CHAPTER 1

INTRODUCTION

In Vector Space Model(VSM), each document is defined as multidimensional vectors of keywords in Euclidean space whose axis corresponds to the keyword. For a document, the keywords are extracted from the document and weight associated with each keyword determines the importance of the keyword in the document and the collection of the documents. Thus, a document is represented as
$di = \{w_{i1}, w_{i2}, w_{i3} ...., w_{ij}\}$ [1].

Similarly, The VSM relies on three sets of calculations. This model can work on selected index of words or no full text. The calculations needed for the vector space model are:

1. The weigth of each indexed word across the entire document set needs to be calculated. This answers the question of how important the word is in the entire collection.
2. The weight of every index word within a given document(in the context of that document only) needs to be calculated for all D documents. This answers the question of how important the word is within a single document.
3. For any query, the query vector is compared to every one of the document vectors. The results can be ranked. This answers the question of which document comes closest to the query, and ranks the others as to the closeness of the fit.

The weight can be calculated using this equation:

Eq1:

$Wi = tf \times idf = tfi \times log(D/dfi)$

Where:

1. *tfi*=term frequency(term counts) or number of times a term i occurs in a document. This accounts for local information.

2. *dfi*=document frequency or number of documents containing term i.
3. *Idf*=inverse document frequency
4. *D*=number of documents in a database[3].

A new approach is adapting fuzzy semantic for extending *tf-idf* to fuzzy *tf-idf*. The fuzzy *tf-idf* ranks a term's level of importance and semantic relevancy. Therefore, fuzzy *tf-idf* is effective in exploring hidden relationships between text documents. Using fuzzy *tf-idf* and Vector Space Model to cluster the texts into small number of groups is one key step in getting an overall idea of a text document, especially when dealing with a high number of documents.

The fuzzy semantic search was shown to have an average improvement over the conventional search of upto 50%[4]
The application of fuzzy set with its membership function,for vector generated from the documents,can be taken as the new vector which possess the semantic values and after calculating the distance, the cluster obtained can be taken as the semantically similar clusters.
The distance can be either Euclidean, or Cosine distance. But here the Cosine distance is taken as the experimental purpose.

1.1 **Motivation**

Vector Space Model is a widely used model in information retrieval. The purpose for using Vector Space Model is to perform relevancy rankings and clustering of relevant documents[4].The computation cost and its simplicity have given the motivation for the reseach. The work is done so as to perforam the task related to review mining or opinion mining easily with semantic approach.Vector Space Model is not a complete clusteing algorithms itself but here,in my thesis, most priority is given as Vector Space Model is the main algorithms if we apply fuzzy set approach with it.

1.2 Problem Statement

Clustering the text using the normal vector space model could not handle the semantic relevancy of words. Due to lack of semantic meanings of the texts in Classical Vector Space Model, the concept of Enhanced Vector Method is proposed. The research has not been performed yet in text mining for Nepali Language which is the leading task for Nepali researcher who wants to work in Nepali language for the text clustering. The algorithms which work in English language may not work in other language. It is due to language dependent algorithms. The clustering task enables the analyst to observe those clusters having maximum number of documents. The analyst can only analyse those clusters having maximum documents which reduces the time for busy people.

1.3 Research Question

What is the relationship between the documents and the number of clusters produced? What is relationship between the performance and the number of documents? What is the value of the individual keyword in the document for clustering process? These relationship are analyzed in this research using huge number of documents to be tested. Also the documents used are taken from variant structure which may be topic related or non topic related for the anlysis purpose.

1.4 Organization of thesis

Chapter 2 gives a brief information on existing studies in the literature related with text similarity. Chapter 3 gives the detail about the similarity of texts. Chapter 4 gives the detail about the proposed method. Chapter 4 gives the information about the result and Chapter 5 gives the conclusion. Lastly it contains the reference, and it follows appendices which includes the source code.

CHAPTER 2

LITERATURE REVIEW

This chapter contains information about the existing related works in the literature conducted by researcher so far. These studies are categorized in three related domains. They are listed as below

- Text similarity studies in the literature
- Application areas of text similarity
- Testing similarity of words.

3.1 Text similarity studies in the literature

There exist several studies on computer-based word similarity in the literature. These studies are grouped into two categories. They are

- Vector Space Model
- Non Vector Space Model or other combined with Vector Space Model.

3.1.1 Vector Space Model

[1, 2] used different important algorithms for clustering of blog information. It includes FCM, LSA, and Classical VSM. [3] Used modified vector space model for protein retrieval. They enhanced of an existing method of information retrieval which is a modification of Vector Space Model for information retrieval. They modified the term frequency and document frequency concept. This enhanced model is modified to be applied on protein sequence data. Similarly, [4] analyzed the blog using vector space model along with the grouping rule. They also applied fuzzy set concept. Using Vector Space Model, [5] performed word similarity using context vector space model which used variant of classical vector space model and corpus available.

3.1.2 Non Vector Space Model or combined with vector space model

For the clustering of texts [5, 7] used WORDNET, a dictionary and graph based model for the semantic similarity of words. They made the graph and calculated the similarity of two given words using their function.Similarly, [6] used WORDNET for text categorization purpose. They calculated synonyms using that dictionary. [8] Used vector space model and ***knn*** for text categorization process. [9] Discussed about Point wise Mutual Information (PMI) for the semantic similarity between words. It is the unsupervised approach and the search engine was used for this purpose. [10] used corpus and depending on that corpus, they calculated the string similarity of the documents.

3.2 Application areas of text similarity

Text or words are the main building blocks of the communication among human beings. Therefore they are very important in a human life. As the technology evolves, words are being used in many areas especially in computers.

The main application areas of text similarity are:

- Word Sense Disambiguation
- Search Engine
- Opinion Mining
- E-commerce

3.2.1 Word Sense Disambiguation

WSD (Word Sense Disambiguation) is a hot topic in the field of Computer Science and it is much related with text similarity [6]. Defining which sense of the word is used in a sentence is very difficult. WSD try to solve this problem. In many of these studies, the main idea is finding the most similar word to the ambiguous word in order to discover which sense of the word is used in the related sentence.

3.2.2 Search Engine

In the search engine, the syntactic constructs are used these days but when the semantic search can be made possible, lots of things can be solved easily. We don't have to bother about the syntax because simply words having similar meaning can easily finds the sites we want to search.

3.2.3 Opinion Mining

Opinion mining means to classify the text documents or opinionated text using some rule. In opinion mining, the users or reviewer post the opinion on a particular things like election post or voting system, then the post or reviews are classified using some rule like text clustering algorithms and analysis is done easily depending on the clustering result.

3.2.4 E-commerce

Like in **Amazon.com**, the products are kept in the websites for selling purpose. These items are delivered to the end user. When the users gets unsatisfied about that product, he/she goes to that sites and comments or reviews are written on that particular items which will be the easiest way for the new consumer to choose the items in a more convenient way depending on that way of reviews written for the item in the sites.

3.3 Testing similarity of texts.

The testing measure is performed using Random Index and confusion matrix process. The programs are written in hypertext preprocessor (php), server side scripting language. And the clusters obtained are taken for performance evaluation purpose. If more clusters have the common elements, then only two clusters having more number of common elements are taken for this purpose because only two clusters at once can be taken for analysis purpose.

CHAPTER 3

MEASURING TEXT SIMILARITY

In this chapter, the concept of our approach is discussed. The concept of semantic (meaning) text clustering is that the cluster are formed according as their respective meanings. The simple idea behind this thesis is that firstly we make a classical vector as obtained in [3]. The relative text or keywords are arranged in the fuzzy set and their respective membership values are calculated using membership function. After calculating the values, the membership values are added to both the document vector and query vector for the respective keywords in the document as well as query. Then, new vector with addition of their respective values obtained from membership function are obtained.After calculating the new vectors, the cosine similarity function is performed to calculate the cluster using threshold values. For measuring text similarity, different parameters are used. The main parameters are:

- Fuzzy set
- Membership function
- Vector Space Model
- Similarity measure functions
- Weight functions

3.1 Fuzzy set

It is a set similar to the classical set theory but with some properties like degree of truthnesss in a set elements are calculated.Suppose, if we want to find the height of a person as degree of height we use the concept of fuzzy set theory.

3.2 Membership function

The membership function is just a function in order to calculate the degree of truthness in the fuzzy set. So as to calculate the degree of truthness of the fuzzy set elements, the membership function is used. Mainly used membership functions are gauss membership function, triangular membership function, trapezoidal membership function , dump-bell membership function, etc. In this thesis, Gauss membership function is used for calculating the degree of truthness of fuzzy set elements.

3.3 Vector Space Model

The vector space model is a widely used model in the information retrieval [4]. In this method, each document is considered as a vector and for the comparison, the query vector is constructed for each document with the help of which the comparison can easily be made using cosine similarity function.

Following diagram can be taken as the exact concept behind the Vector Space Model.

