide a good semantic signature for that topic. It is called *underfitting* or *undertraining* in supervised learning terminology.

7.3 Future Work

I think using lemmatization instead of suffix stripping stemming algorithms can be used to further improve the results. Tools used in this thesis are capable of dealing with Unicode textual data. Therefore these experiments can be done with documents from various languages represented in Unicode format. But for each and every language different stop word list [6] and stemming algorithm has to be used. Currently the keywords are being selected manually and this can be automated to further speed up the process. Currently we are using supervised learning in the Learner Tool. It can be automated and thus making it unsupervised.

7.4 Applications

Our research has led us to the following possible future paths for research in this area.

- Text mining of chat messages for specific content (e.g., sexual predators, identity theft, illegal drug sales)
- Filtering massive data streams for items with specific content (e.g. web pages, chats, blogs, tweets, Face book live news feeds)

- Recognizing messages from a single individual who is using aliases

Appendix A: List of Stop Words

a	beforehand	every	how
about	behind	everyone	however
above	being	everything	hundred
across	below	everywhere	i
after	beside	except	ie
afterwards	besides	few	if
again	between	fifteen	in
against	beyond	fifty	inc
all	bill	fill	indeed
almost	both	find	interest
alone	bottom	fire	into
along	but	first	is
already	by	five	it
also	call	for	its
although	can	former	itself
always	cannot	formerly	keep
am	cant	forty	last
among	co	found	latter
amongst	computer	four	latterly
amoungst	con	from	least
amount	could	front	less
an	couldnt	full	ltd
and	cry	further	made
another	de	get	many
any	describe	give	may
anyhow	detail	go	me
anyone	do	had	meanwhile
anything	done	has	might
anyway	down	hasnt	mill
anywhere	due	have	mine
are	during	he	more
around	each	hence	moreover
as	eg	her	most
at	eight	here	mostly
back	either	hereafter	move
be	eleven	hereby	much
became	else	herein	must
because	elsewhere	hereupon	my
become	empty	hers	myself
becomes	enough	herself	name
becoming	etc	him	namely
been	even	himself	neither
before	ever	his	never

nevertheless	should	thus	within
next	show	to	without
nine	side	together	would
no	since	too	yet
nobody	sincere	top	you
none	six	toward	your
noone	sixty	towards	yours
nor	so	twelve	yourself
not	some	twenty	yourselves
nothing	somehow	two	
now	someone	un	
nowhere	something	under	
of	sometime	until	
off	sometimes	up	
often	somewhere	upon	
on	still	us	
once	such	very	
one	system	via	
only	take	was	
onto	ten	we	
or	than	well	
other	that	were	
others	the	what	
otherwise	their	whatever	
our	them	when	
ours	themselves	whence	
ourselves	then	whenever	
out	thence	where	
over	there	whereafter	
own	thereafter	whereas	
part	thereby	whereby	
per	therefore	wherein	
perhaps	therein	whereupon	
please	thereupon	wherever	
put	these	whether	
rather	they	which	
re	thick	while	
same	thin	whither	
see	third	who	
seem	this	whoever	
seemed	those	whole	
seeming	though	whom	
seems	three	whose	
serious	through	why	
several	throughout	will	
she	thru	with	

Appendix B: List of Academic Paper Titles and 5 Reference Papers from Each Academic Papers Used in Experiment 1

