

2011

Automated Discovery of Relevant Features for Text Mining

Rukmini Ravali Kota
West Virginia University

Follow this and additional works at: <https://researchrepository.wvu.edu/etd>

Recommended Citation

Kota, Rukmini Ravali, "Automated Discovery of Relevant Features for Text Mining" (2011). *Graduate Theses, Dissertations, and Problem Reports*. 4742.
<https://researchrepository.wvu.edu/etd/4742>

This Thesis is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/ or on the work itself. This Thesis has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.

Automated Discovery of Relevant Features for Text Mining

Rukmini Ravali Kota

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
Computer Science

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

2011

Keywords: Text Mining, Semantic Signatures, Automation, Document Retrieval

Copyright 2011 Rukmini Ravali Kota

Abstract

Automated Discovery of Relevant Features for Text Mining

Rukmini Ravali Kota

Text mining refers to the process of extracting information from text. There are massive amounts of data available today due to enhanced data collection capabilities, inexpensive high capacity storage, and the proliferation of World Wide Web pages. A substantial portion of this data is in text format. The main goal of text data mining software tools is to help us learn and benefit from this wealth of text data. Humans cannot cope with the overwhelming text data resources. The information in text data needs to be filtered, summarized, analyzed, and refined for human analysts.

A *semantic signature* is the concept that semantic content in text has characteristic word patterns, such as frequency of words and proximity between words, which can be identified and quantified. A type of quantitative semantic signature was developed by Barnes, Eschen, Para, and Peddada in 2010. The utility and sensitivity of semantic signatures of this type in capturing semantic content in text data was demonstrated by this group via the development of a software package named Semantic Signature Mining Tool (SSMinT). SSMiT is a suite of software tools that assist a data analyst to develop semantic signatures that capture targeted content and then use these semantic signatures to categorize a corpus of text documents with unknown content or to retrieve text documents with the targeted content from a corpus of documents with arbitrary content.

Key features of SSMiT are the expert input from the human analyst and the interaction between the analyst and the software; the tool is designed to assist the analyst and does not work independently. This is a strong feature in the sense that the resulting semantic signatures are tailored by the analyst's expert knowledge of the domain. This was demonstrated by Barnes, Eschen, Para, and Peddada to be a powerful approach to text data mining.

This thesis develops an *automated* version of the SSMiT software package that requires minimal input from an analyst. This work includes an automated keyword group generation and refinement algorithm, automated generation of candidate semantic signatures, methods to prune

irrelevant and redundant relevant semantic signatures from the semantic signature set. Relieving the analyst from the tedious and time consuming task of developing semantic signatures is not the only motivation for an automated tool. The automation is designed to *discover* semantic signatures in text data without human input, except for the choice of training documents. The advantage of automated semantic signature discovery is the ability to identify patterns an analyst may not recognize due to the large volume of data or his point of view bias. The effectiveness of Automated SSMInT in categorizing text documents into groups with closely related content and retrieving documents with content similar to those in its training set is demonstrated in experiments on various corpora. These experiments prove Automated SSMInT to be an efficient, convenient, and powerful text mining tool.

Dedication

To My Uncle

Kota Venkata Harinath

&

To My Advisors

(Dr. Eschen & Dr. Barnes)

Acknowledgments

I would like to thank my advisors Dr. Eschen and Dr. Barnes as my thesis wouldn't have been possible without their support, attention to detail and guidance during my research. Special thanks are due to Dr. Eschen for making me a part of Discrete Algorithms Research Team. It gives me immense pleasure to thank my family especially my parents and my sister who played a major role behind me pursuing my Masters at West Virginia University. Thank you for everything dad! Love you loads!!! 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. Thanks to my student colleagues Uday, Ramya, Kel and Lingaiah for their support and help. I would like to thank all my friends and LCSEE department for the support and encouragement.

Table of Contents

Abstract	ii
Dedication.....	iv
Acknowledgments	v
List of Figures	ix
List of Tables.....	xi
List of Symbols / Nomenclature.....	xii
Chapter 1: Introduction.....	1
1.1 Motivation	1
1.1.1 Identification of the problem	1
1.2 Contribution of my thesis	2
1.3 Flow of the document.....	2
Chapter 2: Background Concepts.....	3
Sample text.....	3
Documents.....	3
Preprocessing	3
Concept of window	4
Window weight function.....	6
Term Frequency-Inverse Document Frequency.....	6
Point back.....	7
Euclidean Distance Measure	8
Cosine Similarity Measure	8
K-Means Clustering	9
Semantic Signature.....	9
Cluster Definitions	10
Singular Value Decomposition	11
WEKA.....	12
Chapter 3: Analyst Driven SSMInT.....	13
Keyword Tool	13

Learner Tool.....	15
Data Analysis Tool.....	16
Chapter 4: Automated Semantic Signature Mining Tools.....	18
4.1 Overview	18
Introduction to Automated Keyword Tool	19
Introduction to Automated Learner Tool.....	20
Introduction to Hits Array Generator Tool.....	21
Introduction to Semantic Signature Refinement Tool.....	22
4.2 Automated Keyword Tool.....	23
4.2.1 Keyword group generation algorithm.....	24
4.2.2 Working of Automated Keyword Tool	25
4.2.3 Output of Automated Keyword Tool.....	26
4.2.4 Comparisons between the Analyst Driven and Automated Keyword Tool	27
4.3 Automated Learner Tool	28
4.3.1 Document vector refinement	28
4.3.2 Ways of selecting clusters as semantic signatures.....	29
4.3.2.1 Save cluster with more component weight	29
4.3.2.2 Save all Clusters.....	29
4.3.3 Working of Automated Learner Tool.....	29
4.3.4 Output of Automated Learner Tool	31
4.3.5 Comparisons between the manual and automated versions of Learner Tool.....	33
4.4 Hits Array Generator Tool	33
4.4.1 Calculation of <i>hits</i>	33
4.4.2 Working of Hits Array Generator Tool	35
4.4.3 Output of Hits Array Generator Tool	35
4.5 Semantic Signature Refinement Tool.....	36

4.5.1 Refinement of semantic signatures.....	38
4.5.1.1 Role of Reduced Singular Value Decomposition in SSRT	38
4.5.2 Dimensionality Reduction	39
4.5.3 Cluster the hits array using K-Means	39
4.5.4 Working of Semantic Signature Refinement Tool	39
4.5.5 Output of Semantic Signature Refinement Tool	40
Chapter 5: Experiments.....	41
5.1.1 Goal of the experiment.....	41
5.1.2 Corpus Used.....	41
5.1.3 Design of Experiment.....	41
5.1.4 Results	43
5.1.5 Interpretation	43
5.2.1 Goal of the experiment.....	44
5.2.2 Corpus Used	44
5.2.3 Design of Experiment.....	44
5.2.4 Results	46
5.2.5 Interpretation.....	48
5.3.1 Goal of the experiment	48
5.3.2 Corpus Used.....	48
5.3.3 Design of Experiment	48
5.3.4 Results.....	50
5.3.5 Interpretation.....	50
Chapter 6: Conclusions and Future Work.....	51
Appendix A: List of Stop Words.....	53
Bibliography.....	57

List of Figures

Figure2.1 Text extracted from a paper on throat cancer [18]	3
Figure2.2 Phases of Preprocessing	5
Figure2.3 Representation of window	6
Figure2.4 Point-back option in Analyst Driven Keyword Tool	8
Figure2.5 Sample text with windows (highlighted in red) and keywords (highlighted in green)	10
Figure3.1 Input and Output of the tools in Analyst-driven SSMInT Package	13
Figure3.2 Human interaction with Keyword Tool. Inputs, Outputs and Functions of Keyword Tool	14
Figure3.3 Human interaction with Learner Tool. Inputs, Outputs and Functions of Learner Tool	15
Figure3.4 Inputs, Outputs and Functions of Data Analysis Tool	16
Figure3.5 Overview of document-semantic signature matrix processing	17
Figure4.1 Input and output of the tools in Automated SSMInT	19
Figure 4.2 Automated Keyword Tool working intelligently to generate keyword groups	20
Figure 4.3 Functions of Automated Keyword Tool	20
Figure 4.4 Automated Learner Tool working intelligently to refine vectors and select clusters .	21
Figure 4.5 Functions of Automated Learner Tool	21
Figure 4.6 Hits Array Generator Tool working on generating the document-semantic signature matrix	22
Figure 4.7 Functions of Hits Array Generator Tool	22
Figure 4.10 Working of Automated Keyword Tool	24

<u>Figure 4.11 Screenshot of Automated Keyword Tool</u>	26
<u>Figure 4.12 Structure of Keyword Descriptor File</u>	27
<u>Figure 4.13 Working of Automated Learner Tool</u>	30
<u>Figure 4.14 Screenshot of Automated Learner Tool</u>	31
<u>Figure 4.15 Structure of Semantic Signature Descriptor File</u>	32
<u>Figure 4.16 Working of Hits Array Generator Tool</u>	34
<u>Figure 4.17 Screenshot of Hits Array Generator Tool</u>	35
<u>Figure 4.18 Structure of Hits Array Descriptor File</u>	36
<u>Figure 4.19 Working of Semantic Signature Refinement Tool</u>	37
<u>Figure 4.20 Structure of Attribute Relation File Format</u>	40
<u>Figure 5.1 Output of Cobweb clustering on the testing corpus</u>	47
<u>Figure 5.2 Output of Cobweb clustering when the keywords <i>singing</i> and <i>cancer</i> are ignored</u>	47

List of Tables

<u>Table 2.1 Term Frequency and Inverse Document Frequency scores for the words <i>drinking</i> and <i>risk</i></u>	7
<u>Table 2.2 showing the occurrence of the keywords and their position in the sample text</u>	11
<u>Table 2.3 Window index table indicating the starting and ending indexes of the active windows of the sample text</u>	11
<u>Table 2.4 Aggregated weights of keyword interactions in an active window</u>	11
<u>Table 4.1 Representation of document vector in matrix format</u>	28
<u>Table 5.1 Abbreviations for professor’s names whose papers are used in this experiment</u>	41
<u>Table 5.2 Inputs and outputs of the training tools of Automated SSMinT</u>	42
<u>Table 5.3 Inputs and outputs of the testing tools of Automated SSMinT</u>	42
<u>Table 5.4 Semantic feature vector clustering results with Spherical K-means</u>	43
<u>Table 5.5 Groupings of documents according to the topic</u>	43
<u>Table 5.6 Inputs and outputs of the training tools of Automated SSMinT</u>	45
<u>Table 5.7 Inputs and outputs of the testing tools of Automated SSMinT</u>	45
<u>Table 5.8 Inputs and outputs of the training tools of Automated SSMinT</u>	46
<u>Table 5.9 Inputs and outputs of the testing tools of Automated SSMinT</u>	46
<u>Table 5.10 Inputs and outputs of the training tools of Automated SSMinT</u>	49
<u>Table 5.11 Inputs and outputs of the testing tools of Automated SSMinT</u>	49
<u>Table 5.12 Semantic feature vector clustering results with Spherical K-means</u>	50

List of Symbols / Nomenclature

1. KGG – Keyword Group Generation
2. KDF – Keyword Descriptor File
3. SSDF – Semantic Signature Descriptor File
4. HADF – Hits Array Descriptor File
5. ARFF – Attribute Relation File Format
6. AKT – Automated Keyword Tool
7. ALT – Automated Learner Tool
8. HAGT – Hits Array Generator Tool
9. SSRT – Semantic Signature Refinement Tool
10. CD– Cluster Definition
11. HTML – Hyper Text Markup Language
12. XML – Extensible Markup Language
13. RCV1 – Reuters Corpus Volume 1

Chapter 1: Introduction

Text mining is a process of deriving potentially useful knowledge from text [1]. The difference between data mining and text mining, as quoted by Marti Hearst, is that data mining tries to extract patterns from structured *databases* which are designed for programs to process automatically whereas text mining processes natural unstructured *text* which is written for people to read [1, 2]. Text mining differs from data mining in the sense that the information to be extracted is explicitly stated in the document. Substantial amount of unstructured data exists in text format [4]. Even though it is explicitly stated, it is infeasible to read the full text and extract information from it. Now-a-days, a lot of unstructured text is available in digital form, thanks to the low memory storage costs. As a result, it is becoming difficult to mine information from the data even if the analyst is aided by a computer. Generally, an analyst also makes a decision by observing a pattern. So, instead of mere automation, if we are able to train the tools to take a knowledge-driven decision, text mining would be an achievable goal.

One major task of text mining is document clustering. Targeting documents containing content of interest from a large pool of corpus containing documents of unknown content is the need of the hour. Many techniques have been developed to aid us in achieving this goal.

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 [26] [27]. 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. This approach has proven to be a remarkably sensitive tool for differentiating semantic content in text data. This thesis presents an automated technique which allows us to group documents of similar content by semantic signature discovery or retrieve documents of a targeted content from an arbitrary corpus with minimal input from the analyst.

1.1 Motivation

1.1.1 Identification of the problem

Semantic Signature Mining Tool (SSMinT) consists of three tools namely Keyword Tool, Learner Tool and Data Analysis Tool developed by Barnes, Eschen, Para [5] and Peddada [6]. This sequence of tools analyzes a corpus and allows an analyst to design semantic signatures with precision, utilizing expert knowledge, that capture targeted content. This works fine, except that it is tedious and laborious requiring great attention from the analyst. Also, the learning documents may contain semantic signatures which the analyst does not recognize which can be attributed to his bias knowledge. (Expert knowledge may be biased).

A computer analyzes a document in a different way compared to an analyst. Also, a computer can do mundane/ repeated tasks with same level of efficiency from start to end

compared to a human analyst whose performance may have an effect on the efficiency as time progresses.

The motivation to automation is to *discover* semantic signatures in text data with minimal analyst input and also to identify patterns an analyst may not recognize due to the large volume of data or his point of view bias. “Intelligent” automation not only helps in semantic signature discovery but also helps in saving the time of the analyst as the work load would be reduced because the tools will be able to produce the desired output by taking decisions with minimal intervention from him. It helps the analyst in discovering the semantic signatures which were previously unknown. This saves money too! The motivation to automate the tools is high even if the success is only partial as automation helps in saving time and money. So, the thought was to automate the existing tools in SSMInT by preserving their efficiency, even though there might be a decrease in the semantic signature’s precision.

1.2 Contribution of my thesis

The contribution of my thesis can be summarized as follows

- The automation of tools in Analyst Driven SSMInT.
- Inclusion of Term Frequency – Inverse Document Frequency to aid in the process of keyword group generation in Automated Keyword Tool.
- Heuristics for retaining vectors with significant content in Automated Learner Tool.
- Overcoming redundant relevant data in Semantic Signature Refinement Tool.

1.3 Flow of the document

Chapter 2 presents the overview of text mining. It covers the basics of already established concepts which are widely used in the field of data mining and are referred in this thesis.

Chapter 3 introduces Analyst Driven SSMInT by presenting an overview of the suite of tools developed by Barnes, Eschen, Para [5] and Peddada [6]. The underlying concepts used in the development of those tools are presented in a detailed manner.

Chapter 4 begins with the outline of the suite of automated tools developed by me under the guidance of Barnes and Eschen. In-depth functioning of the suite is discussed along with the concepts which played a crucial role in building the software package.

Chapter 5 deals with the experimental set up which is needed to benchmark the tools developed as a part of my thesis. Tools are exposed to a variety of corpora in order to test their efficiency. Several experiments are done varying the underlying criteria in order to test whether the goal of refinement was achieved or not. Three experiments are presented in this chapter. The results obtained are also discussed in detail. By analyzing the results, one can understand the power and performance of the automated tools.

Chapter 6 presents an overview of the conclusions derived from my thesis leaving scope for future work by discussing the areas where there are still chances for improvements which when done may further refine the results.

Chapter 2: Background Concepts

This chapter provides an overview of underlying concepts in this thesis. This chapter not only provides an overview of existing and popular data mining concepts, but also introduces new concepts and terminology used in this thesis. A brief idea of these concepts would ensure a better understanding of the further chapters. The sample text presented below is used to demonstrate the techniques developed.

Sample text

Women appeared more knowledgeable about the health risks of heavy drinking than men. It is important to note, however, that "heavy alcohol drinking" was not defined for the survey respondents. On risks of throat cancer and cancer of the mouth, less than half of either the men or the women knew that heavy drinking increased one's chances of developing these cancers. However, more than 90 percent of the men and women knew of the increased risk of liver cirrhosis from heavy drinking.

Figure2.1 Text extracted from a paper on throat cancer [18]

Documents

A document, in this thesis refers to a piece of text stored in digital format. It may contain any type of unstructured text. In the experiments presented, a document may refer to an article from Reuters corpus [24] or a research paper of a Professor or a paper published on a topic.

Preprocessing

Preprocessing is a basic and important data mining task, which if neglected may add noise to the data. This may result in producing misleading results as noise deteriorates the power of the tools. This also makes the knowledge discovery from the training set of documents harder [32]. Data preprocessing includes but is not limited to cleaning, normalization, transformation, feature selection and feature extraction [33]. The output of the data preprocessing is more refined as it undergoes basic filtering, which filters out unwanted data. This step makes training data ready to undergo text mining operations.

In order to filter out unwanted data from the input which is fed to the tools in SSMInT, the training documents undergo preprocessing. These documents may be in different formats. So, the first step in data preprocessing is to extract text from the training documents using the corresponding parser. Then white spaces and stop words are removed. Synonyms Substitution, Phrase Replacement and Stemming can be performed on the training document.

The raw text along with synonyms, phrases (if any) and stemming (enabled or disabled) is given as input to the preprocessor which preprocesses it as follows:

- The format of the training document is identified and the corresponding parser is used to extract the text from the document.
- The entire text is taken as a single string and then it is broken into words wherever it encounters a space. Thus, a ‘list of words’ is generated.
- White space, which is a single or series of characters which represent a horizontal or vertical space in typography, is removed. They do not have a visible output character. Even though it doesn’t leave a visual mark, it occupies area of the page which is of no use. Hence all white spaces are removed while preprocessing.
- *Stop words* is a term used for the words which carry no weight or which add noise in finding semantic signatures [10]. These words, if not filtered, may increase the significance of common words that do not carry content which is undesirable. The Stop word list is language specific. A list of English language stop words [6], which is presented in Appendix A. A word is removed from the ‘list of words’, if it matches to a word in the stop word list.
- Several words may mean the same or almost the same. These are called synonyms. In the figurative sense, two words are often said to be synonymous if they have the same connotation. So, instead of considering them as different words, if they can be recognized as same words, the weight of that word increases in document processing. Synonym Substitution is done if the analyst inputs words and their synonyms when prompted by the tool.
- When two or more words are considered as a single word, it is called a phrase. While preprocessing, Phrase Replacement is necessary to capture the desired semantic signature, where the targeted content can be embodied in more than just one word. For example, consider a scenario where we are trying to extract the semantic signatures embodying the content of “black market”. If we search for “black” and “market” in the document separately, we would end up with a semantic signature which is undesired because considering them as two separate words would give an entirely different meaning to it. Instead, we want to search for “black market” considering it as a phrase. To serve this purpose, an option is given in Keyword Tool where multiple words are joined as a phrase if desired by the analyst.
- Stemming is a process of reducing inflected words to their base form. A Stemmer for English, for example, should reduce the words “functioned”, “functions” and “functioning” to “function” [7]. This is done to ensure weight to words. Stemming is an option in Automated Keyword Tool. If stemming is set, each word is stemmed to roots. The stemmer implemented here is the Porter Stemmer [9].

Concept of window

A window is described as a group of words starting from a keyword. The window length is count of words in a window. Experiments were done varying the length of the window. After some trials, the window length is fixed to be 20. However, there is an option in the developed tools to change the window length if needed. In the sample text presented in Figure2.3, if we choose *drinking* as a keyword (highlighted in green), the window of length 20 is the one highlighted in red. A window is said to be *active* if there appears more than one keyword in that window.