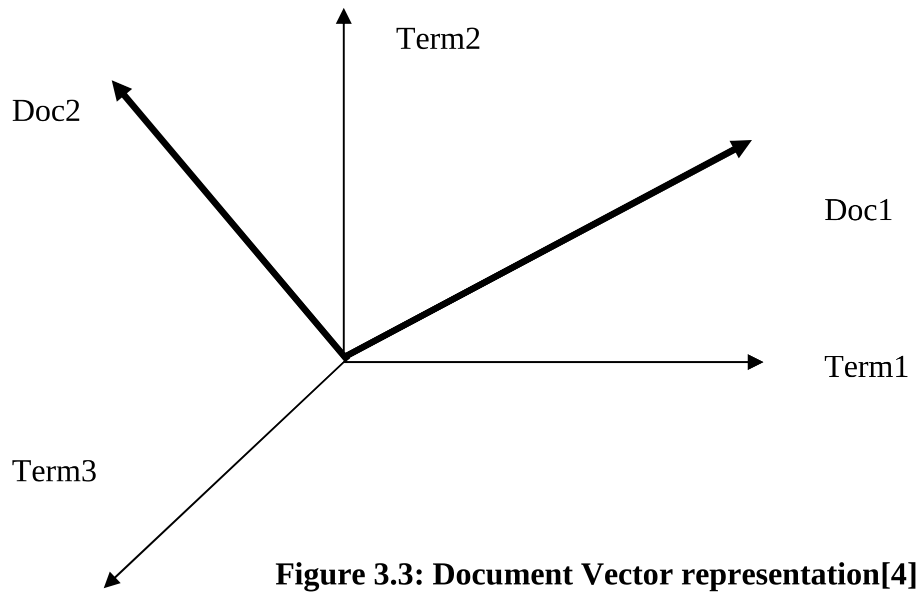


Figure 3.3: Document Vector representation[4]

Here, in the above diagram, thin line represents the term dimension since the number of terms in the documents represents the dimension of the document vector and the thick line represents the documents to be compared. And for calculation of the distance between two document vector, distance formula, Cosine measure was used.

3.4 Similarity measure functions

The main four similarity measure functions described by [6] are:

3.4.1 L1 Norm:

The L1 Norm is called Manhattan or Levensthein distance.

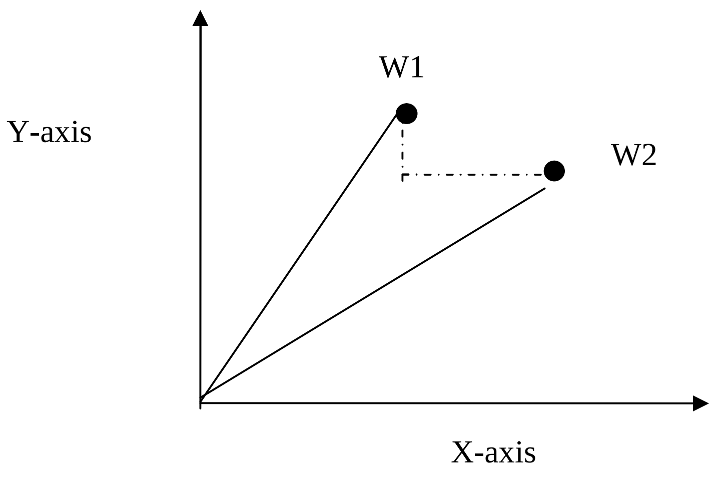


Figure 3.4.1:w1 and w2 are the text vectors, the black line between w1 and w2 is the L1 Norm[6]

The distance between two text vectors are given by

$L_1 Norm = \sum | w_1 - w_2 |$

This method measures the absolute between two vectors dimension-wise. For example A={0,11,1,0,1,1,0,2,2} and B={11,2,6,3,1,2,2,2,1} then Manhattan distance between these two vectors are calculated as followings:

$L_1 Norm$ (A, B)=|0-11|+|11-2|+|1-6|+|0-3|+|1-1|+|1-2|+|0-2|+|2-2|+|2-1|=32

Here the returned value increases as the similarity between two headwords decreases. Therefore there is an inverse ratio between semantic similarity of headwords and the calculated value.

3.4.2 L2 Norrm:

L_2 Norm is also called the Euclidean distance.

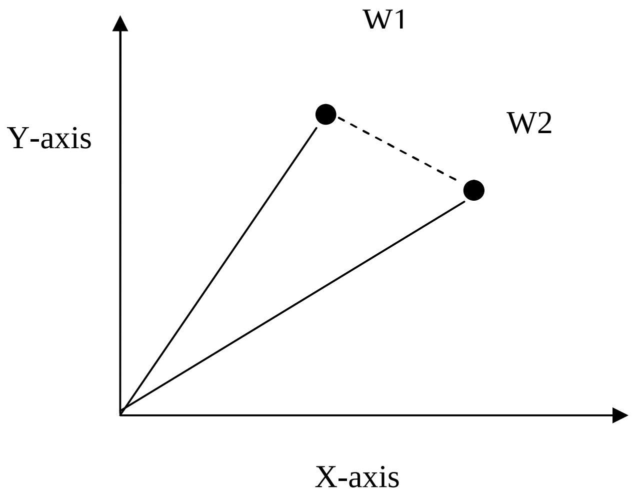


Figure 3.4.2: L2 Norm between two vectors.[6]

If the vectors w1 and w2 are two vectors, then the distance is given by:

$$L_2Norm = \sqrt{\sum (w_1 - w_2)}^2$$

L_2Norm (A, B)=

$$\sqrt{(0-11)^2 + (11-2)^2 + (1-6)^2 + (0-3)^2 + (1-1)^2 + (1-2)^2 + (0-2)^2 + (2-2)^2 + (2-1)^2}$$

=15.5563

It is more or less similar to Manhattan distance.

3.4.3 Cosine

This is an angular distance between two text vectors. Cosine is a function which is heavily used in linear algebra and become a standard function in IR [6]. The thesis is also performed using this distance measure.

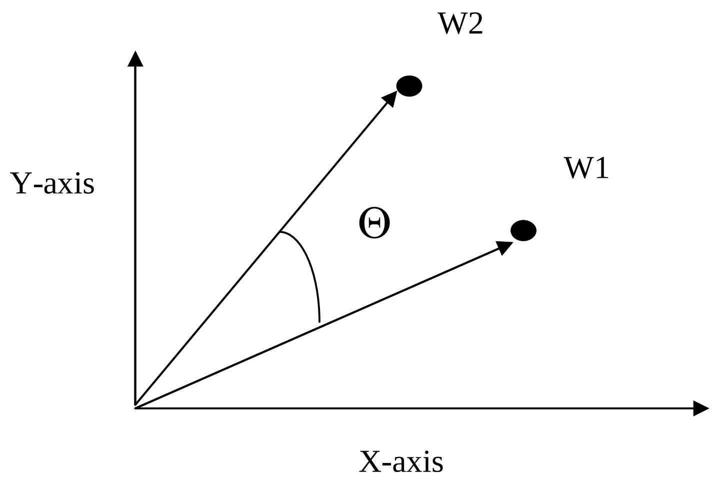


Figure 3.4.3: Showing the angle between two text documents w1 and w2[6]

$$Cos(w_1, w_2) = \frac{\sum(w_1 * w_2)}{\sqrt{\sum w_1^2 . \sum w_2^2}}$$

The distance between w1 and w2 using this measure is given by

Cos (A, B) =

$$\frac{(0*11)+(11*2)+(1*6)+(0*3)+(1*1)+(1*2)+(0*2)+(2*2)+(2*1)}{\sqrt{(0^2+11^2+1^2+0^2+1^2+1^2+0^2+2^2+2^2)*(11^2+2^2+6^2+3^2+1^2+2^2+2^2+2^2+1^2)}}$$

=0.2374

As an angle approaches 1, meaning that the two vectors are getting closer and 0 means not similar.

3.4.4 Jaccard:

Jaccard (from Paul Jaccard 1901) calculates the intersection divided by the union of the text document vectors.

$$Jaccard(w_1, w_2) = \frac{\sum \min(w_{1,} w_2)}{\sum \max(w_1, w_2)}$$

For the two text vectors, the distance is calculated as

Jaccard (A, B) =

$$\frac{\min(0,11) + \min(11,2) + \min(1,6) + ... + \min(2,1)}{\max(0,11) + \max(11,2) + \max(1,6) + ... + \max(2,1)}$$

= 8/40=0.2

3.5 Weight functions

The weight functions may be of two types.

3.5.1 Identity function

This function returns 1 if a relation exists and returns 0, otherwise. The identity property is used for query vector in this thesis.

3.5.1 Frequency

This function counts the number of occurrence in the document. The frequency is used for document vector for the thesis

.

