Concept Extraction from Ambiguous Text Document using K-Means

Md. Mehedi Rahman Rana[1], Rubya Afrin[2], Md. Anisur Rahman[3], Azazul Haque[4], Md. Ashiqur Rahman[5]

[1,2,4,5]Lecturer, Department Of CSE, Northern University of Business & Technology Khulna, Bangladesh
[3]Professor, CSE Discipline, Khulna University, Bangladesh

---***---

Abstract - *Increasing development in web technologies have led to the massive generation of textual data analyzing such quantities of text content and discovering the pattern from the text is a complex and challenging task. A recent approach for the unsupervised analysis of text corpus is grouping the words based on their semantic relationship. Word clustering is such a process which analyzes and extracts significant relations from text documents. Clustering process using K-means is one of the most common and efficient techniques. This paper presents a model to identify the semantic relationship among the words of a document using word clustering. The system is designed to extract the overall concept of a document written on different content. Ambiguity from a document (any article) can be removed by clustering the words of similar semantic values as well.*

***Key Words*: Clustering, Document, K-means algorithm, Natural Language Processing Tool Kit (NLTK), Principal Component Algorithm (PCA), Text Mining.**

1. INTRODUCTION

The modern world is becoming more and more digital with the development of new smart technologies. Web search is very common practice among the people all over the world. A huge amount of information is published on the internet in every second. So, the web is enriched with massive number of textual documents every day. With this information overloaded, looking up for the precise and relevant information resource and extracting the key concept from the resource has become challenging within a very short time. Text mining is one of those techniques which help extract such kind of useful information in an automatic way. Text mining [1] is the automatic technique of discovering novel, previously unknown and non-trivial information from unstructured textual document. All the extracted information is linked together to form new facts or new hypotheses to be discovered further by more conventional means of experimentation.

This research paper entitled **"Concept extraction of ambiguous text document using the K-means algorithm,"** is mainly focusing on using of text mining techniques and the K means algorithm to create the clusters of similar words in a document. The project mainly focuses on the document available in natural languages and extraction of its contents. The document written in a textual format is fetched for clustering the words. Then the document is pre-processed using the common pre-processing techniques and finally grouped into clusters based on their similarities. These clusters are displayed on a page to demonstrate an overall concept of a large ambiguous document.

1.1 Problem Statement

A person who is reading a particular document cannot get the access to the title. Besides, the title may be confusing in understanding the actual content of the document. The problem here is to understand the certain concept without reading the whole document. To read the document line by line and to get an overall idea is very time-consuming and tedious task. The solution implemented here is based on the technique of text mining and clustering which helps the user to get an idea of any ambiguous document without giving much effort and time.

1.2 Objectives of the Research

The primary objective of this paper is to implement the clustering process of words of a text document that can assist to realize the main idea of the document. In addition, this system can help the user differentiate among ambiguous documents. The major purpose of developing the automatic scheme is-

- To discover an overall concept about a document without reading the whole content.
- To identify the key terms of any document.
- To use the key words for further text mining techniques like summarization and classification.
- To distinguish ambiguous documents having similar words but different content.

2. LITERATURE REVIEW AND BACKGROUND:

2.1 Introduction

Previous works on text mining and document clustering mainly focused on different levels like: text clustering, document clustering, information extraction, information retrieval, and topic extraction etc.

J. Sathya and S. Priyadharshini (2012) [8] proposed a system which is designed to identify the semantic relations using the ontology. The ontology is used to represent the term and concept relationship. The synonym, metonym and hyponym relationships are represented in the ontology. Andrew L. Mass and Andrew Y. Ng (2010) introduced a model which exhibits semantic focuses on word vectors using a probabilistic model of documents. The paper evaluated the model's word vector in two tasks of sentiment analysis.

Chihli Hung and Stefan Wermter [25] proposed three novel text vector representation approaches for Neural network based document clustering using word net ontologies.

Ayush Agrawal and Utsav Gupta [27] employed an extraction based technique to generate automatic summaries from a document. The paper describes an algorithm that incorporates k-means clustering, term-frequency inverse-document frequency and tokenization to perform extraction based text summarization.

Vishwanath et.al [24] proposed a model to categorize the documents using KNN based machine learning approach and then returned to the most relevant documents.