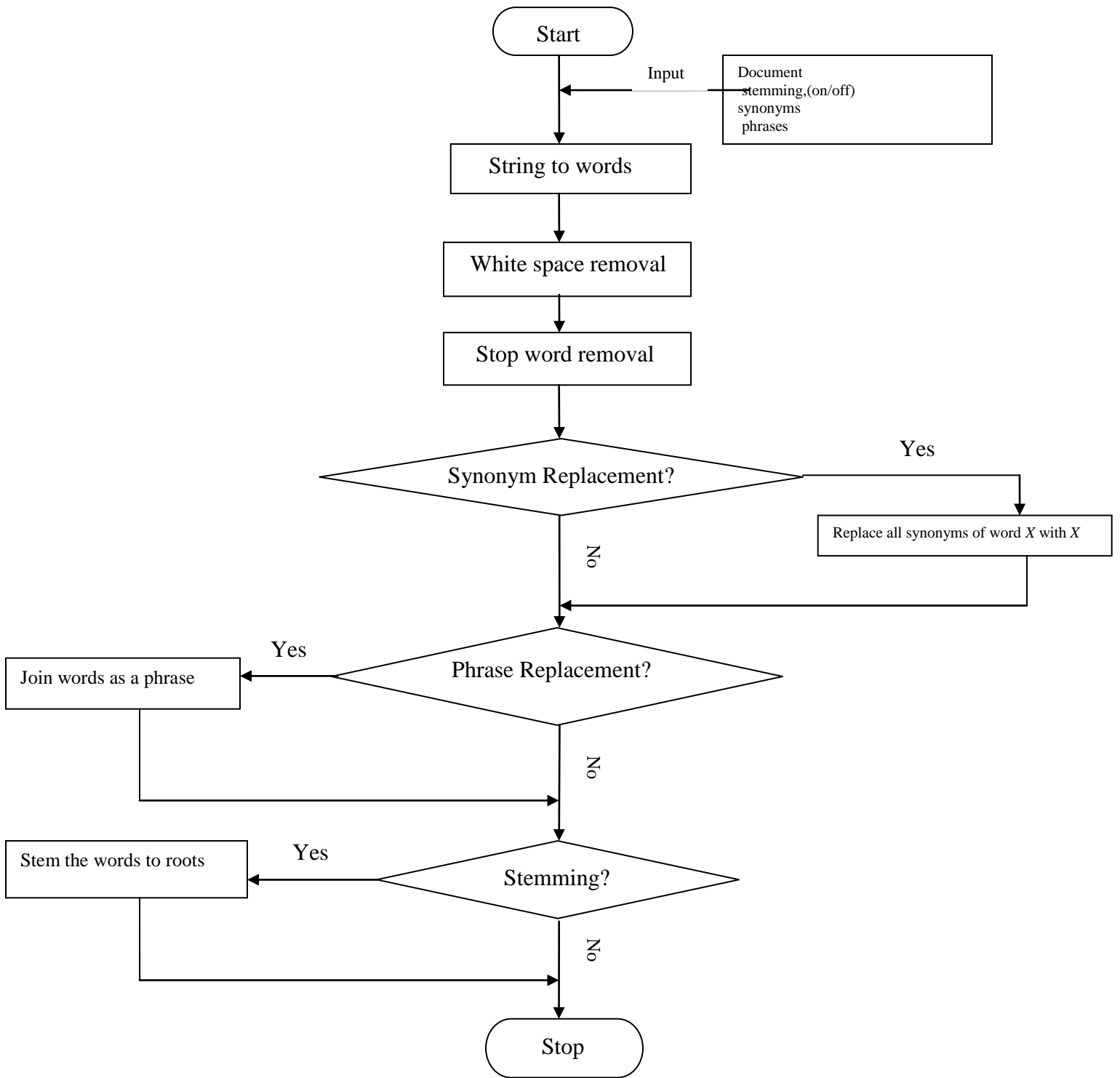


Figure2.2 Phases of Preprocessing

Women appeared more knowledgeable about the health risks of heavy drinking than men. It is important to note, however, that "heavy alcohol drinking" was not defined for the survey respondents. On risks of throat cancer and cancer of the mouth, less than half of either the men or the women knew that heavy drinking increased one's chances of developing these cancers. However, more than 90 percent of the men and women knew of the increased risk of liver cirrhosis from heavy drinking.

Figure2.3 Representation of window

Window weight function

The weighting function between a keyword and a word x is defined to be

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

where “a” is a constant and “d” is the number of words between the keyword and x . In the window highlighted in Figure2.2, the weighting function between the keyword *drinking* and the word *alcohol* is calculated as

$$w(\text{alcohol}) = \sqrt{\frac{5^2}{5^2 + 4^2}} = 0.781$$

Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency (TF-IDF) is widely used in document retrieval. This weight is a statistical measure, which is used to assess the importance of a word to a document in a corpus [8]. Term Frequency is the weight given to each distinct term in the given document based on the number of occurrences of that term [3]. If only term frequency is considered, a term may get more weight if it occurs frequently. Inverse Document Frequency (IDF) is a measurement of general importance of the term. It attenuates the weight of the frequently occurring terms over the weight of the terms that occur rarely [8].

By multiplying TF and IDF, the weights tend to filter out the common terms and determine the terms relevant for a given document. A term is assigned a higher weight if it appears a larger number of times in the document and a smaller number of times in the corpus, compared to another term which appeared a larger number of times in the document as well as in the corpus.

By attenuating the weight of the frequently-appearing terms in the corpus, TF-IDF can act as a good filter for stop words, pronouns, prepositions and articles irrespective of the subject area targeted for mining even if they are not known in advance. It is very widely used in search engines and information retrieval systems because of its simplicity, effectiveness and advantages.

Equations behind TF-IDF

Term Frequency, which is the weight given to each distinct term in the given document based on the number of occurrences of that term, is calculated as:

$$TF(t, d) = \text{No. of times the term 't' appears in the document 'd'} / \text{No. of terms in the document.}$$

Inverse Document Frequency, which is the measure of the importance of the term attenuating the weight of the frequently occurring terms over the weight of the terms that occur rarely, is calculated as:

$$IDF(t) = \log \frac{|D|}{|\{d: t \in D\}|}$$

TF-IDF weight is the product of TF and IDF for a term in the document

Example: The text presented in Figure 2.1 is used in the Term Frequency calculations. For calculating Inverse Document Frequency, suppose that, 10000 documents related to the topic *Health* are taken. The document from which Term Frequency calculations are made is also related to the topic *Health* but in particular talks about the health risks due to alcohol consumption. The words *drinking* and *risk* appear four times and three times respectively in the document which is used to calculate the Term Frequency. The total number of terms in that document, eliminating the stop words, is 45. In the corpus from which Inverse Document Frequency calculations are made, *drinking* and *risk* appear in 100 and 5000 documents respectively.

Table 2.1 Term Frequency and Inverse Document Frequency scores for the words *drinking* and *risk*

	Term Frequency	Inverse Document Frequency	TF-IDF score
<i>drinking</i>	0.089	2	0.18
<i>risk</i>	0.067	0.3	0.02

The term *drinking* records a higher score than the term *risk* making it clear that the term *drinking* is more important to the document compared to the term *risk*.

Point back

This is an option provided in Analyst Driven Keyword and Learner tools. In Keyword Tool, for a given word, it highlights all the occurrences of that word in the input document which helps the analyst to understand the context so as to make a decision whether the word can be selected as a keyword or not. Similarly in Learner tool, by pointing to the document from which the vector is generated, point-back helps the analyst to determine whether the cluster is embodying the content or not. A vector in this case represents the array of weights of interactions of the keywords within themselves and within a group.

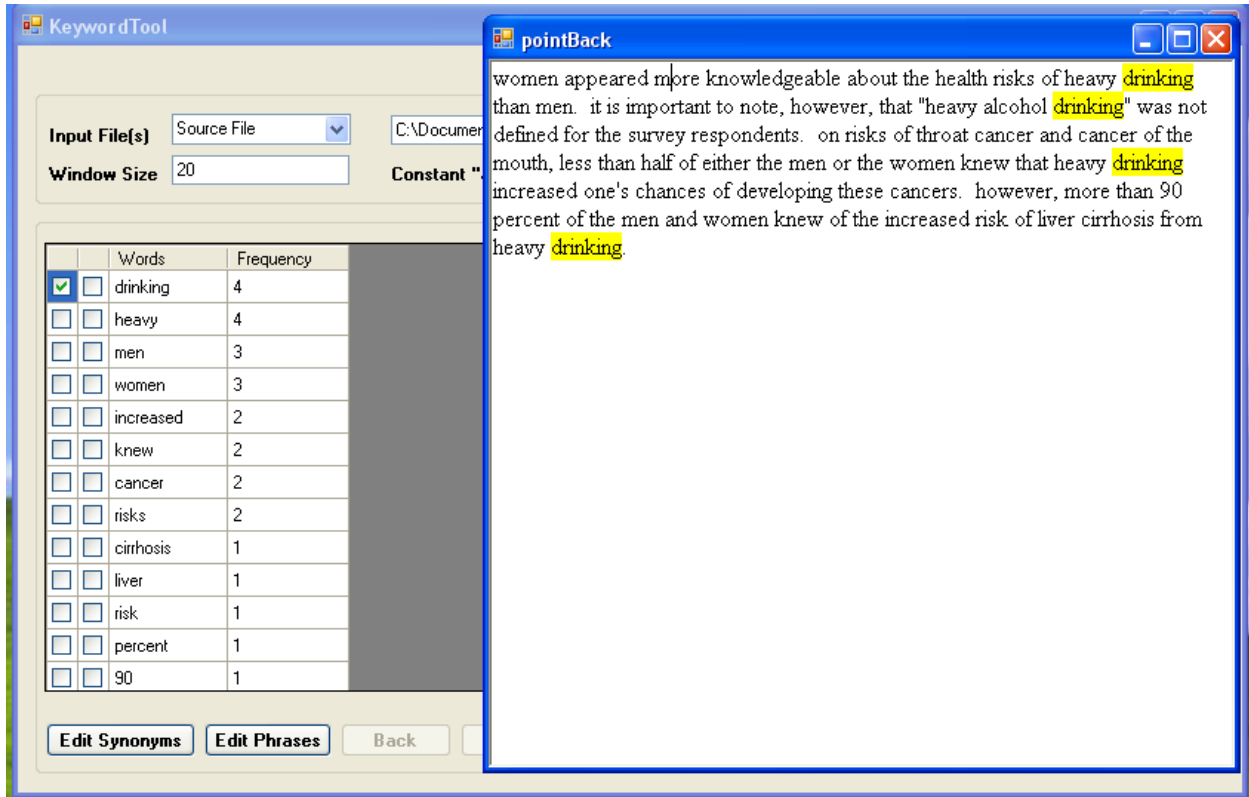


Figure 2.4 Point-back option in Analyst Driven Keyword Tool

Euclidean Distance Measure

Euclidean Distance Measure calculates the distance between two vectors by taking the square root of the sum of squares of the difference between the corresponding coordinates of that vector [17]. For two vectors A and B, having ‘n’ dimensions, the Euclidean distance measure is calculated as:

$$d(A, B) = \sqrt{(A_1 - B_1)^2 + (A_2 - B_2)^2 + (A_3 - B_3)^2 + \dots + (A_n - B_n)^2}$$

. Here A_n and B_n are the coordinates of the vector A and B in the n^{th} dimension.

Cosine Similarity Measure

Cosine Similarity measures the similarity between two vectors by calculating the cosine of the angle between them [18]. This helps in finding out whether the two vectors are pointing in the same direction or not. For two vectors A and B, the cosine similarity, $\cos \theta$, is calculated as:

$$\cos \theta = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$

In n-dimension space, a vector A is said to be similar to another vector B, if the $\cos \theta$ between A and B is more compared to A and other vectors. In the experiments presented, only

the cosine similarity measure is used because it outperformed all other measures in Para's thesis [5]. Compared to Euclidean distance, cosine has the advantage of not being affected by one document being larger than other.

K-Means Clustering

K-Means clustering is one of the most popular clustering technique used in Data Mining. Its wide use can be attributed to its simplicity and ease of applicability even to large data sets. The goal of K-Means clustering is to partition the input vectors into predefined number of clusters where each vector belongs to a cluster with nearest mean. To achieve this goal, it first randomly selects 'k' number of vectors which act as centroids for the initial iteration. The other vectors are then allowed to be grouped into the nearest cluster [28]. This grouping depends on the distance measure calculations. In the experiments presented, I used cosine similarity measure which groups a vector into a cluster if the cosine similarity between that vector and the centroid of the cluster is greater when compared to the cosine similarity of that vector to the centroid of other clusters. As the cosine similarity measure is used for grouping vectors in a cluster, it is called Spherical K-Means. After the initial grouping, update the centroid to be the mean vector of each cluster. Again, the same process is repeated and the vectors move into different clusters if this vector is similar to the centroid of that cluster compared to other clusters.

The implemented K-Means algorithm [20] in brief can be described as:

- Randomly generate 'k' centroids for 'k' clusters.
- Assign each vector to the nearest cluster using the distance measure calculations.
- Compute mean vector of each cluster.
- Update the clusters by moving each vector to the cluster whose mean vector is similar to this vector when compared to other vectors.
- Repeat the above steps till the algorithm converges vectors in the bins are stable or the number of maximum iterations has reached.

As it is a heuristic algorithm, it may not converge and hence there should be an upper limit on the number of times this process repeats so that it will not end up in an indefinite loop. Even though clusters are not stable, all the cluster assignments after a minimum number of iterations make sense. This algorithm is highly sensitive to the initial selection of seed cluster and hence should be repeated multiple times. In simple K-Means, the number of clusters should be determined earlier. This helps in presenting the semantic sensitivity of the developed tools.

Semantic Signature

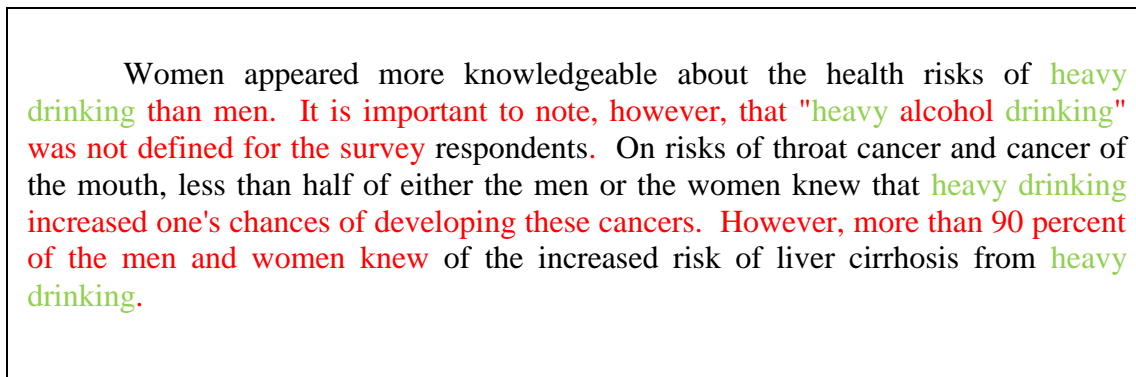
Semantic Signature refers to a group of document vectors extracted from the training documents which embodies the target content in it. A vector in this case represents the array of weights of interactions of the keywords within themselves and within a group. A key step in the design of semantic signatures is the generation of vectors.

Document Vector Generation

A document vector represents the array of weights of interactions of the keywords within themselves and within a group. First, active windows are to be identified in the training document. To serve this purpose, a table is generated which documents the keyword's appearances with their position in the training document.

Table 2.2 lists the appearances of the keywords *heavy* (keyword number 0) and *drinking* (keyword number 1) in the sample text. Then, a window index table will be generated which records the starting and ending index of all the active windows in the document. After identifying the active windows, weights are calculated for the keywords using the window weight function. If a keyword appears more than one time in the window, the aggregate weight function would be normalized by the number of occurrences. While generating vectors, weights are calculated only in forward direction so as to include each distance only once. Table 2.4 shows the interactions between the selected keywords in a matrix format. These weights when written side by side in a vector format, a document vector is generated. For each active window, a document vector will be generated.

The generated document vectors are then clustered using K-Means Algorithm which groups the vectors with similar orientation together. A cluster, which appears to be a better representation of the target content, is selected by the analyst with the help of the point-back tool. This cluster is called the semantic signature.



Women appeared more knowledgeable about the health risks of heavy drinking than men. It is important to note, however, that "heavy alcohol drinking" was not defined for the survey respondents. On risks of throat cancer and cancer of the mouth, less than half of either the men or the women knew that heavy drinking increased one's chances of developing these cancers. However, more than 90 percent of the men and women knew of the increased risk of liver cirrhosis from heavy drinking.

Figure 2.5 Sample text with windows (highlighted in red) and keywords (highlighted in green)

Cluster Definitions

The clusters obtained in Learner Tool may be defined in three different ways:

Cluster Definition 1 defines a cluster with the centroid and a measure which is the Euclidean distance between the centroid and the farthest vector in the cluster. [22].

Cluster Definition 2 defines a cluster with a centroid vector and a measure which is the cosine distance (angle) between the centroid vector and the farthest vector in the cluster.

Cluster Definition 3 is defined by all the vectors in the cluster.

Table 2.2 showing the occurrence of the keywords and their position in the sample text

Index	Keyword Number	Position of the keyword in the sample text
1	0	9
2	1	10
3	0	20
4	1	22
5	0	52
6	1	53
7	0	81
8	1	82

Table 2.3 Window index table indicating the starting and ending indexes of the active windows of the sample text

Index	Starting index of the window	Ending index of the window
1	1	4
2	5	6
3	7	8

Table 2.4 Aggregated weights of keyword interactions in an active window

	<i>heavy</i>	<i>drinking</i>
<i>heavy</i>	0.45	0.79
<i>drinking</i>	0.49	0.41

Singular Value Decomposition

Singular Value Decomposition (SVD) of M is defined as the factorization of M into the form $U\Sigma V^t$ where M is a $m \times n$ real or complex matrix, U is a $m \times m$ real or complex unitary matrix, Σ is a $m \times n$ diagonal matrix and V^t is a $n \times n$ real or complex unitary matrix [29].

$$M_{[m \times n]} = U_{[m \times m]} \Sigma_{[m \times n]} V^t_{[n \times n]}$$

Dimensionality reduction is the transformation of high-dimensional data into a meaningful representation of reduced dimensionality [13]. It is done to refine out irrelevant dimensions which carry no or less information. Removal of noisy dimensions will help improve the accuracy. In order to select a subset of features that are relevant, we need a mechanism to score every feature. SVD scores each feature based on the importance of that feature in the solution space. This scoring is considered in Semantic Signature Refinement Tool which is a tool in Automated SSMInT which refines irrelevant Semantic Signatures.

WEKA

WEKA, an acronym for Waikato Environment for Knowledge Analysis, is a free and popular suite of machine learning software written in java, developed by University of Waikato, New Zealand [34, 14]. It supports several data mining tasks such as preprocessing, clustering, classification, regression, visualization and feature selection. It only accepts data which is in Attribute Relationship File Format, in short ARFF. More about WEKA can be found at [34, 14]

The output of SSMInT is in ARFF so that various data mining tasks can be tried over it using WEKA which opens the scope for further/better interpretation of results.

Chapter 3: Analyst Driven SSMInT

This chapter provides a brief overview of the tools contained in Analyst Driven SSMInT and the motivation to automate the package. This package was initially developed by Barnes, Eschen, Para and Peddada. Refer to Para's [5] and Peddada's [6] theses for in-depth details on Analyst Driven SSMInT.

Tools in Analyst Driven SSMInT have demonstrated their capabilities of clustering the corpus of documents with unknown content according to the semantic content. Also, they can retrieve documents containing targeted content from the corpus of documents with unknown content. Experiments are done on various corpora to test the document classification capability and the semantic sensitivity of the developed tools. All the tools in SSMInT are stand-alone. Also, in SSMInT, second tool's input is designed to accept and understand the first tool's output format and similarly the third tool's input is designed to accept and understand the second tool's output format.