CHAPTER 4

THE PROPOSED SYSTEM

The details of the proposed system are given in the following sections. For the research activities ,90 different test data are used with four related fuzzy sets. The complete four groups of document are given in **APPENDICES**. The documents are of two type i.e Single word documents and multi word documents. And the fuzzy group are not of uniform type. The complete illustration of the proposed system in word level is shown for understanding purpose.

- Fuzzy set
- Membership function
- Vector Generation
- Enhanced Vector Generation
- Cosine Similarity
- Clustering

4.1 Fuzzy set

A fuzzy set is required to demonstrate how to apply fuzzy semantic to *tf-idf*. It is a set which consists of semantically related words are grouped into one group and others in another group. Using this set, we can calculate the degree of truth ness.

For the thesis, four fuzzy sets are used for the activities.For example

```
$arr=array (
        '1'=>array ('ठुलो','बिशाल','भयंकर'),
        '2'=>array ('राम्रो','सुन्दर'),
        '3'=>array ('नराम्रो','रद्दि'),
        '4'=>array ('छ','हो'),
        '5'=>array ('छैन','होईन')
         );
```

Figure 4.1 fuzzy set

It is shown that the semantically related terms like 'ठुलो','बिशाल' and 'भयंकर are kept into one group. Similarly, '2'=>array ('राम्रो','सुन्दर') contains the terms having same meanings. The complete list of fuzzy set used in this thesis are given in APPENDICES.

4.2 Membership function

Fuzzy membership function calculates the degree of truthness from the fuzzy set. Membership function may be of different types like Gaussian membership function, triangular membership function or trapezoidal membership function. But here in order to calculate the vague text, the Gaussian membership function is used.

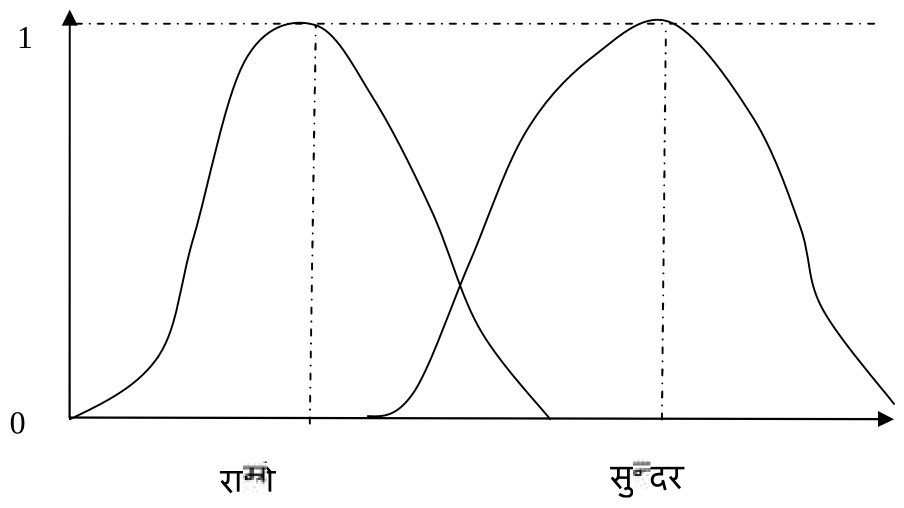


Figure 4.2 Gaussian membership functions

The word “राम्रो” is closer to the word “सुन्दर” in accordance with the fuzzy set description above. Thus, the relevancy score of these two words will be relatively higher. According to [4], Gaussian membership function formula is given by:

Membership function formula:

$$«_{FNS}(z) = \exp(\frac{-(z-\bar{z})^2}{\pm})$$

Where:

z =term to be calculated, whose value is given initially 0

$\bar{z}$ =mean of the Gaussian membership function.

$\pm$ =determines the shape of the Gaussian membership function.

When reducing $\pm$ the shape becomes sharp.

4.3 Vector Generation

This step calculates the vector using normal ***tf×idf*** approach as explained in [3]. For the illustration, let us consider the following documents and their vector generation in word level.

Document	Content
1	ठलो
2	बिशाल
3	सानो
4	सुन्दर

Table 4.3(a) Document to be clustered

Now let us calculate the term frequency of each individual term in a document.

Document	ठुलो
1	1
	बिशाल
2	1
	सानो
3	1
	सुन्दर
4	1

Table 4.3(b) Term-Frequency of the terms

Next phase is the calculation of the inverse document vectors

Inverse document frequency of ‘ठुलो’ is calculated as:

$\log_2$ (4/1, 2) =2. Similarly, the inverse document frequency of other terms is also 2.

Here log is taken as base 2.

Document	IDF weight
ठुलो	2
बिशाल	2
सानो	2
सुन्दर	2

Table 4.3(c) for *idf* calculation

After calculating the *tf* and *idf*, next step is to calculate the *tf-idf* of each document.

Document	*tf*×*idf*
ठुलो	2
बिशाल	2
सानो	2
सुन्दर	2

Table 4.3(d) calculation of *tf*×*idf*

In this situation, although the first documents are of similar type in semantic, they are not in normal traditional *tf*×*idf* concept.

Thus, the document vectors are:

Doc1=<2, 0, 0, 0>

Doc2=<0, 2, 0, 0>

Doc3=<0, 0, 2, 0>

Doc4=<0, 0, 0, 2>

And the query vectors are formed using identity property.

Query1=<1, 0, 0, 0>

Query2=<0, 1, 0, 0>

Query3=<0, 0, 1, 0>

Query4=<0, 0, 0, 1>

4.4 Enhanced Vector Generation

This is the step where actual work is performed. Classical vector space model couldn’t solve the problem of semantic similarity of the documents. So this step is taken as consideration. The calculated vectors can not work easily as we expected because it simply possess the normal vectors from the documents.

So, the document vectors updated from our algorithms are:

Doc1=<2+0.9, 0, 0, 0>

Doc2=<0, 2+0.9, 0, 0>

Doc3=<0, 0, 2, 0>

Doc4=<0, 0, 0, 2>

Here, in the document vectors doc1 and doc2, 0.9 is added because from the fuzzy set, these two terms are taken from Gaussian membership calculation. Using the formula above, we get this value. It is normalized after calculating the vector. Normalization can be done using the formula:

$$\frac{term}{\sqrt{term_1^{\ 2} + term_2^{\ 2} + ...term_n^{\ 2}}}$$

Similarly, for the query vector a little different concept is taken into consideration.

Query1=<1, 0.9, 0, 0>

Query2=<0.9, 1, 0, 0>

Query3=<0, 0, 1, 0>

Query4=<0, 0, 0, 1>

Here, for the query vector, in the first query first position of vector is substituted with 0.9 which is calculated from Gaussian membership function. It is done when both words ‘ठुलो’and ‘विशाल’ lies in the same fuzzy set. Similarly in the query2 same work is done.

4.5 Cosine Similarity

Now after calculation of the query vectors and the document vectors, the cosine similarity is the angular distance between two documents. If the angle between two documents is equal to 0, then it is said that two documents are very dissimilar. It the similarity score is equal or close to 1, then the query and document are very similar.

The cosine similarity between the query and the document are calculated one by one. For example, query1 is performed calculation between doc1, doc2, doc3 and doc4 respectively.

For example, the result after performing the cosine similarity between query1 and doc1 is obtained as 0.74.

4.6 Clustering

This is the final step, where the actual clustering is performed. Here, the clustering threshold 0.50 is taken into consideration. If the cosine similarity falls below greater than 0.50, then they both fall into same cluster. After performing the cosine similarity, the following table is obtained.

The complete algorithms for the proposed model is given in figure..

```
Step1: Calculate the term frequency and inverse document frequency of the
document keywords.
Step2: Calculate the $tf \times idf$ value of each keyword.
Step3: Calculate the value of membership of the term from fuzzy set.
Step4: New vector value=$tf \times idf$+membership value.
Step5: Calculate the cosine similarity of the documents for clustering.
Step 6: If the cosine value>0.50 Then cluster that into base cluster
        Else go to Step5 and perform the same operation with another document
vectors.
Step7: Obain the semantically related clusters.
Step8: End.
```

Figure 4.6(a) Complete algorithms for proposed model.