Jaiswal (2007) investigated the comparison among different clustering methods like K-Means, Vector Space Model (VSM), Latent Semantic Indexing (LSI), and Fuzzy C-Means (FCM) and selected FCM for web clustering.

Sumit Goswami and Mayank Singh [23] proposed a system to apply fuzzy logic in text mining in order to perform document clustering and at the same time, they gave an example of document clustering where the document had to be clustered into two categories.

Maheshwari and Agrawal (2010) proposed centroid-based text clustering for preprocessed data, which is a supervised approach to classify a text into a set of predefined classes with relatively low computation and better accuracy.

2.2 Text Mining

Text mining, also known as knowledge discovery process is the automatic process of extracting significant and meaningful patterns or new knowledge from unstructured textual documents. The extracted information is then represented to new facts and makes it a new source of knowledge to be retrieved further.

The resources stored in web pages are retrieved by millions of people every day. Even if the user has access to the material, sometimes they do not have the access to see the title or the whole document. As a result, to get an overall idea about the document has become a boring and time-consuming task. Besides, the information being searched on web is already known and published by others. The search can provide irrelevant data that are not relevant to the actual information needed. In the process of text mining, text must be accessed, copied, analyzed, annotated and related to existing information and understanding. Thus, it makes the task of acquiring concept extraction easier and efficient.

The text mining techniques [1] begin with the collection of text documents, then a text mining tool is applied which retrieves the document and pre-processes it. The technique cleans the document by checking each character and finds the base word within the text. In the next step, it would go through a phase which analyzes the text, sometimes repeats the techniques until the certain information is extracted. There are several types of techniques for organizing and arranging the documents that depends on the goals of the result. Here three text analyzing techniques are shown in the example that we have implemented in our research. The resulting information can be placed in a management information system, providing meaningful knowledge for the user of that large text document.

Figure 2.1, illustrates a generic process [1] of a text mining application.

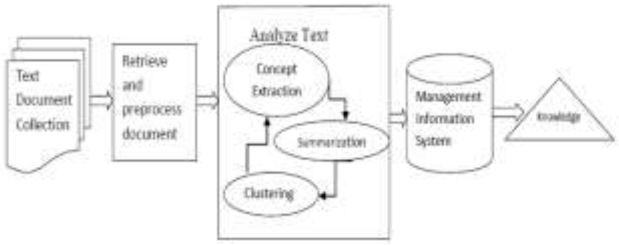


Fig - 2.1: An example of Text Mining

Text mining refers to data mining techniques for searching useful patterns from the text. The information extracted by this process is not organized in a structural format rather than it concerns with the text written in natural language. Information and relations are hidden into language structure and are not explicit as in data mining.

2.3 Document Pre-processing

Document pre-processing is the task of converting a raw text document where each document is represented by a set of index terms. Text mining includes clustering, classification, topic mining, and information extraction etc. All these processes are required to complete the preprocessing step before doing their intended tasks. Pre-processing significantly reduces the size of the input text documents and the processes the document is written in natural language using stop-word elimination, tokenization and stemming.

Pre-processing step includes identifying the terms which are considered as linguistically-meaningful units for the system. The goal of document pre-processing is to represent the documents in such a way that their storage in the system and retrieval from the system are very efficient. However, document pre-processing includes the following stages:

2.3.1 Tokenization

Tokenization is the process of slicing up a stream of text into words, phrases, symbols, or other meaningful elements called tokens which are grouped together as a semantic unit and used as input for further processing such as parsing or text mining. The aim of the tokenization is to identify the words in a sentence. The list of tokens becomes input for further processing such as parsing or text mining. Textual data is only a block of characters at the beginning. All processes in information retrieval need the words of the data set. Hence, tokenization is an important step to complete.

Tokenization is a useful process in the fields of both Natural language processing and data security. It separates the word of a sentence. The process removes some special characters like white space, delimiter, semi-colon etc. from each sentence.

2.3.2 Stop Word Removal

Some words are very common to a document, which would appear the most but have less significance in extraction of the concept, and is completely omitted from the document. These words are called “stop words” and the technique of omitting these words is called “stop word removal”. It is obvious that stop words do not contribute to the context or content extraction of textual documents. Due to their high frequency of occurrence, their presence in text mining presents an obstacle in understanding the content of the documents.

Some of the examples of stop-word are: a, an, the, and, or, but, for, from, has, he, she, in, is, are, as, at, be, it, its, of, on, that, the, to, was, were, will, who, which, with, when, how etc.