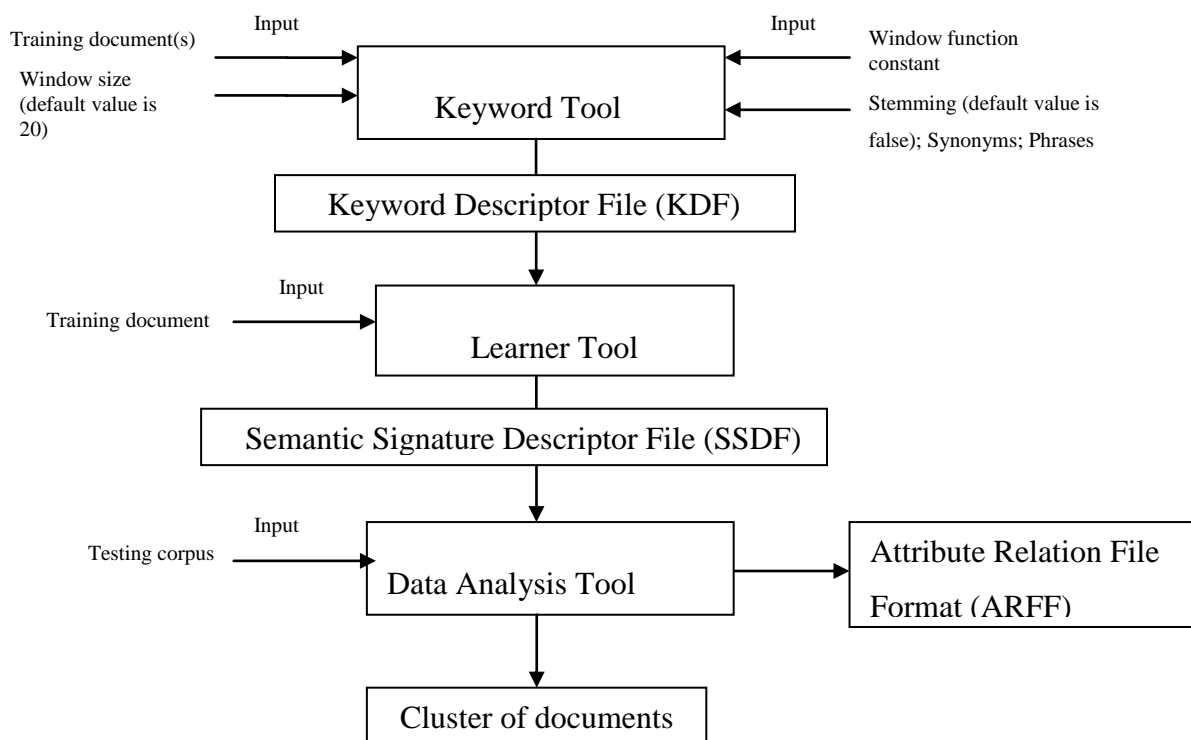


Figure3.1 Input and Output of the tools in Analyst-driven SSMInT Package

Keyword Tool

Keyword Tool helps the analyst to find keyword groups in the training document. Keywords are words that have importance in the document. A group of one or more keywords can be termed as a keyword group. Keyword groups should provide a compact representation of the target content. They should be designed in a way such that they capture semantic content. Immense care should be taken in the design of the keyword groups. The training document is subject to preprocessing and the words in the document, ordered by frequency, are displayed to

the analyst. With the help of the ‘point-back’ option, the analyst selects an appropriate keyword. This will form the first keyword of the keyword group that is going to be generated. Then forward and backward distances are calculated from the selected word to other words in the window. The distances are sorted in descending order. Again, using the point back tool, the analyst selects one of the words to be the second keyword of the keyword group. This process repeats until the analyst feels that the keyword group has semantic meaning embedded in it. This keyword group is saved in a file called Keyword Descriptor File (KDF) which is in ‘.kdf’ format. A KDF contains the details of the keywords in the keyword group, window length, synonyms substituted, phrases used and whether stemming is applied or not.

In the sample text shown in Figure 3.2, the word *drinking* can be selected, from the list which orders the words according to their frequency in the document, as the first keyword. Forward and backward distances are calculated from that word to other words in the window. With the help of point-back, *heavy* can be selected as the second keyword. This process is repeated and *risks* is selected as the third keyword. Thus the keywords which formed the keyword group are *drinking*, *heavy* and *risks*.

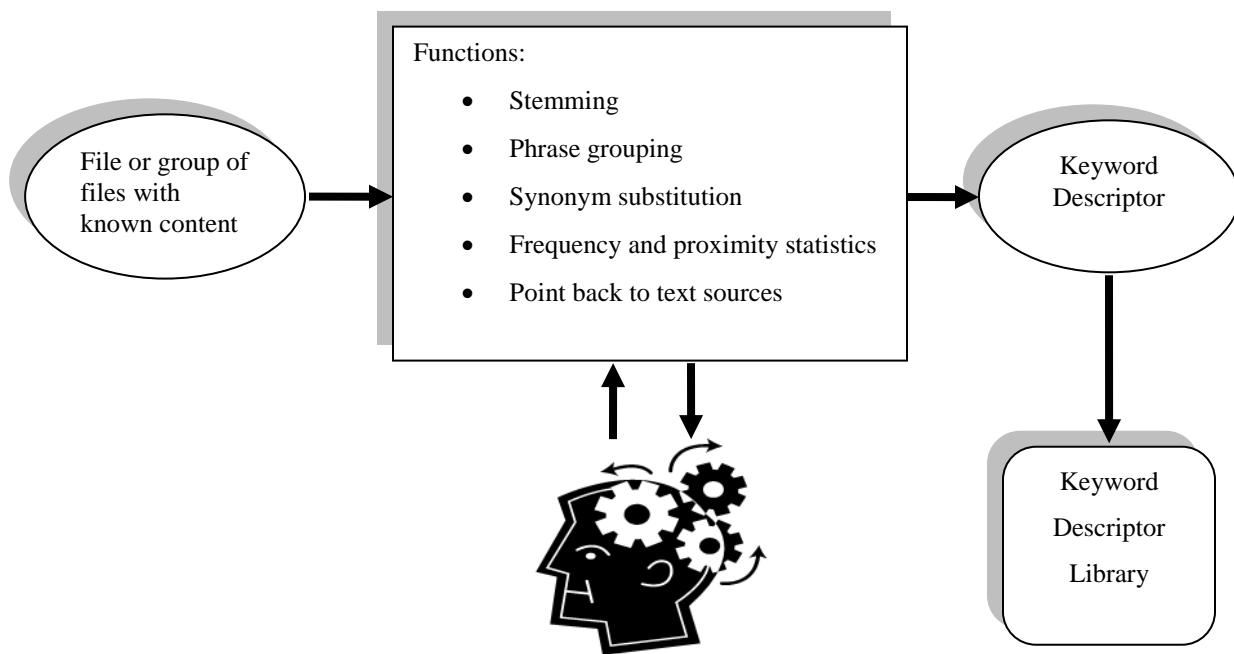


Figure3.2: Human interaction with Keyword Tool. Inputs, Outputs and Functions of Keyword Tool

The main motivation for the development of Automated Keyword Tool (AKT) can be attributed to the fact that the generation of keyword groups in Analyst Driven Keyword Tool is a time consuming and laborious task. Also, rather than taking into account the frequency of words, better heuristics could be used.

Learner Tool

Learner Tool aids in the design of semantic signatures. A cluster of document vectors extracted from the training documents represents the target content and is our instantiation of the concept of a semantic signature. The process of vector generation is explained earlier in Chapter 2. Using the keyword groups generated from Keyword Tool, vectors are generated that capture the semantic relationship between keywords. By clustering the document vectors generated, we achieve the grouping of the vectors that are similar or are pointing in the same direction in the vector space. K-Means Clustering with Euclidean or cosine distance is chosen as the clustering technique and these are implemented in the tool. Clusters that the analyst feels have captured the target content are selected by the analyst. In order to make a decision in choosing the cluster, ‘point-back’ tool is used, which helps in determining whether the cluster selected is capturing the target content or not.

The criteria for a semantic signature ‘cluster’ hit is as described below. If semantic signature Cluster Definition 1 is used to define a cluster of vectors, a document vector is said to *hit* a semantic signature if the Euclidean distance between the vector and the cluster is less than the radius of the cluster. If semantic signature Cluster Definition 2 is used to define a cluster of vectors, a document vector is said to *hit* a semantic signature if the cosine of the angle between the document vector and cluster’s centroid vector is not less than the smallest such measure for a vector within the cluster.

Semantic Signature Descriptor Files (SSDFs) contain all the details about the semantic signature and also about the keywords used in the extraction of the signature. This is saved in xml format with ‘.ssd’ extension along with the Keyword Descriptor File information concatenated to it.

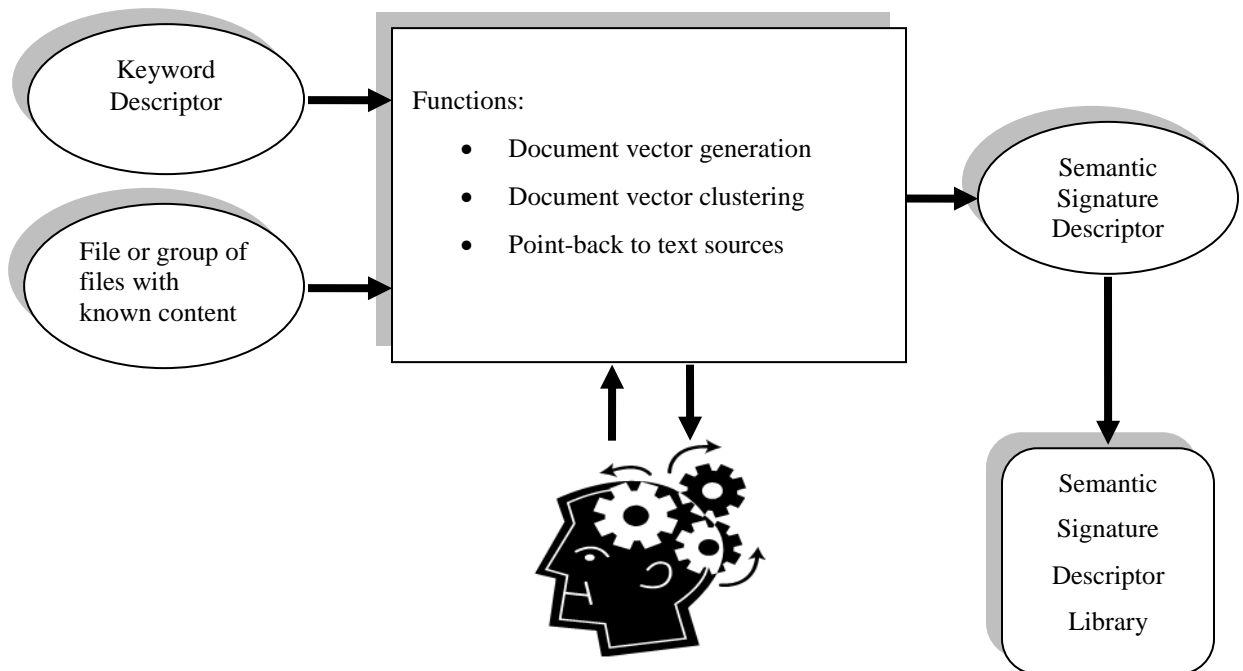


Figure3.3 Human interaction with Learner Tool. Inputs, Outputs and Functions of Learner Tool

The motivation for automating Learner Tool is to select semantic signature clusters without the aid of an analyst. When an expert analyst is involved there is the advantage that the analyst can design semantic signatures for the target content with precision (from the point of view the analyst) that captures nuances. Relieving the analyst from the tedious task of identifying the appropriate clusters is not the only motivation for an automated tool. The automation is designed to *discover* semantic signatures in documents with the target content that an analyst may not know are present or his point of view bias prevents him from seeing.

Data Analysis Tool

The main objective of Data Analysis Tool (DAT) is to detect semantic features in a corpus of documents with unknown content. It is developed to retrieve documents that contain targeted content or cluster a corpus of documents with unknown content based on the semantic content embedded in the semantic signatures. To achieve this, DAT generates a document-semantic signature matrix. In the document- semantic signature matrix, each row represents a document of the testing corpus and each column represents a semantic signature. A row in the document- semantic signature matrix contains the semantic feature vector of the document. The semantic signatures generated from Learner Tool are used in the development of document feature vector from the corpus of documents with unknown content. By computing vector hits and using a clustering algorithm, DAT groups the documents of similar content into clusters. Semantic signatures are quantified as clusters of document vectors from the learning documents.

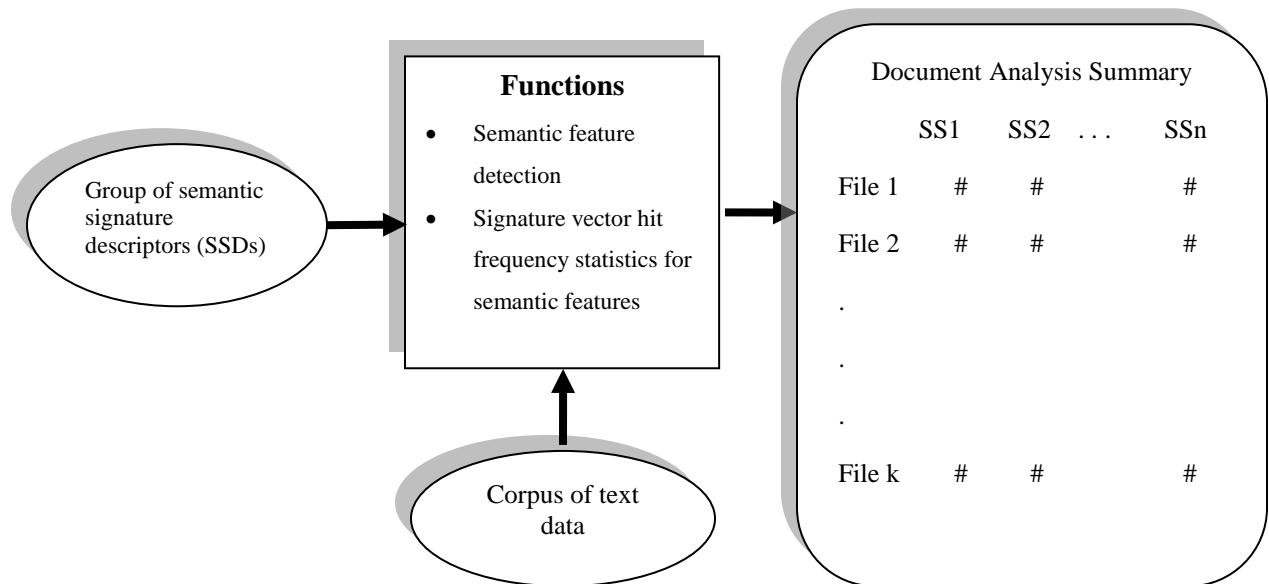


Figure3.4 Inputs, Outputs and Functions of Data Analysis Tool

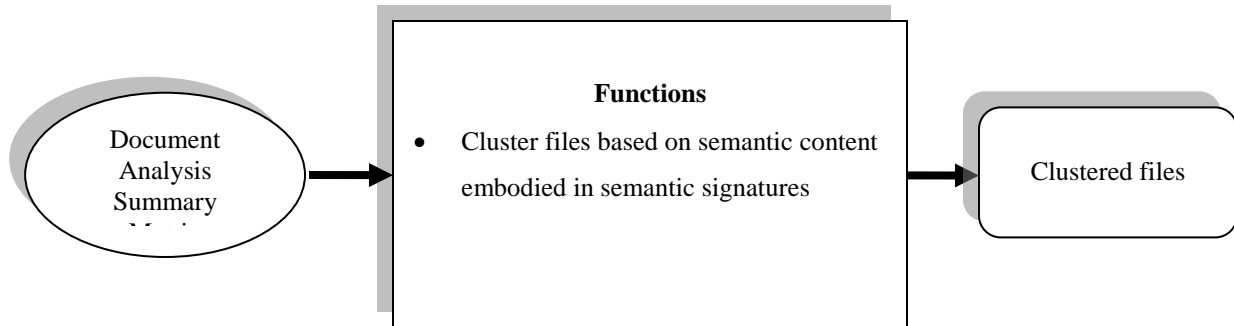


Figure3.5 Overview of document-semantic signature matrix processing

In Automated SSMiT we develop a tool, Semantic Signature Refinement Tool, which refines the set of semantic signature clusters. Even when semantic signatures are carefully handpicked by the analyst, some semantic signatures may be irrelevant. In Analyst Driven SSMiT, no way is provided to prune out semantic signatures that do not have strong directions in the document vector space. From the numerous semantic signature clusters automatically generated, if we can identify semantic signatures that have the potential to form strong document feature vectors, by pruning out noisy and redundant semantic signatures, we can expect a meaningful clustering of the corpus of documents with unknown content.

Chapter 4: Automated Semantic Signature Mining Tools

This section gives insight into the tools in Automated Semantic Signature Mining Tool package. Based on the tools in Analyst Driven SSMInT, these automated tools are developed retaining the stand-alone nature, options provided and the concepts embedded in them. SSMInT requires the analyst to have knowledge about the training documents so that he can select meaningful keyword groups [6]. The goal of Automated SSMInT is to cluster the corpus of documents with unknown content according to semantic content by allowing the system to make knowledge- driven decisions without the intervention of the analyst. The flow of this chapter is as follows: Section 4.1 introduces Automated Semantic Signature Mining Tool Package with the following sections presenting the details of Automated Keyword Tool (AKT), Automated Learner Tool (ALT), Hits Array Generator Tool (HAGT) and Semantic Signature Refinement Tool (SSRT). Tools in the package are developed using Microsoft Visual Studio IDE and C#.

4.1 Overview

Tools in Automated SSMInT are developed with the objective to find the documents in the corpus containing the concepts embodied in the semantic signatures developed with minimal or no intervention of the human analyst. The tools in this package are trained to make decisions intelligently.

The flow of the tools in Automated SSMInT is shown in Figure 4.1. Automated Keyword Tool(AKT) is provided with the training document(s), the window size and window function constant, lists of synonyms and phrases and information about whether stemming has to be applied or not. AKT then generates the keyword groups automatically using TF-IDF and forward and backward distance calculations. Each keyword group is saved into a Keyword Descriptor File (KDF). All KDFs are saved in a folder. Next, Automated Learner Tool (ALT) inputs the training document and the folder of KDFs along with the details of the clustering algorithm to be employed (like the distance metrics to be used and number of clusters to be formed etc). Document Vectors are generated, refined and clustered to form semantic signatures. Each semantic signature is saved in a Semantic Signature Descriptor File (SSDF). Then, the Hits Array Generator Tool (HAGT) generates a matrix of hits, accepting a folder containing semantic signatures and corpus of documents with unknown content as input. This matrix contains semantic signatures as columns and names documents of the corpus as rows. The matrix along with names of semantic signatures and documents of the corpus are saved in a Hits Array Descriptor File (HADF). This is given as input to the most intelligent tool of Automated SSMInT, Semantic Signature Refinement Tool (SSRT). This tool not only removes noisy semantic signatures but also the relevant redundant semantic signatures. The hits array is reconstructed after semantic signature refinement and its row vectors can be clustered using the inbuilt K-Means algorithm, which allows the documents of the corpus to be grouped according to the content embodied in the discovered semantic signatures. The reconstructed hits array is also available in .ARFF format for input to WEKA for further interpretation of results.