The operational flow diagram for the proposed model is given by

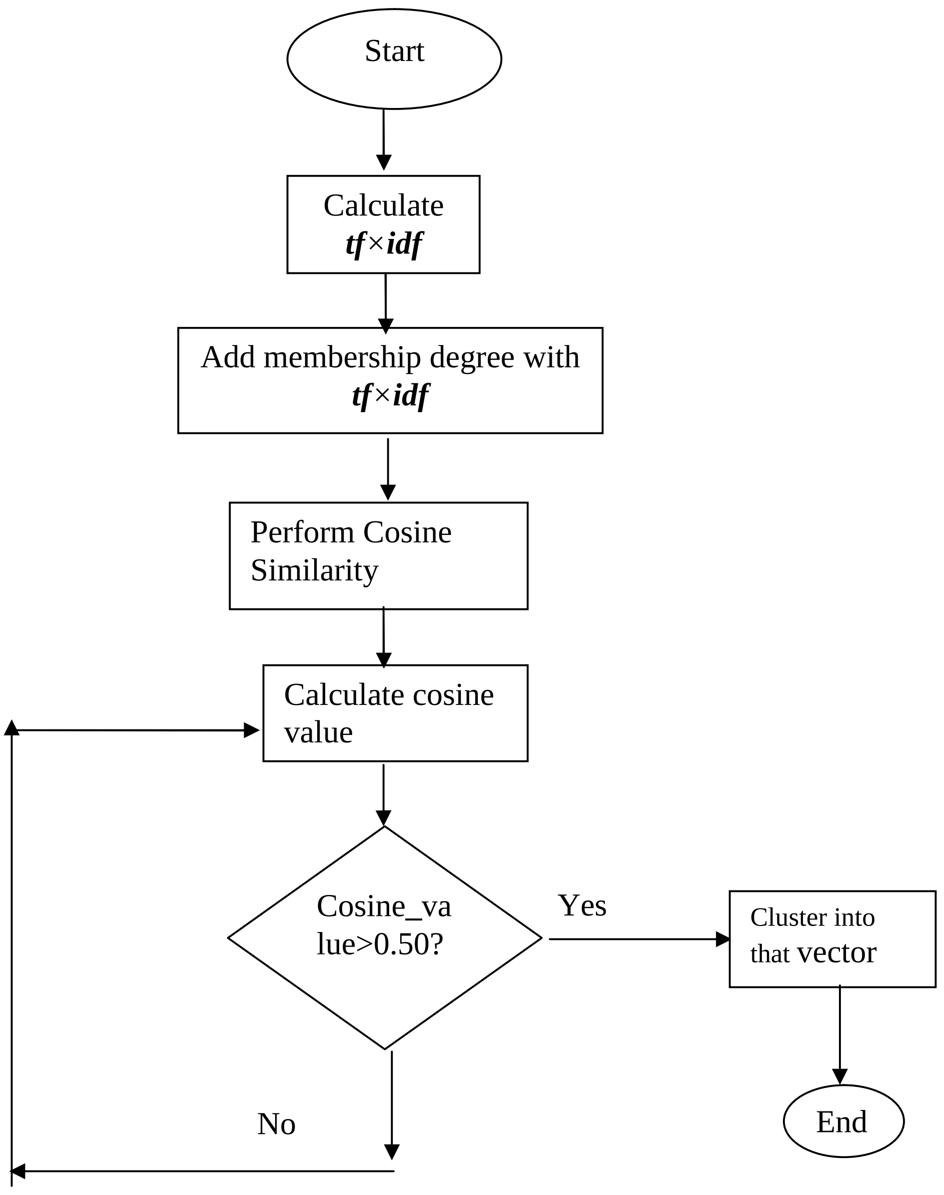


Figure 4.6(b) operational flow of the proposed model.

Figure 4.6(b) is the diagram for showing how the operations are flowed.Through this diagram, it can shown that how the algorithms actually works in real scenario. The document having single word and document with multi keywords are taken for the experiment. The respective fuzzy set are made for the operation.

The complete output for the above documents after programming is shown in Figure 3.1(e).

VECTOR GENERATION STEP

```
Query Vector#1
[1 0.905 0 0 ]
Document#1
[1 0 0 0 ]
Query Vector#2
[0.905 1 0 0 ]
Document#2
[0 1 0 0 ]
Query Vector#3
[0 0 1 0 ]
Document#3
[0 0 1 0 ]
Query Vector#4
[0 0 0 1 ]
Document#4
[0 0 0 1 ]
```

Figure 4.6(c) Vector generated from programming

TABLE FORMATION STEP

0.741	0.671	0	0
0.671	0.741	0	0
0	0	1	0
0	0	0	1

Figure 4.6(d) Table formed after calculating distance between query and document vector

```
Query#1
1 2
Query#2
1 2
Query#3
3
Query#4
4
```

Figure 4.6(e) raw clusters after performing threshold 0.5

```
CLUSTER SHOWN
Cluster1==>1-2-
Cluster2==>3-
Cluster3==>4-
```

Figure 4.6(f) Final cluster

Here, in the above calculation, after generating the vectors i.e. document and query vectors, table is formed to keep the calculated value between the each query with 4 documents one by one. The resulting output is shown in above diagram.

In this way, the proposed system works well in word level and similar operation was applied to sentence level since sentence is formed adding more terms. The performance seems to be better in word level than sentence level.

CHAPTER 5

RESULTS AND DISCUSSION

The results obtained from the above experiment can be studied into following two ways. Four sets of data are used for the performance evaluation of the research performed. 90 Nepali documents (single word and multi-word)are used for the research. 45 documents are of single keyword documents whereas 45 remaining documents are prepared for the multi keyword documents.

5.1 Analysis using Random Index and confusion matrix(Accuracy)

The experimental result of this research is conducted in two ways. Firstly, single keyword documents are taken and second the multi-keyword documents are taken. The performance of the cluster are analyzed by using random index as explained by[7]. The research is implemented in *php* language. Inputs are given using file system. They are kept in the files.

In single word document, the result obtained for two different set of data set which are in Nepali Language

The result obtained from the experiment is shown in Table 1 and Table 2 below.

For single word document

No of documents	Random Index	Accuracy
45	0	0.90

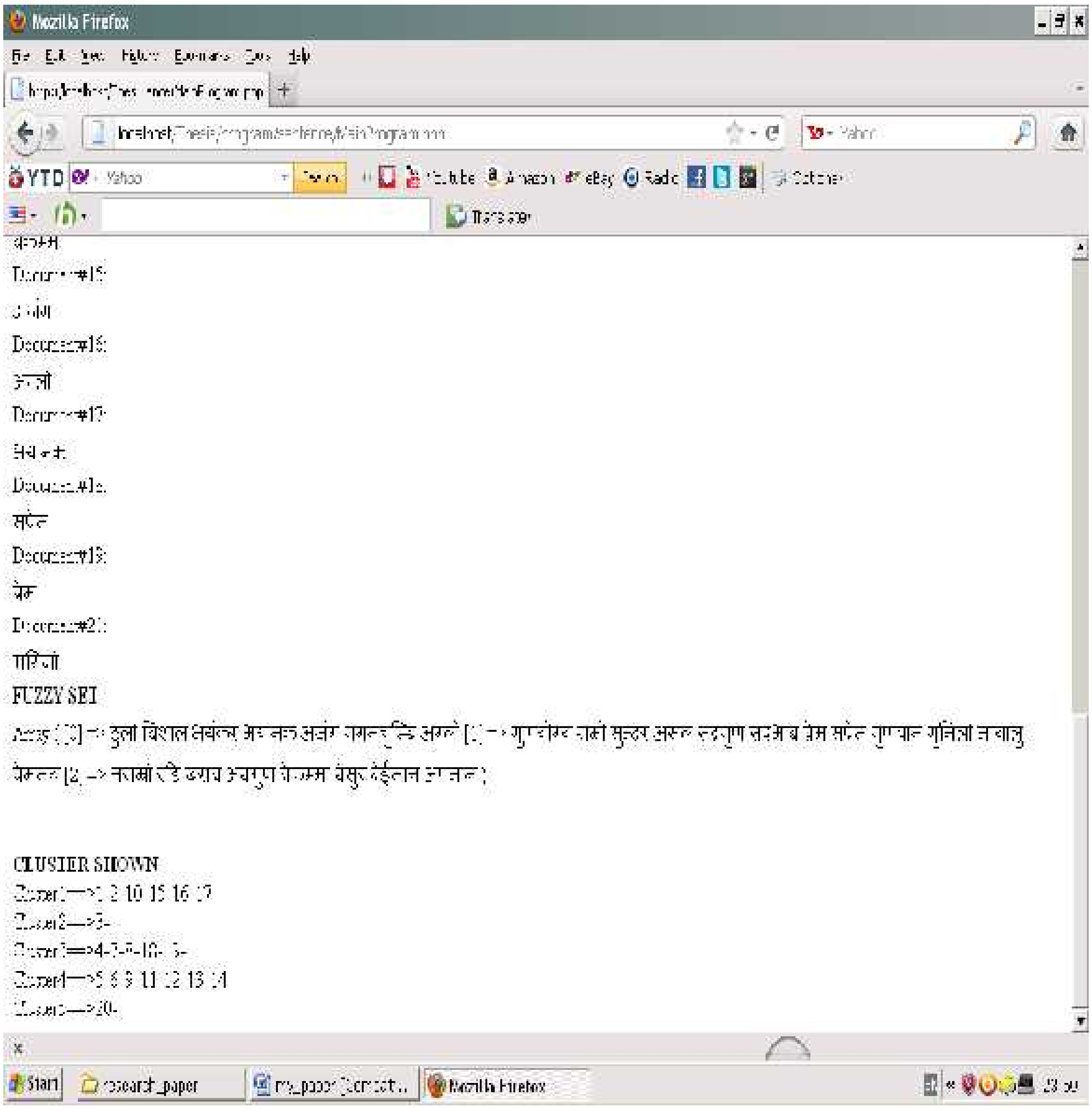
Table 5.1(a):Experimental result for single keyword

Similarly, for the multi word document

No of documents	Random Index	Accuracy
45	0	0.91

Table 5.1(b):Experimental result for multi keyword document

Some of the snapshots of the outputs are given in Figure 4 and Figure 5 as follow.