Index	Academic Paper Title	Authors	References
1	Scene-Adaptive Transform Domain Video Partitioning	Donald A. Adjeroh, M. C. Lee	3, 6, 7, 10, 13
2	Comparing Partition and Random Testing via Majorization and Schur Functions	Bojan Cukic, Philip J. Boland, Harshinder Singh	2, 4, 5, 7, 9
3	Architectural-Level Risk Analysis Using UML	Hany Ammar, Katerina Goseva-Popstojanova, Ahmed Hassan, Ajith Guedem, Walid Abdelmoez, Diaa Eldin M. Nassar, Ali Mili	1, 6, 11, 30, 31
4	Architecture-Based Software Reliability: Why Only a Few Parameters Matter?	Katerina Goseva-Popstojanova, Margaret Hamill	1, 2, 5, 9, 10
5	Empirical Capacity of a Recognition Channel for Single and Multi-Pose Object Recognition under the Constraint of PCA Encoding	Natalia A. Schmid, Xiaohan Chen	1, 10, 2, 3, 7
6	Joint Power Allocation and Relay Selection for Multiuser Cooperative Communication	Daryl Reynolds, Kanchan Vardhe, Brian Woerner	1, 2, 3, 4, 5
7	Generating Synthetic Irises by Feature Agglomeration	Arun Ross, Samir Shah	1, 2, 3, 4, 5
8	The Strangest Thing About Software	Tim Menzies, David Owen, Julian Richardson	16, 7, 3, 4, 19
9	Asynchronous Cooperative Diversity	Matthew C. Valenti, Shuangqing Wei, Dennis L. Goeckel	1, 2, 3, 4, 5

Bibliography

[1] Q. Wu, E. Fuller and C. Zhang. (2009), Text document classification and pattern recognition. *Social Network Analysis and Mining, International Conference on Advances in 0*pp. 405-410.

[2] Q. Wu, E. Fuller and C. Zhang. (2010), Graph model for pattern recognition in text. *V.288*pp. 1-20.

[3] Wikipedia. (2010), Stemming --- wikipedia, the free encyclopedia. Available: http://en.wikipedia.org/w/index.php?title=Stemming&oldid=394152632.

[4] M. F. Porter. (1980, 07). An algorithm for suffix stripping. *Program 14(3),* pp. 130-7.

[5] P. M. Parker, *Han: Webster's Quotations, Facts and Phrases.* 2008.

[6] C. Fox. (1990), A stop list for general text. *SIGIR Forum 24(1-2),* pp. 19-35.

[7] A. Chudnovsky. (2008, August 8, 2008). C# HTML parser (.NET).

[8] D. D. Lewis, Y. Yang, T. G. Rose and F. Li. (2004), RCV1: A new benchmark collection for text categorization research. *J.Mach.Learn.Res. 5*pp. 361-397.

[9] S. Zhong. Efficient online spherical k-means clustering. Presented at Proceedings of the International Joint Conference on Neural Networks 2005.

[10] Wikipedia. (2010), Cluster analysis --- wikipedia, the free encyclopedia. Available: http://en.wikipedia.org/w/index.php?title=Cluster_analysis&oldid=395119009.

[11] S. Weiss, N. Indurkhya, T. Zhang and F. Damerau, *Text Mining - Predictive Methods for Analyzing Unstructured Information.* Springer Science+Business Media Inc., 2005.

[12] G. Salton. (1969, 01). Information storage and retrieval. Cornell Univ., Ithaca, NY, USA. USA.

[13] C. J. Van Rijsbergen. (1975), *Information Retrieval*.

[14] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester and R. Harshmann. (1988), Using latent semantic analysis to improve access to textual information. *SIGCSE Bull 20*pp. 281-281.

[15] S. R. Peddada. (2010), Sensitivity of semantic signatures in text mining.

[16] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten. (2009), The WEKA data mining software: An update. *SIGKDD Explor.Newsl. 11(1),* pp. 10-18. Available: http://dx.doi.org/10.1145/1656274.1656278.

[17] G. Salton and M. J. McGill. (1983), *Introduction to Modern Information Retrieval*.

[18] S. Sitarama, U. Mahadevan and M. Abrol. Efficient cluster representation in similar document search. Presented at Efficient Cluster Representation in Similar Document Search.

[19] Eui-Hong Han and G. Karypis. Centroid-based document classification: Analysis and experimental results. Presented at 4th European Conference, PKDD 2000.

[20] C. D. Manning, P. Raghavan and H. Schutze, "Evaluation in information retrieval," in *Introduction to Information Retrieval*Anonymous 2008, pp. 139.