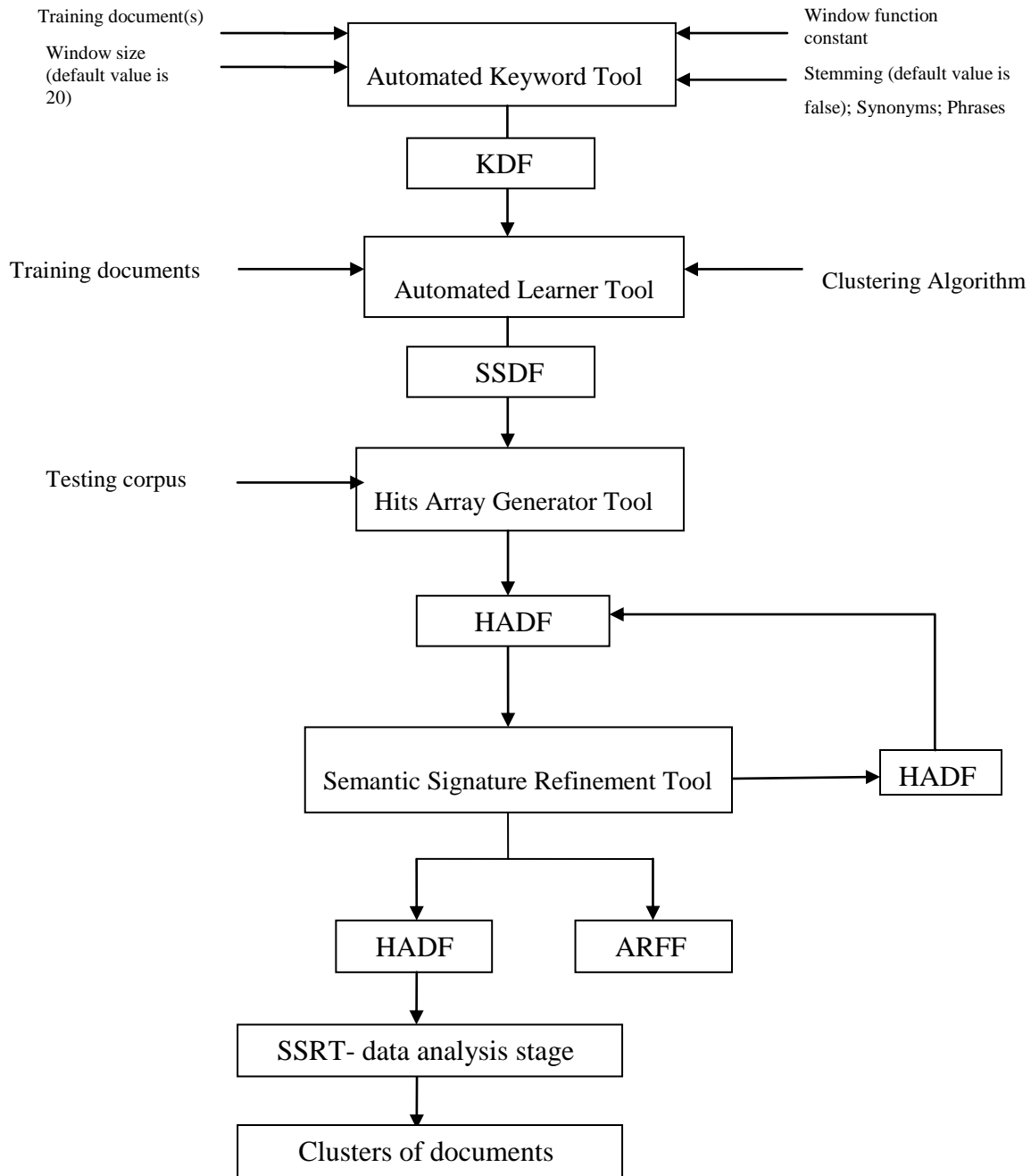


Figure4.1 Input and output of the tools in Automated SSMInT

Introduction to Automated Keyword Tool

Automated Keyword Tool is designed to automate the process of keyword group generation. The development of Automated Keyword Tool (AKT) is motivated by the fact that the generation of keyword groups in Analyst Driven Keyword Tool is a time consuming and laborious task. The idea that better heuristics can be used for generation of the keyword lists also provided motivation for the development of AKT. Automated Keyword Tool is the automation of the

existing Keyword Tool with the addition of TF-IDF. TF-IDF, by weighing the words in the training document, identifies the words likely to be used as keywords without the aid of the analyst. However, some assumptions are made while automating the tool. The size of the keyword group is fixed to be three. The assumption that there should be three keywords in the keyword group was made after varying the size of the keyword group from two to four. Often, two keywords failed to capture semantic content within them and groupings of four keywords are difficult to find in the window size. The size of the keyword group is fixed to three after observing several keyword groupings that held semantic content with three keywords.

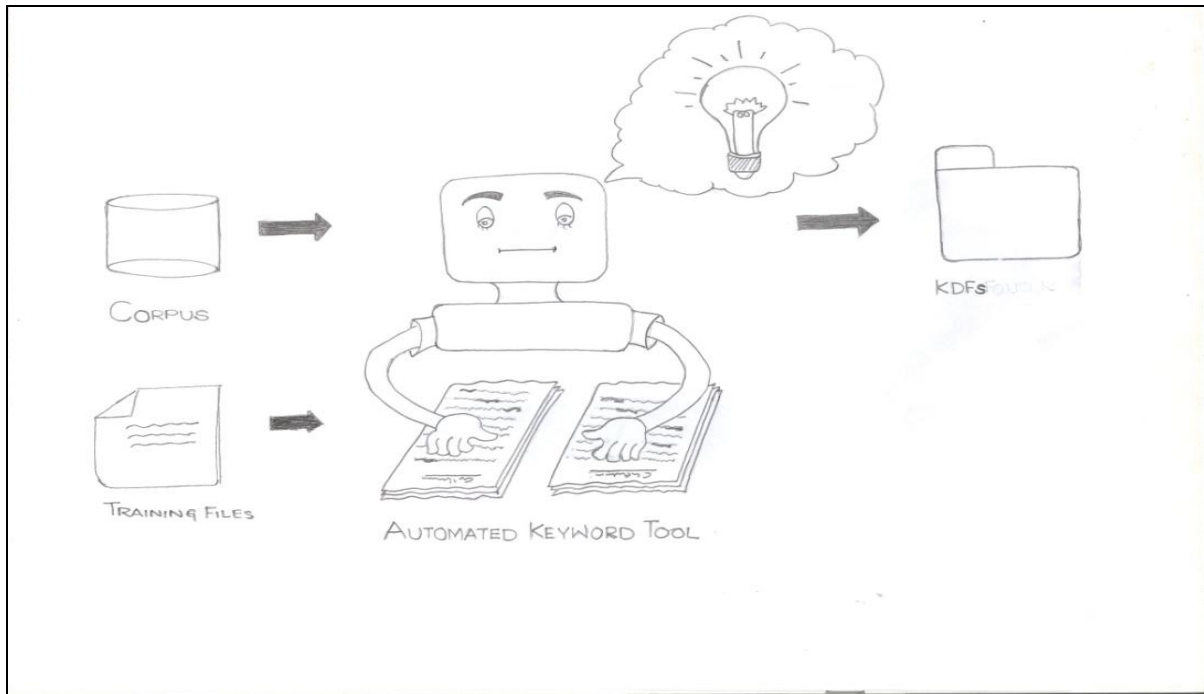


Figure 4.2 Automated Keyword Tool working intelligently to generate keyword groups.

Functions of Automated Keyword Tool

- Keyword group generation
 - Implementation of TF-IDF for Keyword selection.
 - Forward and backward distances between words in the window.
 - Keyword group generation algorithm.
- Keyword Descriptor File generation

Figure 4.3 Functions of Automated Keyword Tool

Introduction to Automated Learner Tool

Automated Learner Tool (ALT) is designed to discover semantic signatures without the aid of an analyst. ALT is capable of selecting the cluster of document vectors with target content which forms the semantic signature as there is no expert analyst involvement in the tool. ALT has an option to choose one cluster among the generated clusters as a semantic signature or can choose to save all the generated clusters as semantic signatures. The automation is also designed

to *discover* semantic signatures in documents with the target content that an analyst may not know are present or his point of view bias prevents him from seeing. In ALT, document vector refinement is achieved by rejecting vectors with limited content. Automated Learner Tool is the automation of the existing Learner tool with the addition of vector refinement which will be presented in detail in Section 4.3.1.

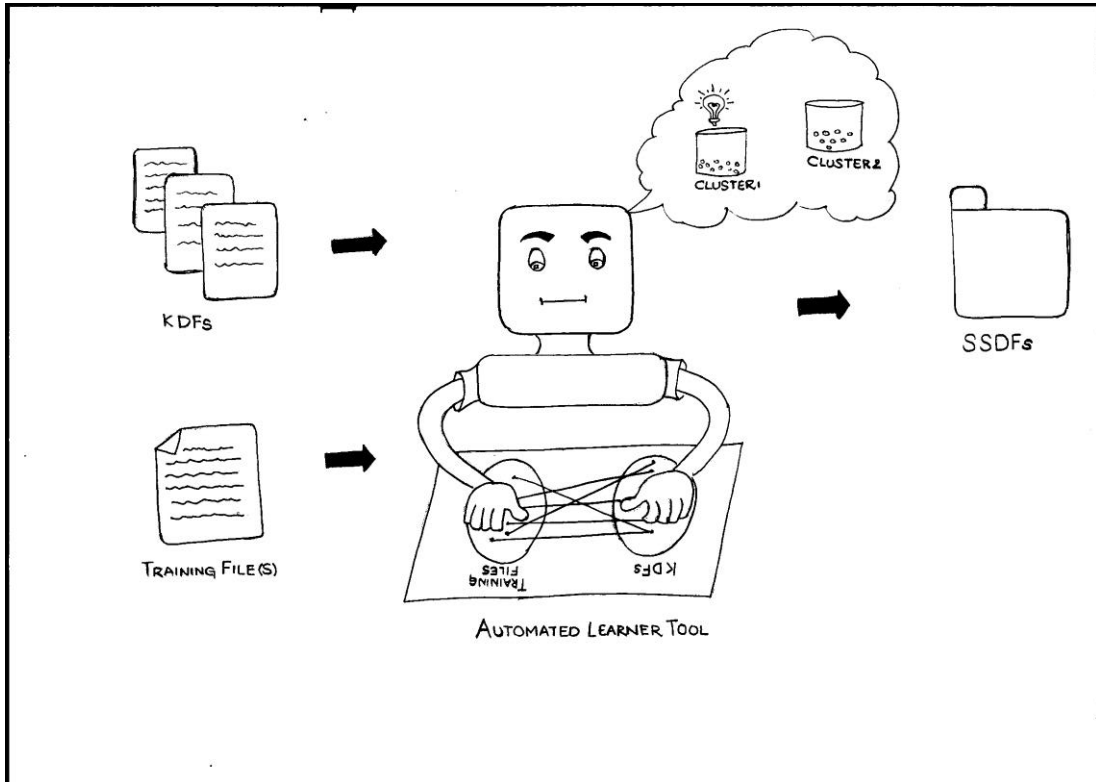


Figure 4.4 Automated Learner Tool working intelligently to refine vectors and select clusters

Functions:

- Document vector generation
- Document vector Refinement
- Clustering of document vectors
- Cluster Selection
- Semantic Signature Descriptor File generation

Figure 4.5 Functions of Automated Learner Tool

Introduction to Hits Array Generator Tool

Hits Array Generator Tool (HAGT) generates a document-semantic signature matrix, also called as the hits array, having semantic signatures as columns and documents with unknown content as rows. Each row contains a document feature vector. Hits Array Generator Tool was a part of Data Analysis Tool in Analyst Driven SSMInT. The motivation for the separation of the document-semantic signature matrix is to save time. Even though the semantic signatures are refined (i.e. some are deleted) in Semantic Signature Refinement Tool, this doesn't

affect the hit calculation of the remaining semantic signatures on the documents. So, it is not necessary to repeat hit calculations after the refinement. Instead, it is easy to reconstruct the already existing matrix by removing the noisy semantic signature columns from the matrix. Hence, the generation of the document-semantic signature matrix is separated from the Data Analysis Tool and is developed as a separate tool named Hits Array Generator Tool.

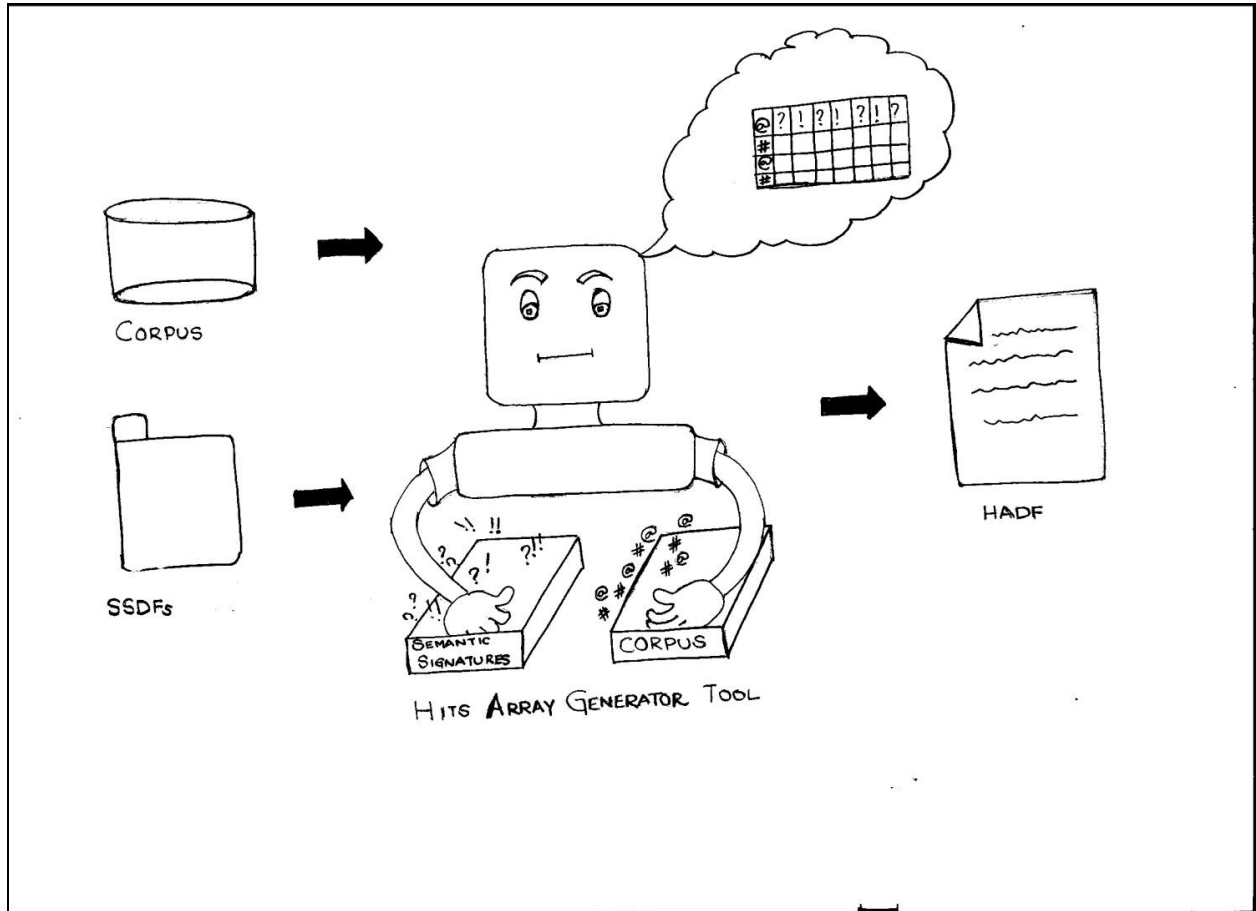


Figure 4.6 Hits Array Generator Tool working on generating the document-semantic signature matrix

Functions:

- Document vector generation
- Hits array calculation
- Hits Array Descriptor File generation

Figure 4.7 Functions of Hits Array Generator Tool

Introduction to Semantic Signature Refinement Tool

Semantic Signature Refinement Tool (SSRT) is designed to intelligently prune the automatically generated semantic signatures. Pruning of semantic signatures not only eliminates the irrelevant semantic signatures but also eliminates redundant relevant semantic signatures. With the aid of Singular Value Decomposition, a mechanism is developed to retain powerful semantic signatures. The tool has an option to refine the semantic signatures further by reducing the dimensionality using SVD. The Hits array which is the output of the Hits Array Generator

Tool is reconstructed from the original document-semantic signature matrix by eliminating the columns containing the irrelevant or redundant relevant semantic signatures. This reconstructed Hits Array can be saved in .ARFF format to be input to WEKA or its document feature vectors are clustered using the built-in K-Means clustering technique to discover groupings of documents with similar content.

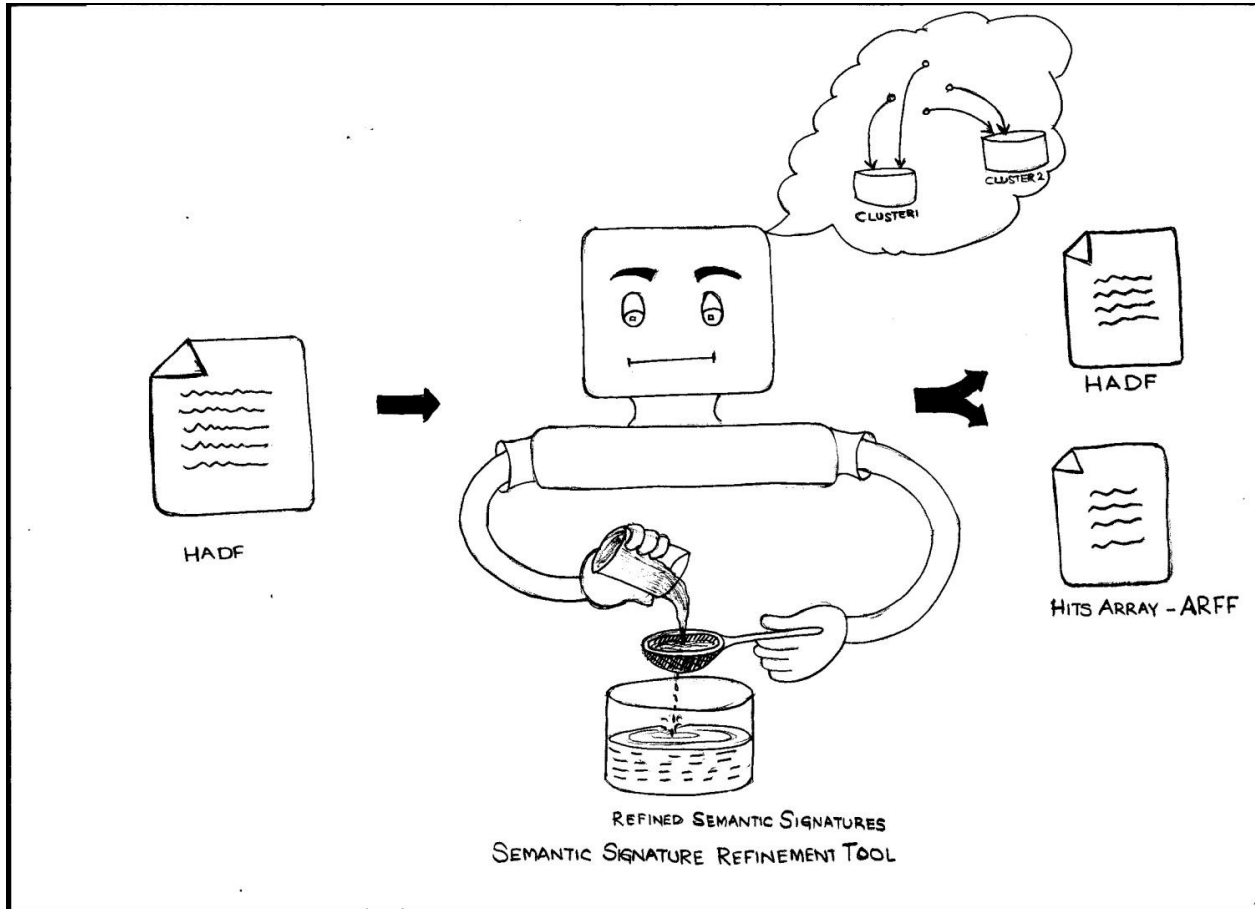


Figure 4.8 Semantic Signature Refinement Tool working intelligently to refine semantic signatures

- | |
|--|
| <p>Functions</p> <ul style="list-style-type: none"> • Semantic Signature Refinement • Dimensionality Reduction • Reconstruction of hits Array • Clustering of document feature vectors |
|--|

Figure 4.9 Functions of Semantic Signature Refinement Tool

4.2 Automated Keyword Tool

Automated Keyword Tool (AKT) generates keyword groupings using Term Frequency-Inverse Document Frequency (TF-IDF). First, the training document is preprocessed and TF-IDF helps AKT in the identification of the keywords. TF-IDF

scores each word in the training document. In AKT, the default size of the window is 20 as the most appropriate average English sentence length for most pieces of writing is about 15 to 20 words [31]. Using the keyword group generation algorithm, keyword groups are generated. Each keyword group is saved in a Keyword Descriptor File.