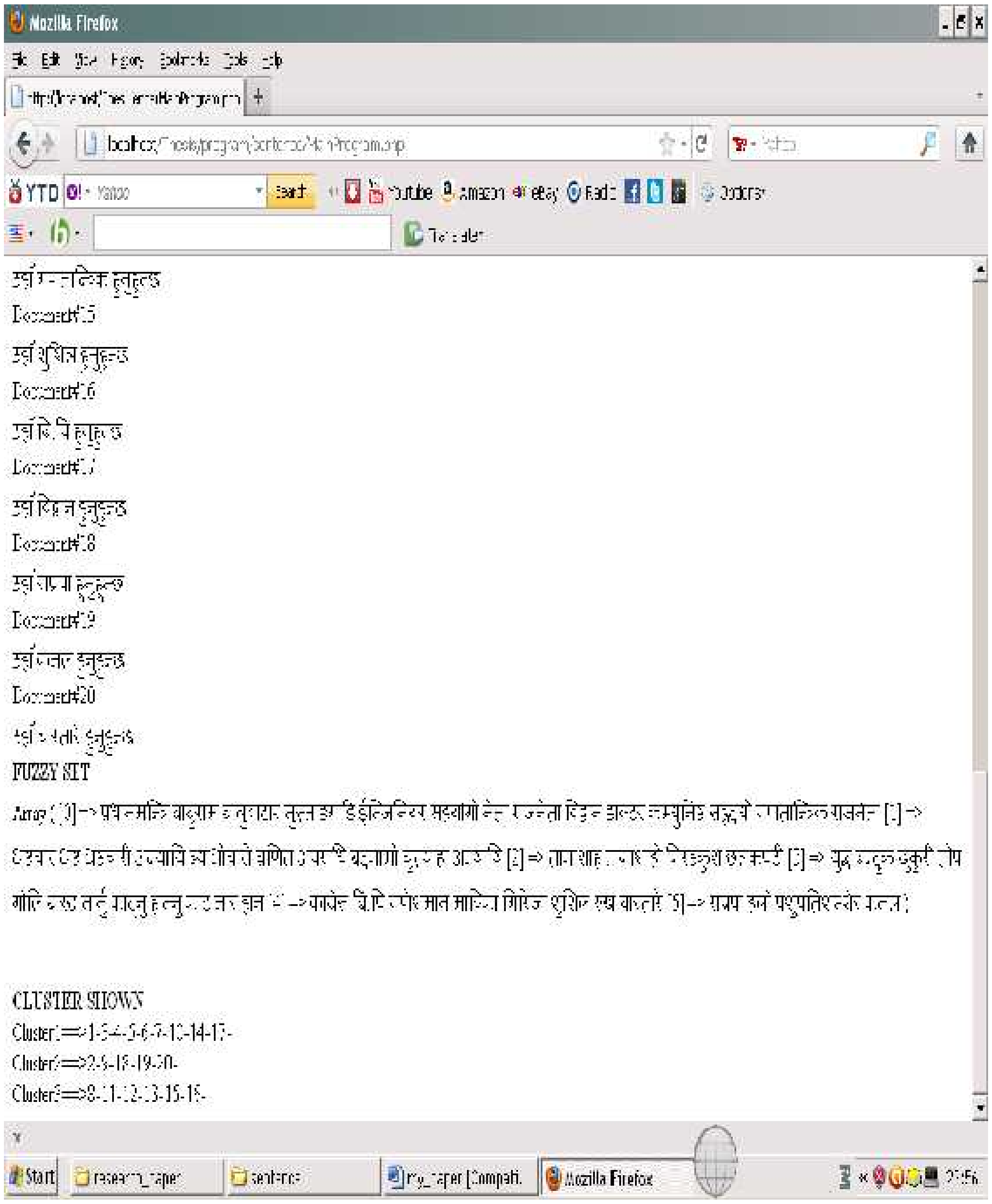


```
[illegible]
Document#15
[illegible]
Document#16
[illegible]
Document#17
[illegible]
Document#18
सचेत
Document#19
प्रेम
Document#20
गरिमा
FUZZY SET
Array ( [0] => [illegible] [1] => [illegible]
[illegible] [2] => [illegible] )


CLUSTER SHOWN
Cluster1--->1 2 10 15 16 17
Cluster2--->3-
Cluster3--->4-7-8-18-19-
Cluster4--->5 6 9 11 12 13 14
Cluster5--->20-
```

Figure 5.1(a): Output obtained for single keyword document

Figure 5.1(a) is the output obtained for the single keyword documents. This output shows that there are no any intersection of the documents so that the value of the random index is zero.



```
[illegible]
Document#15
[illegible]
Document#16
[illegible]
Document#17
[illegible]
Document#18
[illegible]
Document#19
[illegible]
Document#20
[illegible]
FUZZY SET
Array ( [0] => [illegible] [1] =>
[illegible] [2] => [illegible] [3] => [illegible]
[illegible] [4] => [illegible] [5] => [illegible] )


CLUSTER SHOWN
Cluster1==>1-3-4-5-6-7-10-14-17-
Cluster2==>2-9-18-19-20-
Cluster3==>8-11-12-13-15-16-
```

Figure 5.1(b):Output obtained for multi word document

The output obtained using multi keyword is shown in the figure 5.1(b). It has been shown that the algorithms is working properly since it has no intersection of the documents that affects the random index.

5.2 Analysis using Number of Clusters.

The analysis using number of cluster experimented in about 90 documents are as below:

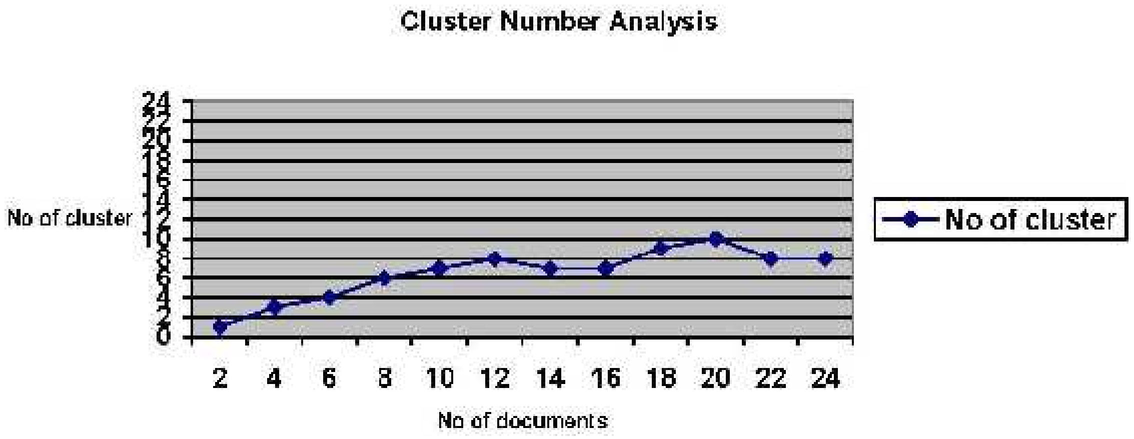


Figure 5.2(a):relation between first 24th documents and number of cluster produced

Here, in figure 5.2(a), it is shown that the number of cluster goes on increasing at some point and it decreases after some point because there may be similar documents after some points due to of which the number of clusters is decreased.

.

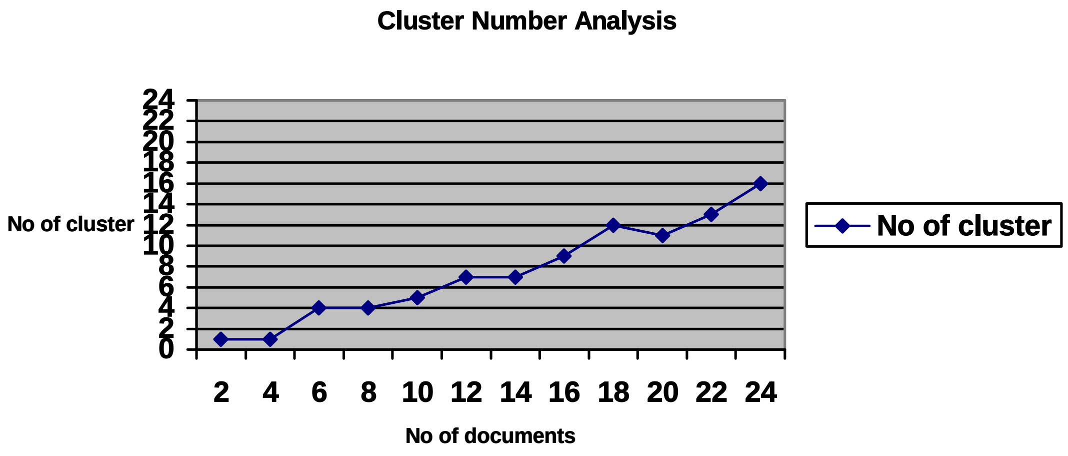


Figure 5.2(b): relation between first 24th documents and number of cluster produced

In the figure 5.2(b), it is seen that the number of cluster goes on increasing rapidly after some point because of the introduction of different types of documents as the number of documents were increases. Up to 16th documents the number of cluster increases slowly but after that number of cluster is increased due to quiet different type of documents .

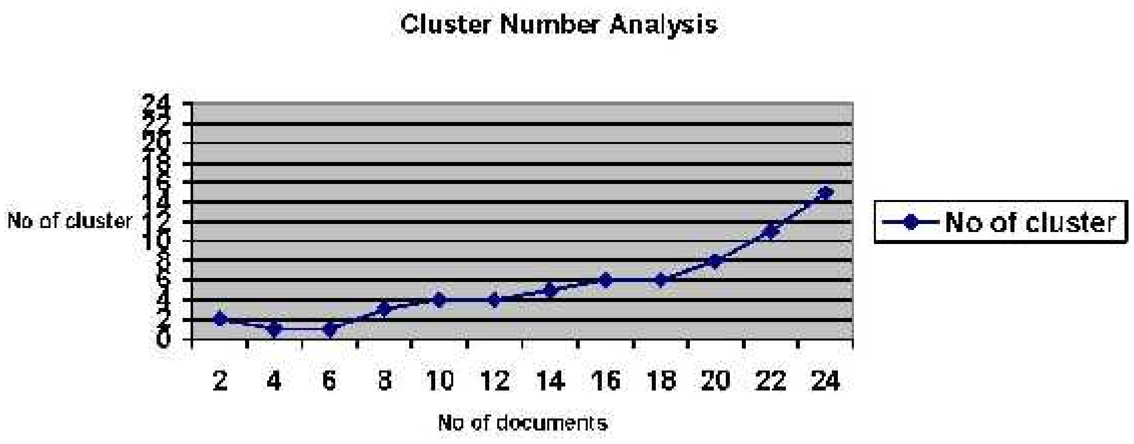


Figure 5.2(c) relation between first 24th documents and number of cluster produced

Seeing in Figure 5.2(c), it can be described that the number of clusters remain minimum up to 18th documents but after that it increases rapidly due to the introduction of quiet different types of data.

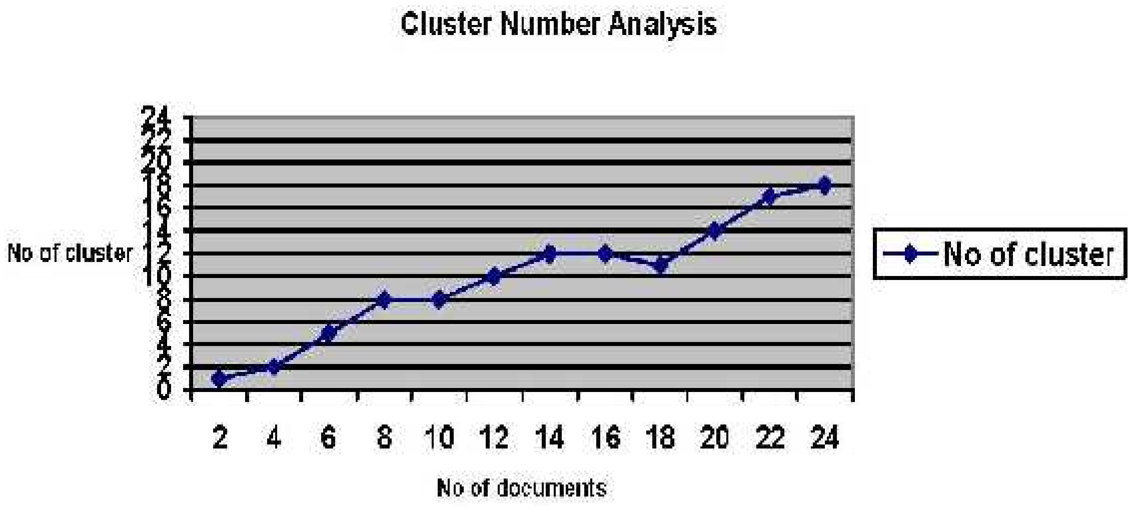


Figure 5.2(d) relation between first 24th documents and number of cluster produced

It is seen in Figure 5.2(d) that there is direct relationship between the number of documents and the number of cluster produced if there is different types of documents which possess few keywords in the fuzzy set. Up to 18th documents, it shows the small changes but after 18th documents the number of clusters increases accidentally.

Hence, the number of cluster can be made minimum if large fuzzy set is used and the structure of the document is made more or less same.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 Summary of the study

In this thesis, semantically similar documents in keyword word level(Single keyword documents and multi keyword documents) are clustered using unsupervised approach i.e Enhanced VSM.

This thesis takes keywords in each document to represent vector elements and then use these vectors to identify document similarities. Classical ***tf×idf*** can not distinguish the semantic words although it is used to keyword's importance. By applying fuzzy semantic to the Vector Space Model , it is able to explore the hidden relationship between documents.

Due to the lack of such features in classical vector space model, the concept of Enhanced Vector Space Model is proposed which produced promising result.

In this methodology, the classical vectors are enriched with the addition of the fuzzy membership values. After enriching the vectors with such values, it becomes ready for normal vector operation which is used in linear algebra. The classical vector is based on ***tf×idf*** approach only and also in this thesis same concept is used with the addition of membership value with it.

Exactly 90th Nepali documents in both single word and multi word level are taken for the experiment. The experimental results are analyzed using Random Index and Confusion Matrix and number of cluster produced.

It can be concluded from this study is that fuzzy set and Vector Space Model, when combined together, performed semantic text clustering causing about 90% accuracy. The time complexity of this algorithms in about O(n×n) where n is the number of documents to be clustered. But for this, it would be better if the number of fuzzy set and number of keywords is increased.

6.2 Limitation and Future works

This new approach has promising result and the result can further be improved with the following future work:

- The large number of fuzzy set can be used. This can be enriched with more semantically related words for each set.

- The number of testing documents can be increased to check its consistency and performance with lots of variants.

- The concept of stop words was not taken into consideration for document processing because of Nepali Unicode which is under research itself.

- The Gaussian membership function is used in this thesis for calculating the degree of truth ness from the fuzzy set. In place of Gaussian membership function, others function like trapezoidal, triangular can be used to check whether it works properly or not.

- The concept can be applied to paragraph level and documents having multiple paragraphs.

- Cosine similarity was used in this thesis. Along with cosine similarity different distance measure can be compared.

- This thesis is not in totally sentence level similarity although it semantically clusters the text of the sentences using the keywords given.

REFERENCES

[1] Jaiswal, Prakash, “Clustering Blog Information”, (2007). *Master’s projects. Paper 36.* http://scholarworks.sjsu.edu/etd_projects/36/

[2] Jain, A.K, Murty, M.N, Flynn, “Data Clustering: A Review”, (1999). ACM Computing Surveys, VOL. 31-No.3.

[3] Abdul-Rub, Mohammed Said, Abdullah, Roshi, Rashid Nur’Ainia Abdul “ A Modified Vector Space Model for Protein Retrieval”, (2007). IJCSNS International Journal of Comptuer Science and Network Security, VOL 7- No.9.

[4] Huo, Chi-Shu, “Blog Analysis with Fuzzy TFIDF”,(2007), Master’s thesis, San Jose State University.

[5] Singh, A.K, Josshi, R.C,, “Clustering of Blogs with Enhanced Semantics”, (2011). International Journal of Computer Applications (0975-8887), VOL. 16-No.7

[6] Esin, Yunus Emre, ‘Improvement of corpus-based semantic word similarity using vector space model”, (2009). *Master thesis.* Middle East University.

[7] Elberrichi, Zakaria, Rahmoun Abdelattif, Bentaalah, Amine, “Using WordNet for Text Categorization”, (2008). The international Arab Journal of Information Technology, VOL 5-No.1.

[8] Shin, Kwangcheol Shin, Abraham ,Ajith, Han, Sang Yong, “Improving kNN Text Categorization by Removing Outliers from Training Set”,(2006). Springer-Velag Belin Heidelberg.