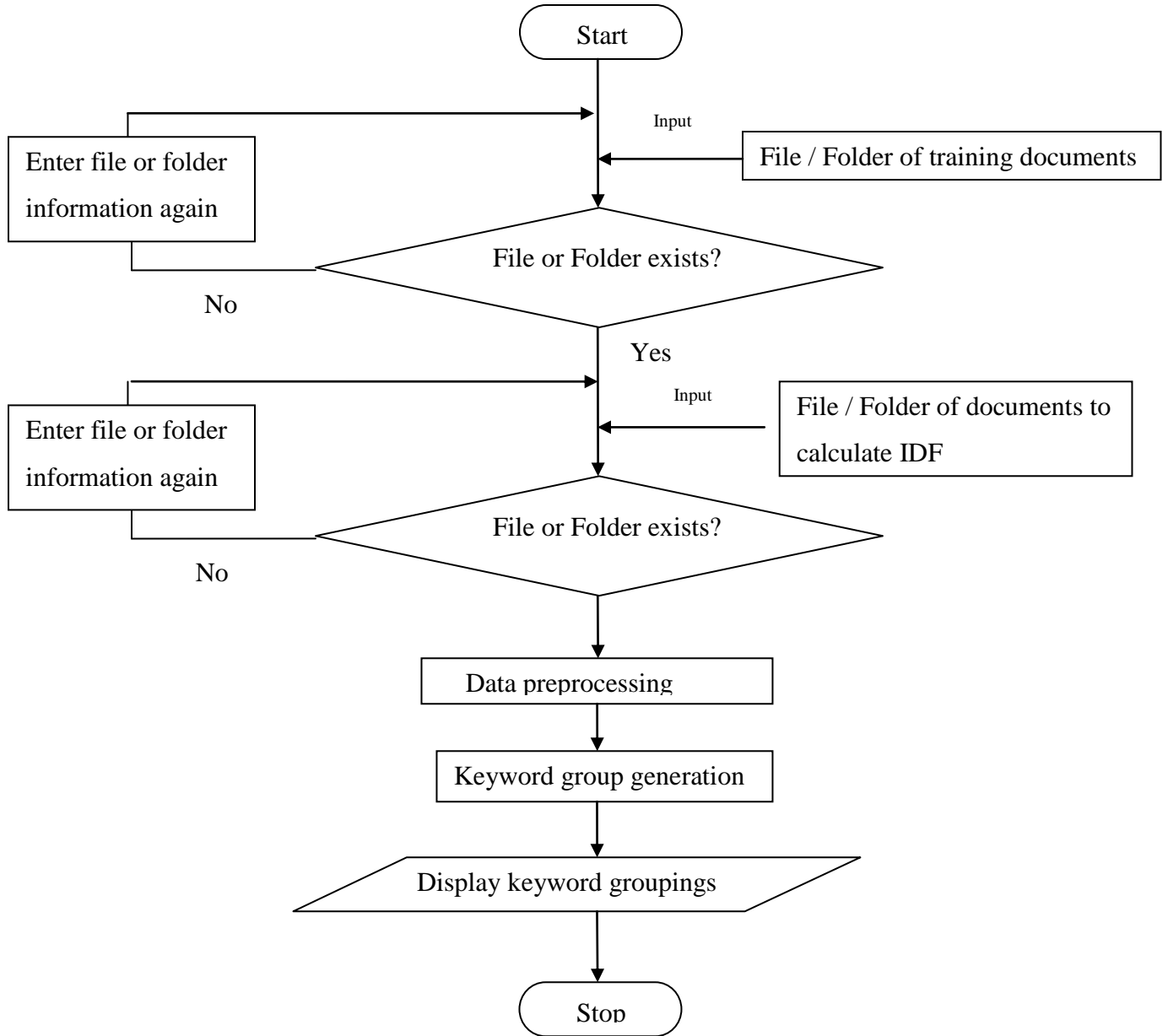


Figure 4.10 Working of Automated Keyword Tool

4.2.1 Keyword group generation algorithm

A greedy algorithm is chosen to automatically generate keyword groups. The process followed for their generation is:

- From the sorted list of words given by TF-IDF, the top n highly scored words are returned. The number n is given as input upon start up of Automated SSMInT. Suppose,

for example, n is five. Let the five words be **a**, **b**, **c**, **d** and **e**. These words are shown in List1.

a
b
c
d
e

- Inside a window, the weights are calculated from each of the five words chosen to every other word in using the window-weight function in both forward and backward direction. From the sorted list of weights, only the three most strongly correlated words are taken. The algorithm ensures that there are no duplicates.

First, keyword **a** is taken and weights are calculated from **a** to other words in the window in both forward and backward direction. **a1**, **a2** and **a3** are found to be top three highly correlated words with **a** and are represented as List2. Also **a** is not equal to **a1** or **a2** or **a3**.

a1
a2
a3

- Next, for each word in List2, forward and backward distances are calculated to find the strongly correlated words with this word. Only the top two are taken, avoiding the duplicates. **a11** and **a12** are found to be the top two strongly correlated words with **a1** and are represented as List3.

a 11
a 12

- Then, keyword groups are generated with all possible combinations of above words. If there are duplicates in a keyword group, that combination of keywords is discarded. That is if **a** is not equal to **a1** and **a11**, **a1** is not equal to **a11**, keyword group is generated as **a**, **a1**, **a11**. Else, it is discarded. All other combinations of keyword groups are also generated in a similar fashion.

a,a1,a11
a,a1,a12
a,a2,a21
a,a2,a22
a,a3,a31
a,a3,a32

- The above three steps are repeated for each word in List1.

4.2.2 Working of Automated Keyword Tool

Automated Keyword Tool (AKT) can accept as input either a file or a folder of training documents. Currently text, xml or html file types can be recognized by the tool. If a single file is given, the type of the file is determined and the corresponding parser is used to extract the text from the file discarding the tags if any. If a group of files are given in a folder, the type of each file is determined and the corresponding parser is used to extract text from the file and concatenates it to another file which contains the text from all files in the folder. Next, the corpus

from which inverse document frequency is to be calculated is loaded. AKT can also substitute synonyms, perform stemming and join two or more words as a phrase if required. Next, AKT performs the folder or file existence check for both the training documents and the corpus. If they exist, the single file or the merged file (in case of a folder) is preprocessed and the preprocessed text is split into words.

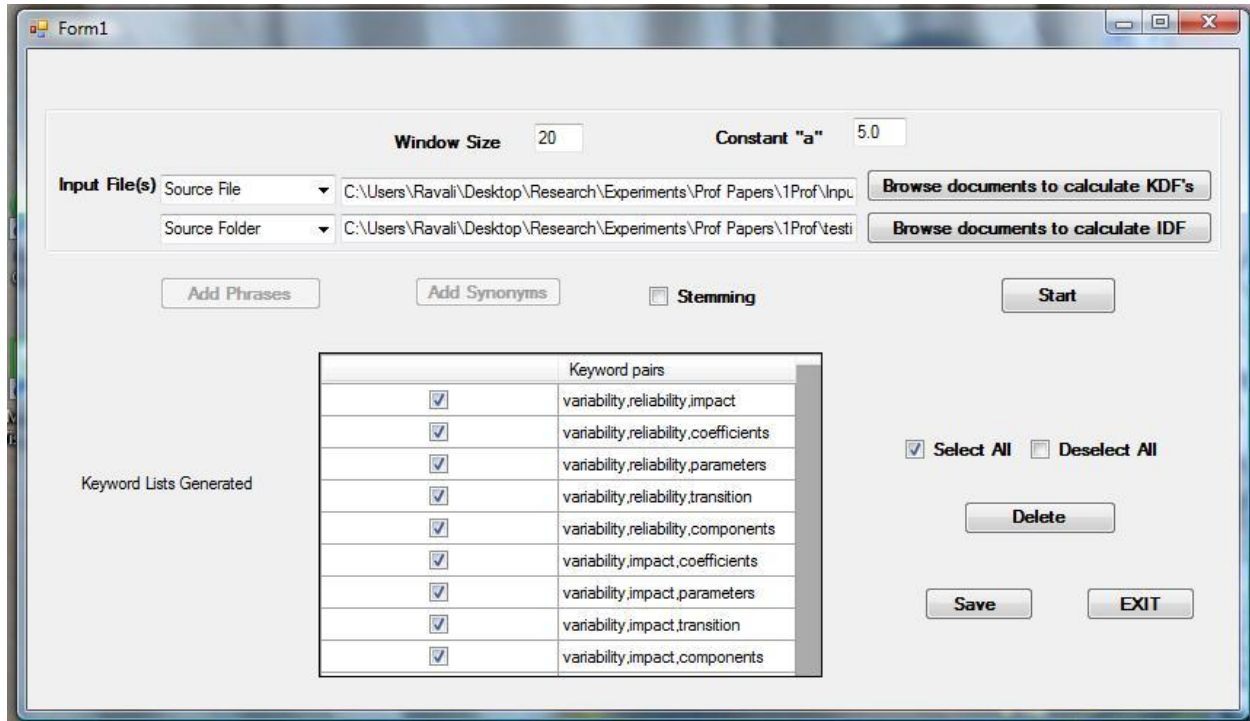


Figure 4.11 Screenshot of Automated Keyword Tool

The term frequency is computed from the training file and the inverse document frequency is computed from the corpus. AKT then generates the keyword groups using the keyword group generation algorithm and displays them to the analyst. The analyst can select either some or all of the keyword groups. Then, each selected keyword group is saved in a separate Keyword Descriptor File (KDF).

4.2.3 Output of Automated Keyword Tool

For a given training document, a group of Keyword Descriptor Files (KDFs) are generated by AKT and are saved in a folder. KDFs are stored in XML format with the extension '.kdf'. XML format is chosen because it is one of the industry standard communication protocol which can be easy for other programs to understand the output. Keyword Descriptor File contains the following tags:

AutomatedKeywordTool: This tag contains the *version* attribute which gives information about the version of Automated Keyword Tool. The purpose of this tag is to maintain a log of versions. At present its version is 1.1

Stemming: This tag gives information on whether the stemming option is set or not. If stemming is *set*, words are stemmed to their roots and all the generated KDFs will have *yes* as a

value in the *used* attribute and the name of the stemmer in the *stemmer* attribute. Else the *used* attribute will have the value *no*.

Source: This tag contains the path of the training document. If the training document is a single file, the value of the *file* attribute is set to *yes*. Else, it is set to *no*. Similarly, if the keywords are extracted from multiple training documents in a folder, the value of the *folder* attribute is set to *yes*. Else, it is set to *no*.

Window length: This tag records the value of length of the window in the *window length* attribute.

Function Constant: This tag records the value of the constant *a* which is used in window weight calculations.

Keywords: This tag contains keywords which formed the keyword group.

Synonyms: This tag contains the words and the synonyms substituted for those words.

Phrases: This tag contains the words which are joined as a phrase in the training document.

```
<AutomatedKeywordTool version="1.1">
<stemming used="no" stemmer="porter"></stemming>
<source folder="no" url="no" file="yes"> C:\Users\Ravali\test.kdf </source>
<windowLength length="20"></windowLength>
<functionConstant a="5"></functionConstant>
<keywords>video, partitioning, adaptive</keywords>
<synonyms> listen, hear; see, look </synonyms>
<phrases> throat singing, throat cancer </phrases>
</AutomatedKeywordTool >
```

Figure 4.12 Structure of Keyword Descriptor File

4.2.4 Comparisons between the Analyst Driven and Automated Keyword Tool

In the process of reducing the analyst burden in the generation of keyword groups, AKT was developed. However, AKT has some limitations and improvements over the Analyst-driven Keyword Tool which are presented below:

- In Analyst-driven Keyword Tool, the first keyword list of the keyword group is generated by calculating the frequency of words. In Automated Keyword Tool, the list is generated using an established technique “Term Frequency – Inverse Document Frequency”.
- AKT can substitute synonyms and replace words as phrases only before the generation of keyword groups unlike the Keyword Tool in Analyst-Driven SSMInT which can substitute synonyms and replace words as phrases even in the middle of the generation of keyword groups.
- Keyword groups are generated automatically using the Keyword Group Generation Algorithm. AKT has the ability to select or deselect all the keyword

groups at once. Each selected keyword groups are saved in a separate KDF and all the KDFs are stored in a folder.

- There is no ‘point back’ option in AKT.
- Some of the keyword groups generated by AKT may not make sense but are not refined at this point of time.

4.3 Automated Learner Tool

Automated Learner Tool discovers semantic signatures in documents with the target content that an analyst may not know are present or his point of view bias prevents him from seeing. It can select semantic signature clusters without the aid of an analyst. ALT takes as inputs KDFs, training document, information about clustering algorithm and generates semantic signature clusters. ALT has an option to choose one cluster among the generated clusters as semantic signature or can choose to save all the generated clusters as semantic signatures. However, deciding whether a cluster can be designed as a semantic signature or not, at this point of time can prove to be immature as it is based only on weight calculations. So, all generated clusters are saved as semantic signatures which are refined at a later stage. Each generated semantic signature is saved in a Semantic Signature Descriptor File (SSDF).

4.3.1 Document vector refinement

The generation of document vectors is explained in detail in Chapter 2. These generated document vectors are subject to refinement. A refinement criterion is designed to filter out the document vectors which have fewer or no interactions between all the keywords in a given group. A document vector is said to pass the refinement criteria only if it has interactions between all keywords in that group. Else, it is rejected.

Consider the nine components of the vector shown in the following table. Here K1 stands for the first keyword in the keyword group, K2 stands for the second keyword in the keyword group and K3 stands for the third keyword in the keyword group.

Table 4.1 Representation of document vector in matrix format

	K1	K2	K3
K1	1	2	3
K2	4	5	6
K3	7	8	9

Document vectors are discarded if:

- It contains zero value in all the components.
If the document vector contains zeroes in all components; it is discarded.
- It only has interactions between same keywords in the group (K1 to K1 and K2 to K2 and K3 to K3)
If the document vector contains values only in first, fifth and ninth components, it is discarded.
- It doesn't have combinations of different keywords interactions in the group. (K1 to K2 or K2 to K1 and K1 to K3 or K3 to K1 and K2 to K3 or K3 to K2)

If the document vector does not contain values in second or fourth component, third or seventh component and sixth or eighth component; it is discarded.

To explain the refinement criteria with an example, consider a cluster of document vectors. Let the document vectors generated are:

0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
0.76, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.21
0.0, 0.32, 0.0, 0.0, 0.0, 0.43, 0.71, 0.0, 0.0
0.0, 0.0, 0.0, 0.0, 0.0, 0.47, 0.0, 0.91, 0.0
0.23, 0.42, 0.0, 0.0, 0.0, 0.26, 0.91, 0.0, 0.32

Out of the document vectors listed above, only fourth and fifth document vectors survived the refinement. All other vectors are filtered out due to the reasons listed below:

- First document vector is filtered out as all the components have zero value.
- Second document vector is filtered out because it only has interactions between same keywords. There is no interaction between different keywords.
- Fourth document vector is filtered out because there aren't interactions between all the different keywords. It is missing interaction between K1 and K2.

4.3.2 Ways of selecting clusters as semantic signatures

In Learner Tool, the expert analyst is involved in the design of semantic signature by selecting a cluster that he feels is the representation of the target content. But, in ALT, the tool itself should decide whether the generated cluster is representing the target content or not. In order to achieve this, we developed a technique which, by some weight calculations, decides whether the cluster is representing target content or not.

4.3.2.1 Save cluster with more component weight

ALT selects a cluster as semantic signature if it has high weight compared to other clusters. This weight is generated by adding the individual component weights of different keyword interactions in the vector and is normalized by number of vectors in the cluster. Referring to Table 4.2, only the weights of components two, three, four, six, seven and eight are added.

4.3.2.2 Save all Clusters

Without deciding on selecting a cluster as semantic signature among the generated clusters, this option when enabled, saves all clusters as semantic signatures. ALT will be relieved of the burden of making judgment on which cluster can form the semantic signature. The refinement of the semantic signatures is done by Semantic Signature Refinement Tool.

4.3.3 Working of Automated Learner Tool

Automated Learner Tool (ALT) takes as input Keyword Descriptor Files (KDFs) and training documents. It then allows the analyst to specify the clustering technique, distance measure (if any) and a way of selecting clusters as semantic signatures. ALT performs the file or folder existence check for the KDFs and training documents. It then generates document vectors. Next, the document vectors are refined employing the refinement criteria. The refined document vectors are then clustered and each cluster is saved in a Semantic Signature Descriptor File (SSDF).

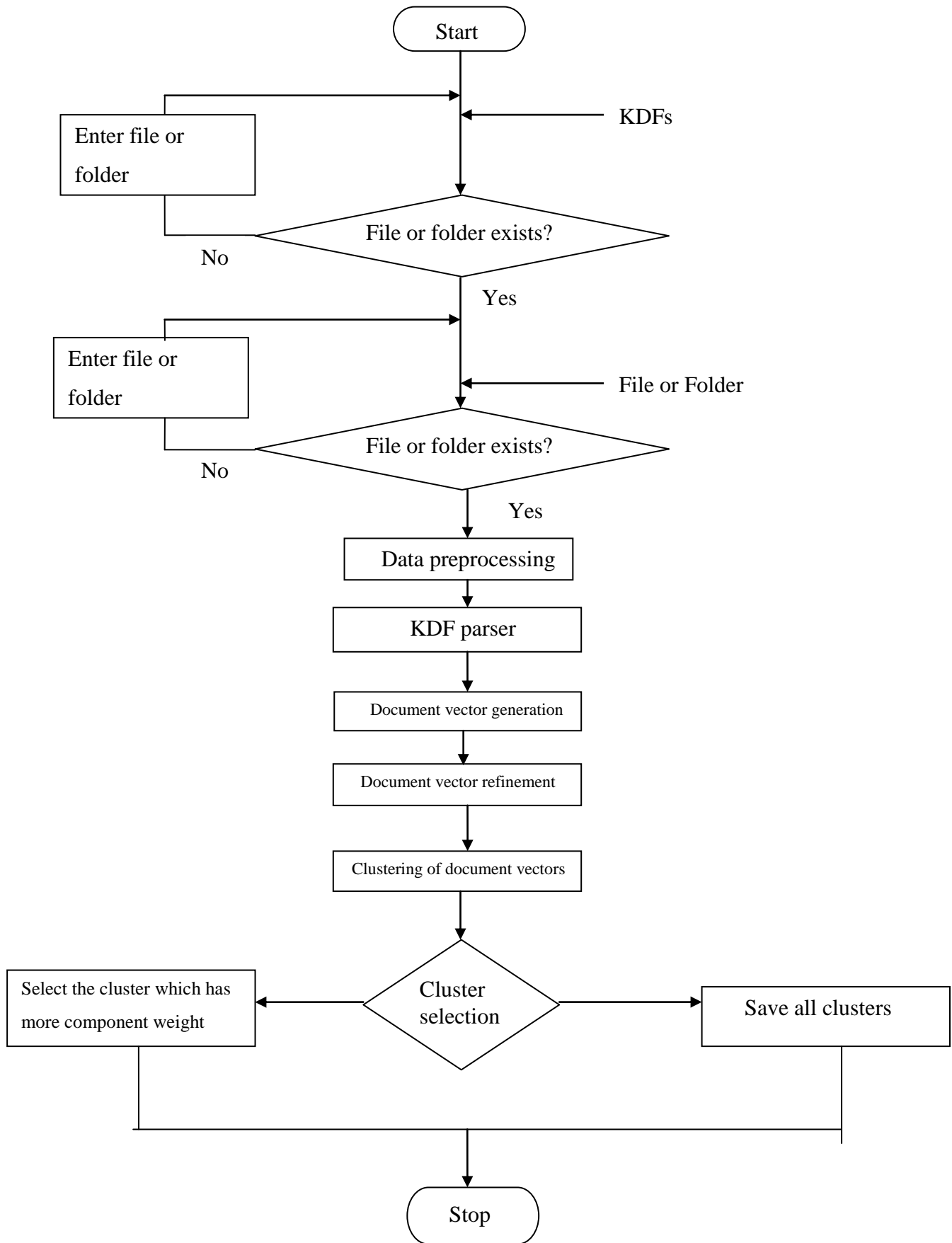


Figure 4.13 Working of Automated Learner Tool

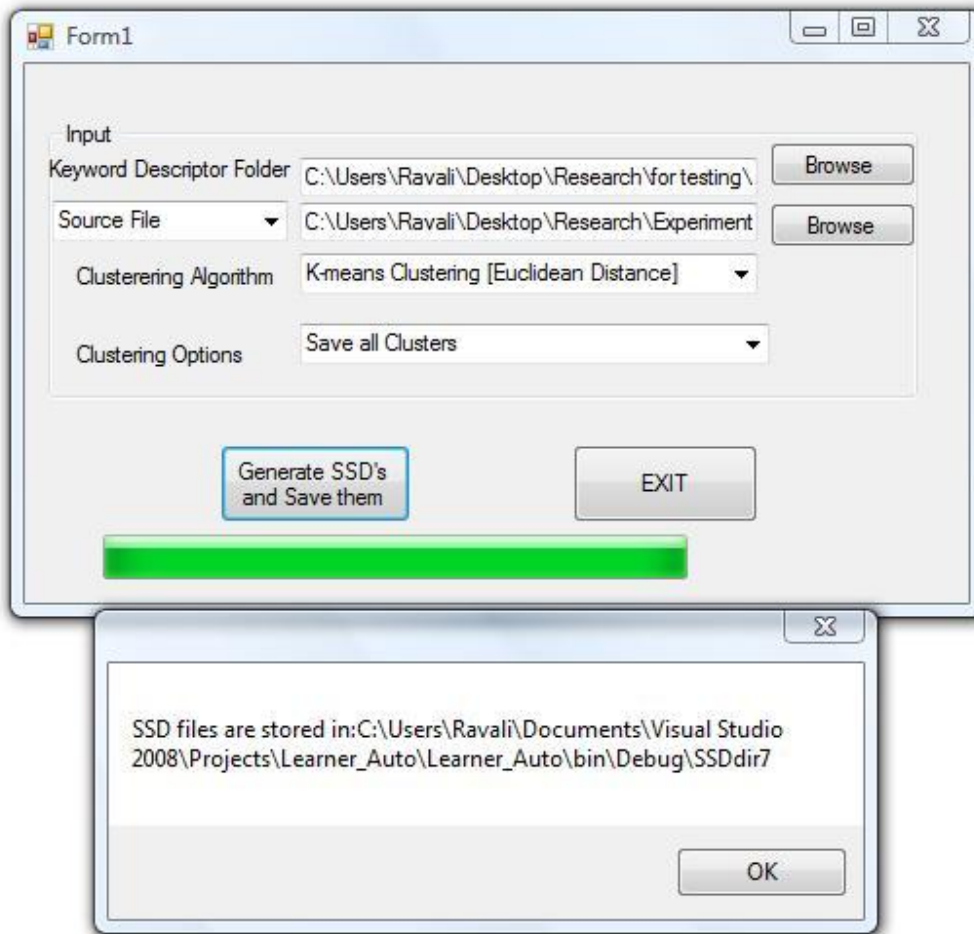


Figure 4.14 Screenshot of Automated Learner Tool

4.3.4 Output of Automated Learner Tool

For a given training document and a group of Keyword Descriptor Files (KDFs), group of semantic signatures are generated by ALT and each of the semantic signature is saved in Semantic Signature Descriptor File (SSDF). All SSDFs are stored in a folder. SSDFs are stored in XML format with the extension ‘.ssd’. XML format is chosen because it is one of the industry standard communication protocol which can be easy for the other programs to understand the output. Semantic Signature Descriptor File contains the following tags:

AutomatedLearnerTool: This tag contains the *version* attribute which gives information about the version of Automated Learner Tool. The purpose of this tag is to maintain the log of versions. At present its version is 1.1

kdfSource: This tag contains the information about the path of the location where Keyword Descriptor Files are stored.