[9] Liu, Bing, “Sentiment Analysis and Subjectivity”, (2010).*Handbook of Natural Language Processing,* Second Edition.

[10] Islam, Aminul, inkpen, Diana, “Semantic Text Similarity Using Corpus-Based Word Similarity and String Similarity”, (July 2008.) ACM Transaction on Knowledge Discovery from Data, VOL. 2, No.2 , Article 10.

APPENDICES

1. Source code for the main program

```
<?php
//define(MAX,4);
include "document_frequency.php";
/**
$links=array('c:/xampp/htdocs/Thesis/program/sentence/1.txt',
        'c:/xampp/htdocs/Thesis/program/sentence/2.txt'
        );



**/
//Enter the number of documents to be clustered//
$MAX=$_POST['size'];
$Data=$_POST['data'];
$Fuzzy=$_POST['fuzzy'];
if($MAX!=null&&$Data!=null &&$Fuzzy!=null)
{
//$MAX=2;

//$directory='c:/xampp/htdocs/Thesis/program/sentence/Data/Data1';
//echo $Data;echo '<br>';

//echo $Fuzzy.'<br>';
//Enter the documents to be clustered.
$links=array();

for($i=1;$i<=$MAX;$i++)
{
        $file=$i."."."txt";
        $links[]="$Data/$file";
}


//echo count($links);
//fuzzy set with the help of which we can clusters the elements
//$Fuzzy='c:/xampp/htdocs/Thesis/program/sentence/Fuzzy_Set/Sample1';
echo '<h1><b>SEMANTIC TEXT CLUSTERING</b></h1>'.'<br>';
$val=array();
$val=calculate_idf($links,$Fuzzy);//call the function for calculating the document
frequency
$cluster=array();
$element=array();
//processing for the clustering
for($i=0;$i<count($links);$i++){
$query=null;
for($j=0;$j<count($links);$j++){
if($val[$i][$j]>0.51)
{
        $element[$i][$j]=$j;
        $query.=($j+1)."-";
}

}//inner for loop close
        $cluster[]=$query;
}//outer for loop close

//after identifying the document similarity
//next stage is to cluster the no. of  documents available for the clustering
//the third and fourth document are only related with themselfes
// so they make cluster them selves.
echo "<br>"."<b>"."CLUSTER SHOWN"."<br>"."</b>";
//make the cluster
$cluster=array_unique($cluster);
```

```
        //print_r($cluster);
        $i=0;
        foreach($cluster as $cluster)
        {
                $i++;
                echo "Cluster".$i."==>";
                echo $cluster;echo "<br>";
        }
        }
        else
        {
                echo " <b><marquee><center>Alert!!!You have not selected the documents
        properly ..please select it again after moving back!!</center></marquee></b>";
        }

?>
```

2. **Source code for document frequency**

```
<?php
/**********************************************
Programmed by:Chiranjibi Sitaula.Email:chiru@cdcsit.edu.np/info@schiranjibi.com.np M.Sc
Thesis. Central Department of Computer Science and Information
Technology, T.U.
Visit http://www.schiranjibi.com.np for further information
*******************************************/
include "tokenize.php";
include "docCounts.php";
include "gauss_set.php";
include "Main_function.php";
include "normalise.php";
include "cosine.php";
include "fuzzyset.php";
function calculate_idf($links,$f_file)
{
 $nodoc=array();//problem occurs when we put this inside for loop
$new1=array();
$new2=array();

echo '<B>DOCUMENTS TO BE CLUSTERED</B>'.'<BR>';
for($i=0;$i<count($links);$i++)
{
//$file=fopen($links[$i],"r");
$token_string=null;
$token_string=file_get_contents($links[$i],NULL,NULL,3);
$nodoc[$i]=$token_string;
//fclose($file);
//print the documents to be cluseterd
echo 'Document#'.($i+1).":".'<br>';
echo $token_string;echo '<br>';
//$token_string=null;
$new1=tokenize($nodoc[$i]);
foreach($new1 as $nw) { $new2[]=$nw; unset($new1);}
}//for loop close
//echo count($nodoc);
$new3=array();
$new4=array();
//tf-idf calculation step;
for($i=0;$i<sizeof($new2);$i++)
{
        //echo $val=docCounts($elements,$nodoc);//it worked properly now.Thnx.
        $new3[$i]=$new2[$i];
        $DFI=docCounts($new2[$i],$nodoc);
        //echo $DFI;
        $IDF=log((sizeof($nodoc)/$DFI),2);
        //in this phase calculate the term frequency
        //echo $IDF;
        $new4[$i]=$IDF;
        if($new4[$i]==0){unset($new4[$i]);unset($new3[$i]);}
```

```
}//for loop

//array declaration step
$tf=array();
$q_vector=array();
$que_vector=array();
$ifarray=array();
$elsearray=array();
$doc_vector=array();
$new_array=array_combine($new3,$new4);
//print_r($new_array);
/*################
Document Vectors Generation
#####################*/
//calculate the gauss value for each term
//$f_file='c:/xampp/htdocs/Thesis/program/sentence/Fuzzy_Set/Sample1';
$count=countFiles($f_file);
//echo $count;
$fuzzyset=fuzzyarray($f_file,$count);
echo '<b>FUZZY SET</b>';
echo '<br>';
print_r($fuzzyset);echo '<br>';

for($i=0;$i<count($nodoc);$i++)
{
        foreach($new_array as $key=>$value)
        {
                 $mem_value= calculate_gauss($key,$count,$fuzzyset);
                $que_vector[]=(calculate_tf($nodoc[$i],$key)*$value)+$mem_value;
        }

        $doc_vector[]=$que_vector;
        unset($que_vector);
```

```
}
//print_r($doc_vector);

//test for group

//this step for the calculation of the enhanced query vector
/*###########
Query Vectors Generation
#############*/
$temp=array();
$new4=array_values($new3);
//print_r($new4);
/** problem occur due to reinitiate the value of the keys from 0 in the newly formed array***/
for($i=0;$i<sizeof($nodoc);$i++)
{
        $tokens=tokenize($nodoc[$i]);
        //test for tokens
        //print_r($tokens);
        for($j=0;$j<count($new4);$j++)
        {

                if(in_array($new4[$j],$tokens))$temp[]=1; //else $temp[]=0;
                else{

                 $else=$new4[$j];
                //loop over $tokens array
                //$value=null;
                foreach($tokens as $element)
                {
                        if(return_group($element,$count,$fuzzyset)==
                        return_group($else,$count,$fuzzyset))
                        {
                                $val=0.91;break;//using break statement
```

```
                              }
                              else $val=0;

                        }
                        $temp[]=$val;
                        }//else close

                        //print_r($temp);

            }
            $qu_vector[$i]=$temp;
            unset($temp);
}
//print_r($qu_vector);

/*##########
printing operations
###########*/
$val=array();
//print_r($new_array);
//perform cosine similarity operation with threshold operation
/*###############
Cosine Operation
###################*/
for($i=0;$i<count($qu_vector);$i++)
{
for($j=0;$j<count($doc_vector);$j++)
{
$val[$i][$j]=cosine($qu_vector[$i],$doc_vector[$j]);
}//first for loop close
}//second loop close
echo "<br>";
//echo '<B>RESULT AFTER COSINE SIMILARITY</B>'.'<BR>';
//print_r($val);
```

```
return $val;
}
?>
```

3. **Source code for fuzzy set**

```
        /****Copyright: info@schiranjibi.com.np*************/
function fuzzyarray($fileDirectory,$count)
{

        for($i=0;$i<$count;$i++){
        $tokens=null;
         $fname=($i+1).".".""txt";
         $fileName="$fileDirectory/$fname";
        $string=file_get_contents($fileName,NULL,NULL,3);
        $f_array[$i]=$string;
        }//for close
        return $f_array;
}
```

4. **Source code for cosine and normalise function**

```
<?php

//calculating the cosine similarity using two  vector space model

function cosine($vector1,$vector2)

{
//in order to find the cosine similarity of two vectors firstly we normalise and them find the
multiplication of two vectors
$mult1=0;
//so multiplying between two vectors
for($i=0;$i<sizeof($vector1);$i++)
{
```

```
$mult1+=$vector1[$i]*$vector2[$i];

}

//calculate the square of two vectors for each vectors
$vect1=0;$vect2=0;
for($j=0;$j<count($vector1);$j++)
{

 $vect1+=$vector1[$j]*$vector1[$j];
$vect2+=$vector2[$j]*$vector2[$j];
}

//now this step is the taking square root of vect1 and vect2
$temp=sqrt($vect1*$vect2);
//this time is the complete calculation of the ration
//echo "Numerator is ".$mult1;echo "<br>";
//echo "Denominator is ".$temp; echo "<br>";
$cosine_value=$mult1/$temp;
//echo "The cosine value is ".$cosine_value;
return round($cosine_value,3);
}
?>
<?php
function normalise($input)
{

$sum=0;
 for($i=0;$i<count($input);$i++)
{
$sum+=$input[$i]*$input[$i];
}
//now we have to normalise each element of the array
```

```
for($j=0;$j<count($input);$j++)
{

$input[$j]=round(($input[$j]/sqrt($sum)),2);

}
return $input;
}
?>
```