Source: This tag contains the path of the training document from which keywords are extracted. If the training document is a single file, the value of the *file* attribute is set to *yes*. Else, it is set to *no*. Similarly, if the keywords are extracted from multiple training documents in a folder, the value of the *folder* attribute is set to *yes*. Else, it is set to *no*.

clusterer: This tag contains the *name* attribute which gives information about the clustering algorithm used and number of clusters.

centroid: This tag contains the *r* attribute whose value is the distance between the centroid and the farthest vector in the cluster if the distance measure used is Euclidean distance. In case of cosine distance, the attribute *r* stores the angle between the centroid and the farthest vector in the cluster. The *distance measure* attribute tells about the metric used. This tag also includes information about the centroid of the cluster.

Document vectors: This tag contains the group of document vectors in that cluster.

Stemming: This tag gives information on whether the stemming option is set or not. If stemming is *set*, words are stemmed to their roots and SSDF will have *yes* as a value in the *used* attribute and the name of the stemmer in the *stemmer* attribute. Else the *used* attribute will have the value *no*.

Window length: This tag records the value of length of the window in the *window length* attribute.

Function Constant: This tag records the value of the constant *a* which is used in window weight calculations.

Keywords: This tag contains keywords which formed the keyword group.

Synonyms: This tag contains the words and the synonyms substituted for those words.

Phrases: This tag contains the words which are joined as a phrase in the training document.

```
<AutomatedLearnerTool 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</vectors>
<stemming used="no" stemmer="porter"></stemming>
<>windowLength length="20"></windowLength>
<functionConstant a="5"></functionConstant>
<keywords>adaptive, video, partitioning</keywords>
<synonyms> see, watch; hear, listen </synonyms>
<phrases> throat singing; throat cancer </phrases>
</AutomatedLearnerTool >
```

Figure 4.15 Structure of Semantic Signature Descriptor File

4.3.5 Comparisons between the manual and automated versions of Learner Tool

To discover semantic signatures without the aid of analyst, ALT was developed. However, ALT has some limitations and improvements over the Analyst-driven Learner Tool which is presented below:

- Document vector refinement is not performed by Learner Tool in Analyst-Driven SSMInT.
- ALT has the ability to select a cluster based on weight calculations or to select all clusters. Each selected cluster forms a semantic signature and is saved in a separate SSDF.
- There is no ‘point back’ option in ALT.
- Some of the semantic signatures generated may not be a good representation of target content but are not refined at this point of time.
- ALT can capture semantic signatures which the analyst may not know are present.

4.4 Hits Array Generator Tool

Hits Array Generator Tool generates document - semantic signature matrix. Document vectors are generated, as explained in Chapter 2, in the testing documents and using the distance measure specified in the SSDF, hits are calculated for each document against each semantic signature. Based on the hit frequencies of document vectors of the testing corpus on semantic signatures, the matrix is populated. This along with relevant information is stored in Hits Array Descriptor File (HADDF).

4.4.1 Calculation of *hits*

The calculation of a *hit* varies with the cluster definitions.

If Cluster Definition 1(CD1) is used, a document vector is said to *hit* a semantic signature iff the distance between the document vectors and the centroid of the cluster is less than the radius of the cluster and all the keywords are present in the testing document’s window.

If Cluster Definition 2 (CD2) is used, a document vector is said to *hit* a semantic signature iff the cosine of the angle between the document vector and cluster’s centroid vector is not less than the smallest such measure for a vector within the cluster.

If Cluster Definition 3 (CD3) is used, similarity scores are computed. Cosine distance between each document vector and all the semantic signatures 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.

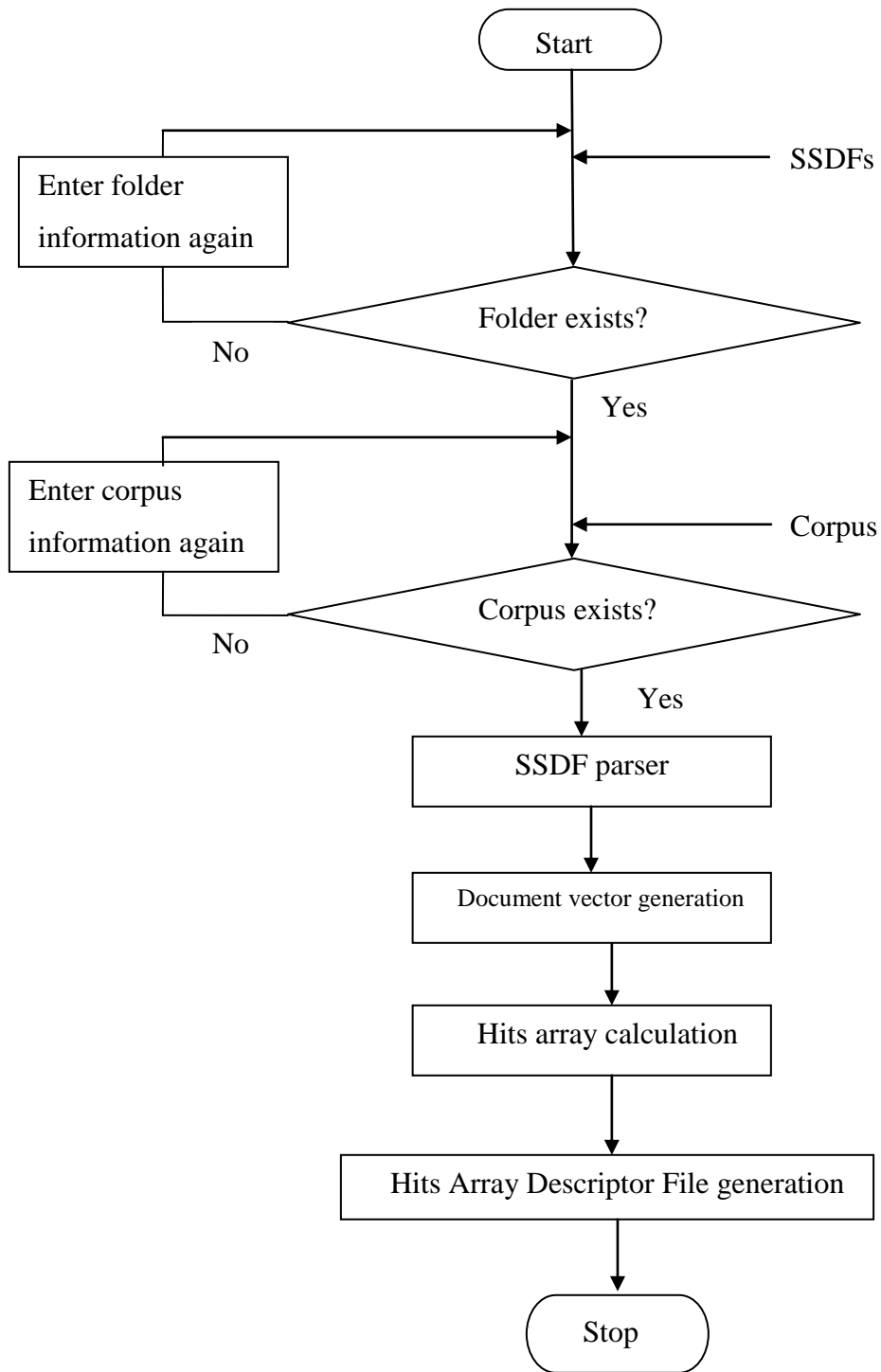


Figure 4.16 Working of Hits Array Generator Tool

4.4.2 Working of Hits Array Generator Tool

Hits Array Generator Tool (HAGT) takes Semantic Signature Descriptor Files (SSDFs) and testing documents as input. Currently, only documents of file types text, html and xml can be recognized by the tool. HAGT performs the file or folder existence check for the SSDFs and testing documents and then hits array calculations are made. Thus document – semantic signature matrix is generated which has semantic signatures as columns and documents with unknown content as rows. Each row contains a document feature vector. All this information is saved in a Hits Array Descriptor File (HADF).

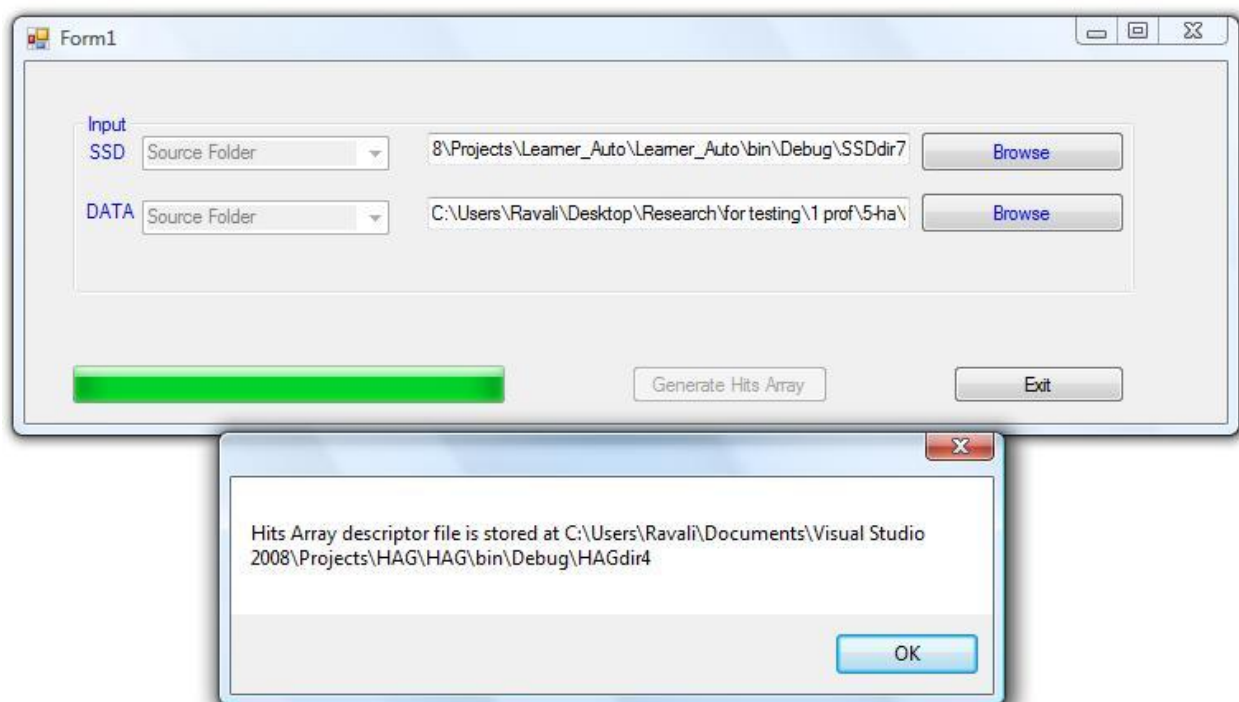


Figure 4.17 Screenshot of Hits Array Generator Tool

4.4.3 Output of Hits Array Generator Tool

The Hits Array Descriptor File is in xml format and contains the following tags:

HitsArrayDescriptorFile: This tag contains the *version* attribute which gives information about the version of Hits Array Generator Tool. The purpose of this tag is to maintain the log of versions. At present its version is 1.1

ssdSource: This tag contains the location of the SSDFs. If there is a single SSDF, the value of the *file* attribute is set to *yes*. Else, it is set to *no*. If SSDFs are in a folder, the value of the *folder* attribute is set to *yes*. Else, it is set to *no*.

dataSource: This tag contains the path of location where testing corpus is stored. If the testing document is a single file, the value of the *file* attribute is set to *yes*. Else, it is set to *no*. If the corpus contains multiple files placed in a folder, the value of the *folder* attribute is set to *yes*. Else, it is set to *no*.

documents: Names of the documents in testing corpus are listed here.

ssds: Names of SSDFs which are used in the generation of hits array are listed here.

HitsArray: This contains document feature vectors of all documents in the testing corpus.

```
<HitsArrayDescriptorFile version="1.1">
<ssdSource folder="yes" file="no">C:\Users\Ravali\Desktop\SSDs</ssdSource>
<dataSource folder="yes" file="no">C:\Users\Ravali\Desktop\papers </dataSource>
<documents>
C:\Users\Ravali\Desktop\Exp\test1.txt
C:\Users\Ravali\Desktop\Exp\test2.txt
C:\Users\Ravali\Desktop\Exp\test3.txt
</documents>
<ssds>
C:\Users\Ravali\Desktop\Exp\SSDs\c1_abuse_alcohol_national_0.ssd
C:\Users\Ravali\Desktop\Exp\SSDs\c1_abuse_alcohol_national_1.ssd
</ssds>
<HitsArray>
2 5
1 3
2 0
</HitsArray>
</HitsArrayDescriptorFile>
```

Figure 4.18 Structure of Hits Array Descriptor File

4.5 Semantic Signature Refinement Tool

Semantic Signature Refinement Tool is designed to intelligently refine noisy and redundant semantic signatures which are generated automatically by ALT. In order to overcome the problem of strong redundant relevant semantic signatures dominating some good but not so strong signatures, a mechanism is developed. Semantic signatures are grouped according to their relevancy and are clustered using KMeans. Then, SSRT selects potential semantic signatures by weights assigned to each semantic signature with the help of Reduced Singular Value Decomposition (SVD). The semantic signatures retained by Semantic Signature Refinement Tool (SSRT) are saved in a folder and are used in the classification of the documents of the testing corpus. The hits array is reconstructed with retained semantic signatures by eliminating the columns of noisy and redundant semantic signature from the document – semantic signature generated by HAGT.

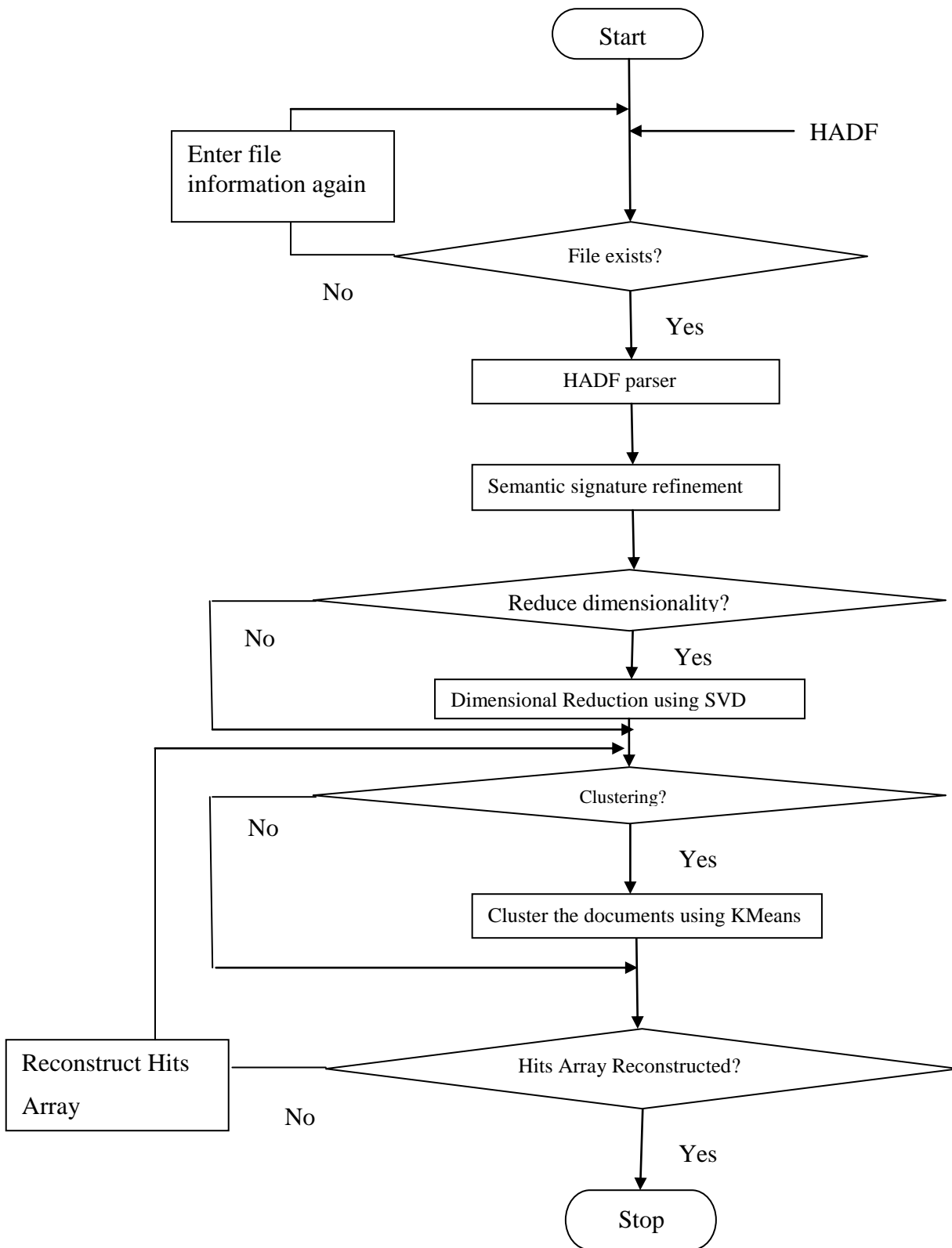


Figure 4.19 Working of Semantic Signature Refinement Tool

4.5.1 Refinement of semantic signatures

Few clicks on AKT and ALT would generate lot of keyword groups and semantic signatures from the training documents, thanks to automation. Undoubtedly, automation has the advantage of saving time and burden of the analyst. In spite of these advantages, a disadvantage is that it adds in a lot of noise. Lot of semantic signature clusters generated by ALT may not capture semantic content. So, they must be pruned. This means the pruning of semantic signatures is an important step and should be done with immense care and attention.