5. **Source code of main function and gauss set**

```
<?php

//include "fuzzyset.php";
//include "tokenize.php";
function countFiles($dir){
   $files = array();
   $directory = opendir($dir);
   while($item = readdir($directory)){
   // We filter the elements that we don't want to appear ".", ".." and ".svn"
       if(($item != ".") && ($item != "..") && ($item != ".svn") ){
           $files[] = $item;
       }
   }
   $numFiles = count($files);
   return $numFiles;
}
function calculate_tf($links,$df_token)
{


         //tokenize the file and compare every term with it
         //we obtain the individual file from the links and compare it with the term obtained
from the document frequency
         $ar=tokenize($links);
```

```
        //echo count($ar);echo "<br>";
        $counter=0;$i=0;
        while($i<sizeof($ar))
        {
                if($ar[$i]==$df_token){ $counter++;}
                $i++;
        }
        return $counter;
}


?>


<?php

//include "fuzzyset.php";
//include "tokenize.php";

function calculate_gauss($term,$count,$fuzzyset)
{

        //$fuzzyset=fuzzyarray('c:/xampp/htdocs/Thesis/program/sentence/Fuzzy_Set/Sample
1',$count);
        //print_r($fuzzyset);
        $z=0;$zbar=1;
        foreach($fuzzyset as $elements)
        {
                $tokens=tokenize($elements);
                //foreach($tokens as $token){if($term==$token){echo $token;}}
                if(in_array($term,$tokens))
                        {
                                //echo $term;
                                $numerator=-1*pow(($z-$zbar),2);
                        $denominator1=2*pow($value,2);
                        $deno=10;
                        $val=$numerator/$deno;
                         $val2=exp($val);
                        return round($val2,3);

                }

        }

}

function return_group($term,$count,$fuzzyset)
{
        //calculate the return
        //return the fuzzy set array
        //$fuzzyset=fuzzyarray('c:/xampp/htdocs/Thesis/program/sentence/Fuzzy_Set/Sample
1',$count);
        //print_r($fuzzyset);
        for($i=0;$i<count($fuzzyset);$i++)
        {
                $tokens=tokenize($fuzzyset[$i]);
                foreach($tokens as $token)
                {
                        if($term==$token)return ($i+1);
                }
        }

}

?>
<?php
/**
$file=file_get_contents('2.txt',NULL,NULL,3);
```

```
echo calculate_gauss($file,5);echo '<br>';
echo return_group($file,5);
***/
?>
```

6. Source code of Doccounts and tokenize

```
<?php

function tokenize($sentence){
   $token_ar=array();
   $token=strtok($sentence," ");
   while($token)
   {
            $token_ar[]=$token;
            $token=strtok(" ");

  }
return $token_ar;
}//function close
?>
<?php
function docCounts($term,$doclists)
{

//in this function the strings are tokenized and and $term is checked
$a=array();
///echo "The term is".$term;
///echo count($doclists);
$counter=0;
for($i=0;$i<count($doclists);$i++)
{

            $a=tokenize($doclists[$i]);
```

```
        //looping over tokenized array and check term whether it exists there or not
        for($j=0;$j<count($a);$j++)
        {

                if($term==$a[$j]){$counter++;break;}

        }


}//for loop
//echo $counter;
return $counter;
}//function close
?>
<?php
/****
$links=array('म राम्रो छु', 'म नराम्रो होईन');


//pass $links into the array
print_r($links);
$val=docCounts('राम्रो',$links);
echo $val;

function tokenize($sentence){
  $token_ar=array();
  $token=strtok($sentence," ");
  while($token)
  {
        $token_ar[]=$token;
        $token=strtok(" ");

  }
return $token_ar;
```

```
}//function close
*********/
?>
```

7. **Fuzzy set used in this dissertation**

   (Sample 1)

First Set

ठुलो बिशाल भयंकर भयानक अजंग गगनचुम्बि अग्लो

Second Set

गुणयोग्य राम्रो सुन्दर असल सद्गगुण सद्बभाब प्रेम सप्रेम गुणवान गुनिलो मायालु प्रेममय

Third Set

नराम्रो रद्दि खराब अबगुण बेकम्मा बेसुर बेईमान अपमान

   (Sample 2)

First Set

प्रधानमन्त्रि बाबुराम बालुवाटार मुस्ताङगाडि ईन्जिनियर सहयोगी नेता राजनेता विद्वान डाक्टर कम्युनिष्ट सहृदयी गणतान्त्रिक राजनेता

Second Set

भ्रष्टचार भ्रष्ट भ्रष्टचारी अन्यायि ब्याभीचारी घृणित अपराधि बदनामी घुस्याहा अन्यायि

Third Set

तानाशाह तानाशाही निरङकुश छलकपटी

Fourth Set

युद्ध बन्दुक खुकुरी तोप गोलि बारुद मानु काट्नु हान्नु काट मार हान

Fifth Set

कांग्रेस बि.पि गणेशमान मात्रिका गिरिजा शुशिल रुख चारतारे

Sixth Set

राप्रपा हलो पशुपतिशम्शेर कमल

8. **Test Data used for this dissertation**

(Data 1)

Document#1:

ठुलो

Document#2:

बिशाल

Document#3:

सानो

Document#4:

सुन्दर

Document#5:

नराम्रो

Document#6:

रद्दि

Document#7:

गुणवान

Document#8:

गुणयोग्य

Document#9:

अबगुण

Document#10:

भयंकर

Document#11:

खराब

Document#12:

बेईमान

Document#13:

बेसुर

Document#14:

बेकम्मा

Document#15:

अजंग

Document#16:

अग्लो

Document#17:

भयानक

Document#18:

सप्रेम

Document#19:

प्रेम

Document#20:

मसिनो

(Data 2)

Document#1:

प्रधानमन्त्रि

Document#2:

बाबुराम

Document#3:

भ्रष्टचार

Document#4:

बालुवाटार

Document#5:

मुस्ताङगाडि

Document#6:

ईन्जिनियर

Document#7:

भ्रष्टचारी

Document#8:

युद्ध

Document#9:

तानाशाह

Document#10:

तानाशाही

Document#11:

नेता

Document#12:

घृणित

Document#13:

बारुद

Document#14:

अपराधि

Document#15:

काट्नु

Document#16:

छलकपटी

Document#17:

तोप

Document#18:

गोलि

Document#19:

मानु

Document#20:

खराब

(Data 3)

Document#1:

यो ठुलो छ

Document#2:

यो बिशाल छ

Document#3:

यो खराब छ

Document#4:

यो रद्दि छ

Document#5:

यो सुन्दर छ

Document#6:

यो नराम्रो छ

Document#7:

यो भयंकर छ

Document#8:

यो प्रेममय छ

Document#9:

यो गुणवान छ

Document#10:

यो अजंग छ

Document#11:

यो मायालु छ

Document#12:

यो बेसुर छ

Document#13:

यो बेकम्मा छ

Document#14:

यो असल छ

Document#15:

यो भयानक छ

Document#16:

यो खराब छ

Document#17:

यो नराम्रो छ

Document#18:

यो ठुलो छ

Document#19:

यो गुणवान छ

Document#20:

यो प्रेममय छ

(Data 4)

Document#1:

उहाँ बाबुराम हुनुहुन्छ

Document#2:

उहाँ राप्रपा हुनुहुन्छ

Document#3:

उहाँ सहयोगी हुनुहुन्छ

Document#4:

उहाँ राजनेता हुनुहुन्छ

Document#5:

उहाँ ईन्जिनियर हुनुहुन्छ

Document#6:

उहाँ डाक्टर हुनुहुन्छ

Document#7:

उहाँ सहृदयी हुनुहुन्छ

Document#8:

उहाँ कांग्रेस हुनुहुन्छ

Document#9:

उहाँ हलो हुनुहुन्छ

Document#10:

उहाँ प्रधानमन्त्रि हुनुहुन्छ

Document#11:

उहाँ गणेशमान हुनुहुन्छ

Document#12:

उहाँ मात्रिका हुनुहुन्छ

Document#13:

उहाँ गिरिजा हुनुहुन्छ

Document#14:

उहाँ गणतान्त्रिक हुनुहुन्छ

Document#15:

उहाँ शुशिल हुनुहुन्छ

Document#16:

उहाँ बि.पि हुनुहुन्छ

Document#17:

उहाँ विद्वान हुनुहुन्छ

Document#18:

उहाँ राप्रपा हुनुहुन्छ

Document#19:

उहाँ कमल हुनुहुन्छ

Document#20:

उहाँ चारतारे हुनुहुन्छ