4.5.1.1 Role of Reduced Singular Value Decomposition in SSRT

Reduced Singular Value Decomposition (SVD) of a matrix M is a factorization of M as following:

$$M_{[m \times n]} = U_{[m \times k]} \Sigma_{[k \times k]} V^t_{[k \times n]}$$

M : $m \times n$ represents a matrix which contains ‘ m ’ documents and ‘ n ’ semantic signatures

U : $m \times k$ represents a matrix which contains ‘ m ’ documents and ‘ k ’ concepts

Σ : $k \times k$ represents a matrix which represents the strength of each concept. Here ‘ k ’ stands for the rank of the matrix

V^t : $k \times n$ represents a matrix which contains ‘ k ’ concepts and ‘ n ’ semantic signatures

In the above decomposition, U matrix gives a relation between the documents and concepts and V^t gives the relation between the concepts and semantic signatures. Reduced SVD was used to prune semantic signatures that do not embody semantic content in them. For each concept in V^t , semantic signatures are sorted in descending order of square of their value. Then, only the semantic signatures which add to seventy percent of the concept are considered. This is done for all concepts in V^t . Distinct semantic signatures that are responsible for the top seventy percent of a concept are retained and others are discarded. Thus semantic signatures are pruned and hits array is reconstructed with the retained semantic signatures. This process is repeated iteratively.

The Problem of redundant relevant semantic signatures:

Initially, refinement is done iteratively and this criterion proved to be successful in the elimination of undesired semantic signatures. However, at this point, the problem of *redundant relevant data* is encountered. While the strong semantic signatures are retained and weaker ones are discarded, the retained semantic signatures though relevant were redundant dominating semantic signatures weaker than them which also have powerful directions in space. That is, the retained semantic signatures are capturing strong but redundant information. The conclusion is that:

The strongest semantic signatures overwhelmed other good semantic signatures!

The Solution to overcome redundant relevant semantic signatures:

To overcome the problem of redundant relevant semantic signatures, columns of Hits Array (semantic signatures) are clustered to find groupings in the semantic signature. A threshold is fixed and accordingly those many semantic signatures from each cluster. Selection of semantic signatures from each cluster is based on the weight of the semantic signature. Weights are assigned to semantic signatures using Reduced SVD on Hits Array. Sum of squares of n columns of V matrix is the weight assigned to that semantic signature. Here n corresponds to the

number of singular values. The retained semantic signatures are stored in a directory and Hits Array Descriptor File is reconstructed.

4.5.2 Dimensionality Reduction

Automated Learner Tool generates clusters with the document vectors from the training document. Then, all these clusters are saved as semantic signatures. But, most of them do not capture semantic content in them. This led to the vast generation of semantic signatures which made the tool exposed to *curse of dimensionality* [15]. When dimensionality increases, data becomes sparse. So, it would be difficult to detect the group of similar documents. Hence, dimensionality reduction is used to downsize the data. Irrelevant dimensions may add noise and thus deteriorate the performance of the tool. Dimensionality reduction can be achieved using many ways. Here, we achieve by following feature selection which tries to select an optimal subset of attributes from the original attributes [16].

Reduced SVD is applied on hits array which decomposes into U , Σ and V^t . Based on the input from the user about the number of dimensions to be reduced, the matrix U and V are reconstructed by retaining the specified number of columns and rows in matrices U and V respectively. Again the hits array is reconstructed with reduced dimensionality.

4.5.3 Cluster the hits array using K-Means

After refining semantic signatures following the process mentioned in Section 4.5.1.1, hits array is reconstructed. Dimensionality reduction is applied on that hits array and is again reconstructed. Surprisingly, clustering of hits array using the built-in K-Means algorithm to group similar documents yielded the same results in both the cases indicating that the refinement criteria developed was effective in the removal of noise.

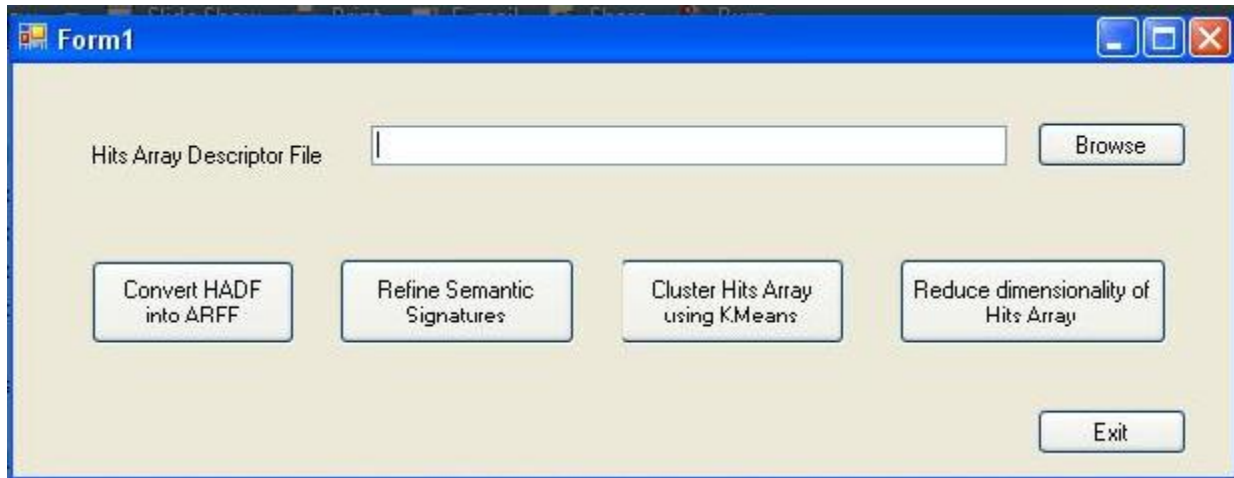


Figure 4.20 Screenshot of Semantic Signature Refinement Tool

4.5.4 Working of Semantic Signature Refinement Tool

Semantic Signature Refinement Tool (SSRT) accepts Hits Array Descriptor File as input. After performing the file existence check, HADF parser is used to extract the hits array, documents and semantic signatures. Using the data extracted from HADF, the user can perform different tasks which include:

- Refining the automatically generated semantic signatures.
- Reducing the dimensionality of the hits array using Singular Value Decomposition.
- Clustering the hits array using KMeans clustering algorithm.
- Converting .hadf file into ARFF in order to be input into WEKA.

If the user selects to *Refine Semantic Signatures* which are generated automatically by ALT, SSRT refines them and reconstructs the HADF. It also saves the retained semantic signatures in a folder.

If the user selects to *Reduce dimensionality of Hits Array*, SSRT applies dimensionality reduction on the hits array using SVD and reconstructs the HADF.

If the user selects to *Cluster Hits Array using KMeans*, SSRT groups the testing documents into clusters by taking information about the type of semantic signatures, the number of clusters and distance measure.

If the user selects *Convert HADF into ARFF*, SSRT converts the HADF into ARFF.

4.5.5 Output of Semantic Signature Refinement Tool

SSRT generates more than one type of output based on the task performed. If the performed tasks are *Refine Semantic Signatures* or *Reduce dimensionality of Hits Array*, the output of SSRT is Hits Array Descriptor File which is explained in detail in Section 4.4.3. If the performed task is *Cluster Hits Array using KMeans*, the output of SSRT is a text file which consists of the file number and the cluster number to which the file belongs to. If the performed task is *Convert HADF into ARFF*, the output of SSRT is in ARFF.

ARFF file is an ASCII file that contains two distinct sections. The header contains the name of the relation, a list of attributes and their type. Each attribute in the data set defines the name and the data type of the attribute. The data section contains the data with data of each instance occupying a row where the attribute values of that instance are separated by commas [19].

```
@relation 'Data Clustering'
@attribute 's4_singers_vocal_throat_1.ssd' numeric
@attribute 's4_sound_vocal_folds_1.ssd' numeric
@data
5 10
2 0
0 11
```

Figure 4.20 Structure of Attribute Relation File Format

Chapter 5: Experiments

This chapter gives an insight into the experiments which were done to study the behavior of the tools in the developed package. Several experiments were performed to benchmark the developed tools. Three experiments are presented in this chapter. Each experiment presented here was done with a goal to evaluate the capability of the developed package in performing its tasks.

5.1.1 Goal of the experiment

To check whether the output of Automated SSMInT is comparable to the output of Analyst driven SSMInT in categorizing documents according to their similarity in content

5.1.2 Corpus Used

The corpus used in this experiment is collected by Barnes, Eschen, Para [5] and Peddada [6] which contains 54 research based papers out of which 9 papers are written by 9 Professors of Lane Department of Computer Science and Electrical Engineering, West Virginia University and remaining 45 papers are the collection of 5 references taken from each of the nine main papers. The 9 papers written by 9 Professors are termed as *main papers* in this document. Even though it is a small corpus, it contains papers with diversified content. As it is a small corpus, a manual analysis of the result is feasible.

5.1.2.1 Notations used

The corpus contains main papers which are written by different professors of Lane Department of Computer Science and Electrical Engineering. As it is difficult to refer and represent the full names of professors every time, two letters of their name are used as abbreviation. Table 5.1 gives information about the professor's names and the abbreviations.

Professor 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	TI
Dr. Mathew Valenti	VA

Table 5.1 Abbreviations for professor's names whose papers are used in this experiment

5.1.3 Design of Experiment

Training Set: The automated tools are trained on nine main papers. The keyword groups and the semantic signatures are extracted from these 9 main papers using AKT and ALT respectively. A total of 1659 keyword groups are generated using AKT. Using ALT 780 semantic signatures are generated. Table 5.2 briefs the inputs and outputs of the training tools of Automated SSMInT.

Name of the tool	Input / Output	Description
AKT	Input	9 main papers
	Output	1659 keyword groups
ALT	Input	1659 KDFs, K-Means clustering with cosine distance as distance measure. Generated document vectors are allowed to be grouped into 2 clusters. All generated clusters are saved.
	Output	780 semantic signatures

Table 5.2 Inputs and outputs of the training tools of Automated SSMInT

Testing Set: The testing set comprised of 45 reference papers. HAGT was made to run on 54 papers against all semantic signatures to produce a hits array which is input to SSRT. SSRT refined the semantic signatures to 90 which still contain signatures from all main papers. Hits array is reconstructed again. The reconstructed hits array is allowed to cluster the documents into 9 clusters using the K-Means Clustering with cosine distance as distance measure. These papers are grouped into clusters with the aid of SSRT and the groupings are studied. Table 5.3 briefs the inputs and outputs of the testing tools of Automated SSMInT.

Name of the tool	Input / Output	Description
HAGT	Input	54 papers and 780 semantic signatures
	Output	Hits Array Descriptor File
SSRT	Input	Hits Array Descriptor File
	Output	Hits Array Descriptor File and 90 potential semantic signatures.
	Input	Hits Array Descriptor File, K-Means clustering with cosine distance as distance measure. Document feature vectors are allowed to be grouped into 9 clusters.
	Output	Clusters of documents.

Table 5.3 Inputs and outputs of the testing tools of Automated SSMInT

5.1.4 Results

The papers in the testing corpus are placed in nine clusters. Table 5.4 gives the information about papers in each cluster. Out of 54 papers, AD’ two of five reference papers and TI’s two of five reference papers did not get any hits. So, HAGT produced document feature vectors only for fifty documents.

	AD	CU	HA	KA	NA	RE	RO	TI	VA
C0									
C1	1				M,1,2,5				
C2	M,2								
C3	3						M,1,2,3,5		
C4		1,4	2,3	M,1,2,3,4,5				1	
C5			M,1,4,5						
C6						M,2,4,5			2
C7						1,3			M,1,3,4,5
C8		M,2,3,5			3,4		4	M,2,3	

Table 5.4 Semantic feature vector clustering results with Spherical K-means

5.1.5 Interpretation

To be able to understand the groupings, I observed the semantic signatures which made them fell together in a cluster. Automated SSMInT found groupings of main and reference papers of different authors who shared content in common. Also, Table 5.5 presents the groupings of documents according to the topic which is done by manually analyzing the outline of the papers.

Topic	Papers related to that topic
Video processing	AD (M,1,2,3,4,5)
Image processing	NA(M,2,5) RO(M,2,3,4,5)
Software reliability	CU(M,2,3,4,5), HA(3)
Software testing	KA(M,1,2,3,4,5)
Software development	HA(M,1,2,4,5) TI(M,1,5)
Wireless communications	RE(M,1,2,3,4,5) VA(M,1,2,3,4,5)
Wireless networks	RE(5) VA(1,3,5)
Spatial diversity	RE(M,1,2)
Co-op diversity	VA(M,2,4) RE(3,4,5)
Object/ target recognition	NA(M,3,5)
Pattern recognition	NA(4) RO(1)
Biometrics	NA(1,2) RO(M,1,2,3,4,5)
Artificial Intelligence	TI(M,1,2,3)

Table 5.5 Groupings of documents according to the topic

All the clusters generated by Automated SSMInT are meaningful groupings of documents and they share a great similarity with the manual analysis of those documents also. For the papers which differ with manual classification, I read the documents and found that internally the paper talks about those concepts.

Cluster 0 was an empty cluster. Cluster 1 has NA's main paper, three of her references and one of AD's references. All these papers are on image processing and most of these papers are on object or target recognition. Cluster 2 has AD's main paper and one of his references. These papers are on video processing. Cluster 3 has RO's main paper, four of his references and one of AD's references. All these papers are on biometrics and image processing. Clusters 1 and 3 are on image processing as a whole. But, they fell into two clusters because papers in cluster 3 are on biometrics also. Cluster 4 has KA's main paper, all her references, two of HA's references, two of CU's references and one of TI's reference. All these papers are on software reliability. Cluster 5 has HA's main paper and three of his references. All these papers are on software development. Cluster 6 has RE's main paper, three of his references and one of VA's references. Cluster 7 has VA's main paper, four of his references and two of RE's reference. Cluster 6 and 7 can be broadly classified as papers about wireless communications. But papers in cluster 7 papers are on wireless networks. Cluster 8 has TI's main paper, two of his references, CU's main paper, three of his references, two of NA's references and one of RO's references. All these papers are on software testing except RO's reference. That paper has only one semantic signature hit which made it fall in the wrong cluster.

5.2.1 Goal of the experiment

The goal of the experiment was to test the semantic sensitivity of Automated SSMInT.

5.2.2 Corpus Used

This corpus was collected by Peddada [6] as a part of her thesis work. This contains ten papers from *throat singing* genre and ten papers from *throat cancer* genre. The papers on each genre may have an overlap of topics related to *singing* and *cancer* and also a paper from *throat singing* genre may talk about *throat cancer* and vice versa.

5.2.3 Design of Experiment

The experiment is done in two stages. First Automated SSMInT is trained on four papers from *throat singing* genre and four papers from *throat cancer* genre (case I). The groupings are observed. Then the experiment is repeated by ignoring most important and distinguishing words in the papers (*singing* and *cancer*) (case II). These groupings are also analyzed.

The training and testing set are represented as Training Set 1 and Testing Set 1. Then, the experiment is repeated by ignoring the most important keywords *singing* and *cancer*. The groupings are analyzed and results are presented in section 5.2.4. The training and testing set in this case are represented as Training Set 2 and Testing Set 2.

5.2.3.1 Case I:

Training Set: The tools in Automated SSMInT are trained using four papers from *throat singing* genre and four papers from *throat cancer* genre. The keyword groups and the semantic signatures are extracted from those eight papers using AKT and ALT respectively. A total of 2319 keyword groups are generated using AKT when five highly weighted words are taken from the TF-IDF list. Forward and backward distances are calculated from each of these words to other words in the window and only five highly correlated words are taken. For each of the highly correlated word, the process is repeated and only the top three correlated words are taken. Using ALT, 410 semantic signatures are generated.

Name of the tool	Input / Output	Description
AKT	Input	8 papers (four papers from <i>throat singing</i> and four papers from <i>throat cancer</i> .)
	Output	2319 keyword groups
ALT	Input	2319 KDFs, K-Means clustering with cosine distance as distance measure. Generated document vectors are allowed to be grouped into 2 clusters. All generated clusters are saved.
	Output	410 semantic signatures

Table 5.6 Inputs and outputs of the training tools of Automated SSMInT

Testing Set: The testing set comprised of twenty papers. HAGT was made to run on 20 papers against 410 semantic signatures to produce a hits array which is to be input into SSRT. Here, the semantic signatures undergo refinement and are reduced to 79 which still contain signatures from all training documents and the hits array is reconstructed again.

Name of the tool	Input / Output	Description
HAGT	Input	20 papers and 410 semantic signatures
	Output	Hits Array Descriptor File
SSRT	Input	Hits Array Descriptor File
	Output	Hits Array Descriptor File and 79 potential semantic signatures.
	Input	Hits Array Descriptor File
	Output	ARFF file

Table 5.7 Inputs and outputs of the testing tools of Automated SSMInT

5.2.3.1 Case II:

Training Set: The tools in Automated SSMInT are trained using four papers from *throat singing* genre and four papers from *throat cancer* genre. In order to evaluate the sensitivity of the tools, the most common words which appear in both the genres are ignored. The keyword groups and the semantic signatures are extracted from these eight papers using AKT and ALT respectively ignoring the keywords *singing* and *cancer*. A total of 2011 KDFs are generated using AKT when five highly weighted words are taken from the TF-IDF list. Forward and backward distances are calculated from each of these words to other words in the window and only four highly correlated words are taken. For each of the highly correlated word, the process is repeated and only the top two correlated words are taken. When ALT is used to extract semantic signatures from the main files using the generated KDFs and K-Means Clustering with cosine distance and

allowing the generated vectors to be grouped into two clusters, 225 semantic signatures are generated.

Name of the tool	Input / Output	Description
AKT	Input	8 main papers (four papers from <i>throat singing</i> and four papers from <i>throat cancer</i>)
	Output	2011 keyword groups
ALT	Input	2011 KDFs, K-Means clustering with cosine distance as distance measure. Generated vectors are allowed to be grouped into 2 clusters. All generated clusters are saved.
	Output	225 semantic signatures

Table 5.8 Inputs and outputs of the training tools of Automated SSMInT

Testing Set: The testing set comprised of twenty papers. HAGT was made to run on 20 papers against 225 semantic signatures to produce a hits array which is to be input to SSRT. Here, the semantic signatures undergo refinement and are reduced to 101 which still contain signatures from all training documents and the hits array is reconstructed again.

Name of the tool	Input / Output	Description
HAGT	Input	20 papers and 225 semantic signatures
	Output	Hits Array Descriptor File
SSRT	Input	Hits Array Descriptor File
	Output	Hits Array Descriptor File and 101 potential semantic signatures.
	Input	Hits Array Descriptor File
	Output	ARFF file

Table 5.9 Inputs and outputs of the testing tools of Automated SSMInT

5.2.4 Results

The generated ARFF file is input to WEKA [30]. Cobweb clustering which is one of the built-in clustering techniques in WEKA [30] is used to analyze outputs in both the cases. Cobweb is used to observe the hierarchical grouping in both the cases.

5.2.4.1 Case I:

Two papers from throat singing genre and one paper from throat cancer genre did not get any hits. Hence, document feature vectors are generated only for 17 papers.

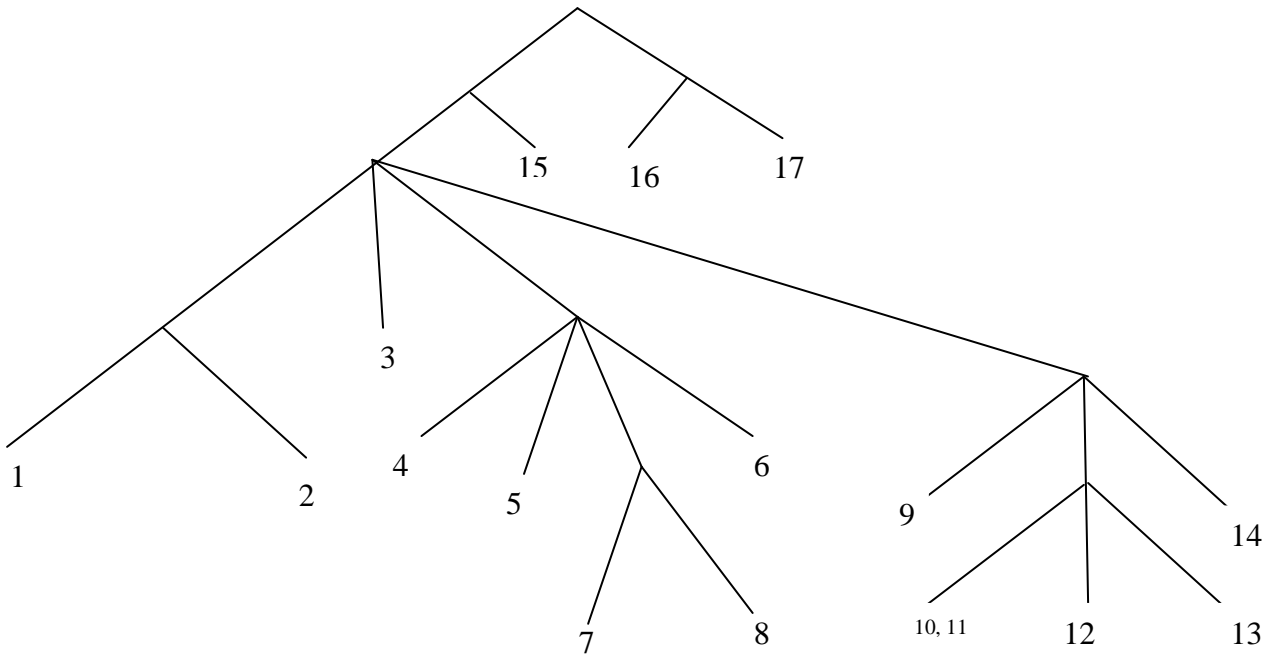


Figure 5.1 Output of Cobweb clustering on the testing corpus

5.2.4.2 Case II:

Two papers from throat cancer genre and three papers from throat singing genre did not get any hits. Hence, document feature vectors are generated only for 15 papers.

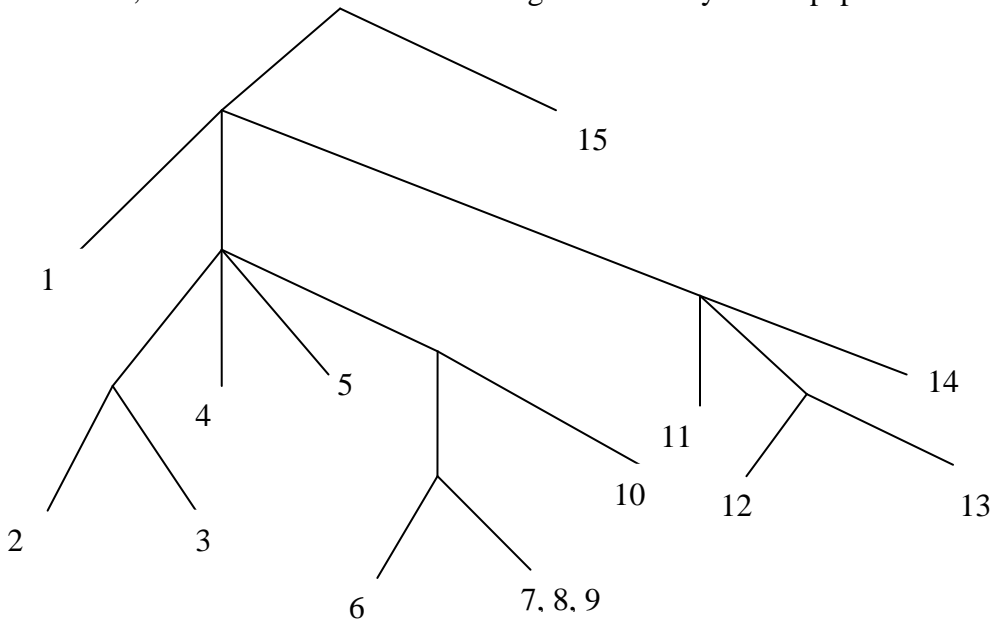


Figure 5.2 Output of Cobweb clustering when the keywords *singing* and *cancer* are ignored

5.2.5 Interpretation

5.2.5.1 Case I:

After analyzing the groupings in Figure 5.1, it is found that cobweb is able to separate throat singing and throat cancer papers using the automatically generated semantic signatures. Papers at nodes numbered 1,2,3,4,5,6,7 and 8 are on throat singing. Though all are on throat singing, cobweb with the help of potential semantic signatures is able to generate internal groupings in these papers according to their content. Papers at nodes 9, 10, 11, 12, 13 and 14 are on throat cancer except paper 11 which is a paper on throat singing. Paper 11 fell into a wrong cluster as the document feature vector generated for that paper was very weak with two semantic signature hits. Papers 15, 16 and 17 are papers which are related to *cancer* and are not especially on *throat cancer*. Even though there is a mention about *throat cancer* in those papers the main content of those papers are about the causes of *cancer* in general.

5.2.5.2 Case II:

By ignoring the words singing and cancer, the tools identified internal topics and grouped the papers accordingly. The automated tools identified subgroups within *throat cancer* and *throat singing* papers by discovering similarities in content. We wouldn't have discovered the papers together but Automated SSMinT brought them together. However, the performance of this package is not as good as the performance of Analyst Driven SSMinT package for this case in the experiment [6].

Papers at nodes 1, 2, 3, 4, 5 and 6 are on throat singing. Though paper 3 is from throat cancer genre, it is an overview paper which talks about throat stress. Hence, it got classified in throat singing group. Papers 7, 8, 9 and 10 are from the genre throat cancer in women. Papers 11, 12, 13 and 14 are on throat cancer. Paper 15 is a paper on cancer in general and does not talk about throat cancer in specific.

5.3.1 Goal of the experiment

To evaluate the performance of the developed package in retrieving documents which have target content from a corpus of documents with unknown content.

5.3.2 Corpus Used

For training and testing corpus, articles from Reuters Corpus Volume 1 are used [12]. This set contains approximately one year of Reuters wire service articles which contain category tags that are manually added to indicate the type of content or category of each article [5]. Articles in the corpus are formatted using a consistent XML schema which is an open standard conceived within Reuters [25]. Some studies noted that the category assignment may not be consistent throughout the corpus as the coding scheme involved is complex [25]. For experiment purposes, articles in the corpus are preprocessed by stripping out all xml tags. This experiment used articles with the category tags 'GHEA' which stands for General health. This experiment is done taking Reuters classification of categories as bench mark.

5.3.3 Design of Experiment

Training Set: The tools in Automated SSMinT are trained using eighty articles belonging to the category GHEA. As the articles in Reuters corpus are short, one or two articles wouldn't be enough to extract keyword groups/ semantic signatures. Hence 80 articles are placed in a folder and are merged into a single file. These articles belong to 21st to 23rd August 1996. The keyword groups and the semantic signatures are extracted from these eighty articles using AKT

and ALT respectively. A total of 4560 keyword groups are generated using AKT when ten highly weighted words are considered from the TF-IDF list. Forward and backward distances are calculated from each of these words to other words in the window and only five highly correlated words are taken. For each of the highly correlated word, the above process is repeated and only the two highly correlated words are taken. 188 semantic signature clusters are generated from the training files by ALT.

Name of the tool	Input / Output	Description
AKT	Input	80 articles concatenated into a single file.
	Output	4560 keyword groups
ALT	Input	4560 KDFs, K-Means clustering with cosine distance as distance measure. Generated document vectors are allowed to be grouped into 2 clusters. All generated clusters are saved.
	Output	188 semantic signatures

Table 5.10 Inputs and outputs of the training tools of Automated SSMInT

Testing Set: The testing set comprised of 800 articles. These articles belong to 24th to 25th August 1996. HAGT was made to run on 800 articles against 188 semantic signatures to generate a hits array. Semantic signatures are refined and hits array is reconstructed by SSRT.

Name of the tool	Input / Output	Description
HAGT	Input	800 articles and 188 semantic signatures
	Output	Hits Array Descriptor File
SSRT	Input	Hits Array Descriptor File
	Output	Hits Array Descriptor File and 90 potential semantic signatures.

Table 5.11 Inputs and outputs of the testing tools of Automated SSMInT

5.3.4 Results

Table 5.12 summarizes the result of the experiment in a tabular format for easy interpretation.

Description	No. of Articles
Articles with Reuters classification as GHEA	14
Articles retrieved by Automated SSMInT	23
Articles with Reuters classification as GHEA and retrieved by Automated SSMInT	6
Articles with Reuters classification as GHEA but are not retrieved by Automated SSMInT	8
Articles retrieved by Automated SSMInT but are not classified as GHEA in Reuters corpus	17

Table 5.12 Semantic feature vector clustering results with Spherical K-means

5.3.5 Interpretation

Articles having Reuters classification as GHEA but are not retrieved by Automated SSMInT and articles without Reuters classification as GHEA but are retrieved by Automated SSMInT are manually analyzed. This manual analysis led to the discovery of some interesting observations. The observations made from this experiment are:

- 17 articles are retrieved by Automated SSMInT even though they did not have GHEA tag in the article metadata. The quality of retrieved articles is high as most of the articles are related to health. These articles can have GHEA tag on them. This concludes that some articles in Reuters database could have additional tags added to them.
- 8 articles are not retrieved by Automated SSMInT even though they had GHEA tag in the article metadata. Some of these articles aren't really related to health topic. This noted that Reuters classification isn't that good. This is also noted by some studies [25].
- Some articles are not retrieved by Automated SSMInT even though they are related to general health because the training set is not sufficiently large to capture features from every health related topic. So, the topics which are not targeted by the training set are not retrieved.
- Even though semantic signatures are extracted from articles with Reuters classification as GHEA, surprisingly, most of the semantic signatures which retained the refinement process are successful in capturing health topics. So, an article which is misclassified as GHEA by Reuters failed to generate a potential semantic signature from it.

Chapter 6: Conclusions and Future Work

Automated SSMInT is capable of grouping documents of similar content and can also retrieve documents related to a particular topic from a large pool of documents by *semantic signature discovery*. The power of the tool is dependent on the training set provided. Automated SSMInT builds a library of semantic signatures automatically and refines noisy and relevant but redundant semantic signatures from the library. It also discovers semantic signatures which the analyst might have missed due to his bias. In Analyst Driven SSMInT, human analyst carefully identifies potential words in the document to be used as keywords and also selects clusters which can be formed as semantic signatures. But, in Automated SSMInT, there is no analyst to perform these tasks. Analyst's absence is subsided by the use of TF-IDF and KGG algorithm in AKT and by vector refinement and cluster selection based on component weight in ALT. By weighing words in the training document based on term frequency and inverse document frequency calculations, TF-IDF identifies potential words in the document that can be used as keywords. Then by using Keyword Group Generation algorithm, keyword groups are generated automatically. By retaining the vectors which have interactions between all keywords in the group, vector refinement substitutes for analyst's absence in Learner Tool. To avoid calculating hits multiple times, Hits Array Generator Tool is developed. Finally, to overcome the problem of relevant but redundant data, hits array is clustered and with the help of SVD potential semantic signatures are discovered.

The experiments presented above stresses one thing, Automation of SSMInT worked the way we thought! It is successful in generating potential keyword groups and discovering strong semantic signatures. The output generated by this package is as good as the output generated by Analyst Driven SSMInT. First experiment shows that Automated SSMInT can categorize documents with similar content into meaningful groups accordingly. Second experiment shows that the package can discover groupings in documents with very similar content which the human analyst may overlook due to bias. The package discovered certain groupings which the analyst might not have discovered. Third experiment proves the strength of the tool in handling a corpus of articles. It is successful in retrieving articles related to a particular topic targeted by the semantic signatures from a large pool of articles.

Future Work

In order to get to a point to test whether the developed package works or not, we have left so many things undone. The work is still in progress. This thesis mainly presents the technical details behind the automated discovery of semantic signatures. Future work may include but not limited to:

- The quantitative measurement of the quality of tools in the developed package needs to be assessed.
- Tools in Automated SSMInT are capable of processing unstructured text which is in text, html or xml formats. The functionality of the tools can be extended by making them capable of processing text which is in other formats like pdfs, dynamic web pages etc.
- Better heuristics can be used for the generation of keyword groups. KGG algorithm generates three lists of words while forming keyword groups. Words in the first list are ranked according to the weights calculated using TF-IDF. The words in second and third lists are ranked based only on the proximity heuristics. This can be extended by calculating the second and third lists of words by using TF-IDF and proximity heuristics.
- The developed semantic signatures are capable of capturing documents having targeted content in the pool of documents with unknown content. But, it cannot capture the emotion of the text. This work can be extended by making semantic signatures to capture emotive shift in the text by ranking the words according to their emotional value.
- KMeans clustering is being used in the tools. Though it is simple, it is an unstable clustering algorithm. Implementation of other clustering algorithms can also be embedded in the tools.
- With the automation of semantic signature generation, numerous semantic signatures are generated in no time. As the refinement of semantic signatures is not done at that point of time, hits array generated will be very sparse. So, implementation of sparse SVD in SSRT would help in the improvement of efficiency and performance of the tool.
- Heuristics can be developed to calculate number of clusters and for selecting a fixed number of semantic signatures from each cluster in SSRT while refining semantic signatures.
- As of now all generated clusters in ALT are saved as semantic signatures. But, a cluster selection criteria needs to be designed in ALT
- Dimensionality reduction in SSRT is achieved by feature selection with the help of Singular Value Decomposition. To reduce dimensionality, feature extraction can also be done. This can be achieved by the implementation of Principal Component Analysis in SSRT.
- Automated SSMInT should be capable of grouping documents from foreign languages based on their semantic content.

Appendix A: List of Stop Words

A	before
about	beforehand
above	behind
across	being
after	below
afterwards	beside
again	besides
against	between
all	beyond
almost	bill
alone	both
along	bottom
already	but
also	by
although	call
always	can
am	cannot
among	cant
amongst	co
amongst	computer
amount	con
an	could
and	couldnt
another	cry
any	de
anyhow	describe
anyone	detail
anything	do
anyway	done
anywhere	down
are	due
around	during
as	each
at	eg
back	eight
be	either
became	eleven
because	else
become	elsewhere
becomes	empty
becoming	enough
been	etc
even	i
ever	ie

every	if
everyone	in
everything	inc
everywhere	indeed
except	interest
few	into
fifteen	is
fify	it
fill	its
find	itself
fire	keep
first	last
five	latter
for	latterly
former	least
formerly	less
forty	ltd
found	made
four	many
from	may
front	me
full	meanwhile
further	might
get	mill
give	mine
go	more
had	moreover
has	most
hasnt	mostly
have	move
he	much
hence	must
her	my
here	myself
hereafter	name
hereby	namely
herein	neither
hereupon	never
hers	nevertheless
herself	next
him	nine
himself	no
his	nobody
how	none
however	noone
hundred	nor

not	someone
nothing	something
now	sometime
nowhere	sometimes
of	somewhere
off	still
often	such
on	system
once	take
one	ten
only	than
onto	that
or	the
other	their
others	them
otherwise	themselves
our	then
ours	thence
ourselves	there
out	thereafter
over	thereby
own	therefore
part	therein
per	thereupon
perhaps	these
please	they
put	thick
rather	thin
re	third
same	this
see	those
seem	though
seemed	three
seeming	through
seems	throughout
serious	thru
several	thus
she	to
should	together
show	too
side	would
since	yet
sincere	you
six	your
sixty	yours
so	yourself

some	yourselves
somehow	top
until	toward
up	towards
upon	twelve
us	twenty
very	two
via	un
was	under
we	using
well	being
were	
what	
whatever	
when	
whence	
whenever	
where	
whereafter	
whereas	
whereby	
wherein	
whereupon	
wherever	
whether	
which	
while	
whither	
who	
whoever	
whole	
whom	
whose	
why	
will	
with	
within	
without	

Bibliography

- [1] Sanchez, D.; Martin-Bautista, M.J.; Blanco, I.; Torre, C.; , "Text Knowledge Mining: An Alternative to Text Data Mining," *Data Mining Workshops, 2008. ICDMW '08. IEEE International Conference on* , vol., no., pp.664-672, 15-19 Dec. 2008 doi:10.1109/ICDMW.2008.57
- [2] Marti Hearst, "What is Text Mining?", 2003;
<http://people.ischool.berkeley.edu/~hearst/text-mining.html>
- [3] Ricardo BY, Berthier RN. Modern Information Retrieval, New York: Addison-Wesley, ACM Press, 1999. Page 19-34
- [4] Seth Grimes. Unstructured data and the 80 percent rule. Experts Corner: Seth Grimes, Clarabridge Bridgepoints, Issue 3, 2008, Q3 2008. White Paper.
- [5] Uday Kiran Para, "Computer-aided semantic signature identification and document classification via semantic signatures," master's thesis, Dept. Computer Science and Eng., West Virginia University., 2010.
- [6] Sri Ramya Peddada, "Sensitivity of semantic signatures in text mining," master's thesis, Dept. Computer Science and Eng., West Virginia University., 2010.
- [7] Julie Beth Lovins (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics* **11**:22–31.
- [8] Kenneth W. Church, William A. Gale Inverse Document Frequency (IDF): A Measure of Deviations from Poisson, *Natural language processing using very large corpora*, Kluwer Academic Press, Boston (1999), pages 283–29
- [9] M. F. Porter. (1980, 07). An algorithm for suffix stripping. *Program* *14*(3), pp. 130-7.
- [10] Armand Brahaj, "List of English stop words," blog 14th Apr 2009;
<http://armandbrahaj.blog.al/2009/04/14/list-of-english-stop-words/>
- [12] 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.* 5pp. 361-397.

[13] Laurens van der Maaten, Eric Postma, Jaap van den Herik, “Dimensionality Reduction: A Comparative Review” TiCC TR 2009–00

http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction_files/TR_Dimensiereductie.pdf

[14] G. Holmes; A. Donkin and I.H. Witten (1994). "[Weka: A machine learning workbench](#)". *Proc Second Australia and New Zealand Conference on Intelligent Information Systems, Brisbane, Australia*. Retrieved 2007-06-25.

[15] Richard Ernest Bellman; Rand Corporation (1957). *Dynamic programming*. Princeton University Press. [ISBN 9780691079516](#). Republished: Richard Ernest Bellman (2003). *Dynamic Programming*. Courier Dover Publications. [ISBN 9780486428093](#).

[16] P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice/Hall International, 1982

[17] Elena Deza & Michel Marie Deza (2009) *Encyclopedia of Distances*, page 94, Springer.

[18] P.-N. Tan, M. Steinbach & V. Kumar, "Introduction to Data Mining", Addison-Wesley (2005), [ISBN 0-321-32136-7](#), chapter 8; page 500

[19] Witten, I. H. *Data Mining: practical machine learning tools and techniques*/ Ian H. Witten, Eibe Frank – 2nd ed.

[20] S. Weiss, N. Indurkha, T. Zhang and F. Damerou. (2004, October). *Text Mining: Predictive Methods for Analyzing Unstructured Information* Available: <http://www.worldcat.org/isbn/0387954333>

[21] Williams GD, Dufour M, Bertolucci D. Drinking levels, knowledge, and associated characteristics, 1985 NHIS findings. *Public Health Rep.* 1986;101:593–598.

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

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

- [24] 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.* 5pp. 361-397.
- [25] Tony Rose, Mark Stevenson, Miles Whitehead, "The Reuters Corpus Volume 1 - from Yesterday's News to Tomorrow's Language Resources"
- [26] Q. Wu, E. Fuller and C. Zhang. (2010), "Graph model for pattern recognition in text," in *Studies in Computational Intelligence* (V.288 ed.), I. Ting, H. -. Wu and T. -. Ho, Eds.
- [27] Q. Wu, E. Fuller and C. Zhang. (2009), Text document classification and pattern recognition. *Social Network Analysis and Mining, International Conference on Advances in Opp.* 405-410.
- [28] MacQueen, J. B. (1967). "[Some Methods for classification and Analysis of Multivariate Observations](#)". Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. pp. 281–297. [MR0214227.Zbl 0214.46201](#). Retrieved 2009-04-07.
- [29] Wall, Michael E., Andreas Rechtsteiner, Luis M. Rocha (2003). "[Singular value decomposition and principal component analysis](#)". In D.P. Berrar, W. Dubitzky, M. Granzow. *A Practical Approach to Microarray Data Analysis*. Norwell, MA: Kluwer. pp. 91–109.
- [30] Ian H. Witten; Eibe Frank, Mark A. Hall (2011). "[Data Mining: Practical machine learning tools and techniques, 3rd Edition](#)". Morgan Kaufmann, San Francisco. Retrieved 2011-01-19.
- [31] Bryan A. Garner, "Exercises from Legal writing in Plain English", 2001;
<http://press-pubs.uchicago.edu/garner/documents/section6.html>
- [32] Pyle, D., 1999. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers, [Los Altos, California](#).
- [33] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, "Data Preprocessing for Supervised Learning", *International Journal of Computer Science*, 2006, Vol 1 N. 2, pp 111-117.