The common scheme for determining a “stop word list” is to sort the terms by collecting frequency and then to enlist the most frequently used terms, in a stop list, the members of which are excluded from the document during indexing.

2.3.3 Stemming

Another important way to reduce the number of words in the representation is to use stemming. This is based on the observation that words in documents often have many morphological variants. For example, we may use the words computing, computer, computation, computational, computes, computable, computability etc. in the same document. These words clearly have the same linguistic root. Putting them together as if the system is processed with single word, generally gives a strong indication of the content of the document whereas each similar word individually may not. Thus the use of stemming can be a very effective way to manage the number of words in a document.

2.4 Document Representation

Document representation is the most significant part of text mining and concept retrieval systems. It is very important to transform the textual format of the documents into vector form to reduce the complexity of the documents and make it easier to handle. Such a transformed document describes the contents of the original documents based on the principal terms called base words. Each term is assigned to a vector value for the representation. Then these terms are used in processing and applying text mining technique to achieve useful information and concept retrieval.

There is an automatic tool called word2vec [5, 6] which takes a text document as input and produces the vector form of that word as output. It first constructs a vocabulary from the training textual data and then learns vector value of each word. Thus, it represents the whole document as a separate vector value for separate words. The resulting word vector representation captures relationship in word’s function and can be used as features in many natural language processing applications.

All of the training words (vector values) in any document are stored in an n-dimensional pattern space. When an unknown word is given, a k-means algorithm searches the pattern space for the k training words that are closest to the unknown word. "Closeness" is defined in terms of Euclidean distance, where the Euclidean distance between two points are, X=(x1, x2,……,xn) and Y=(y1,y2,….,yn) is [20] :

$$d(X,Y) = \sum_{i=1}^{n} \sqrt{(x_i - y_i)^2} \qquad \text{..........(1)}$$

Where X and Y are the two compared vector values of words and n is their number of words in the document.

For the problem of clustering text documents, there are different criterion functions available [2]. The most commonly used one is the cosine function. The cosine function measures the similarity between two documents as the correlation between the document vectors representing them. For two documents or words di and dj, the similarity between them can be calculated as [8]

Cosine (di, dj) = di * dj / || d i || |d j|| …….. (2)

Where X represents the vector dot product and |di| denotes the length of vector di. The cosine value is 1 when two documents are identical and 0 if there is nothing common between them. The larger cosine value indicates that these two documents share more terms and are more similar.

2.5 Dimensionality Reduction

Dimensionality reduction process transforms the data in the high dimensional space to a space of fewer dimensions. The data transformation may be linear, as in principal component analysis (PCA), but many nonlinear dimensionality reduction techniques also exist [7]. For multidimensional data, tensor representation can be used in dimensionality reduction through multi linear subspace learning.

Principal component analysis (PCA) [26] is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. The resulting vectors are an uncorrelated orthogonal basis set. The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is symmetric. PCA is sensitive to the relative scaling of the original variables.

2.6 Clustering words

Clustering [7] is the grouping of a particular set of objects based on their characteristics, aggregating them according to their similarities. Regarding data mining, this methodology partitions the data implementing a specific join algorithm, most suitable for the desired information analysis. For document clustering, it is the process of grouping of contents such as words, word phrases or documents based on their content to extract knowledge. Word clustering is the similar process of clustering that groups the words according to its semantic values. The word belonging to one cluster is similar and the word belonging to another cluster is dissimilar. Grouping of words into individual cluster can be described by its own concept.

2.6.1 K-means Clustering

K-means is one of the most popular and efficient techniques for clustering. This clustering algorithm was developed by McQueen, and is one of the simplest and the best known unsupervised learning algorithms that solve the well-known clustering problem. The K-Means algorithm aims to partition a set of objects, based on their content/features, into k clusters, where k is a predefined or user-defined constant.

From the given set of n data with k cluster represents the grouping of words of a document. The value of k is set at the beginning of the algorithm which represents the number of cluster to be obtained. For grouping 'n' objects into 'k' cluster, k centric are initialized randomly. Each time a data is appeared, it is immediately assigned to its closest cluster centric and the centric of that cluster is updated. The process is continued until there is no change of the value of the centric in each cluster. The centric of a cluster is formed in such a way that it is closely related (in terms of similarity function; similarity can be measured by using different methods such as cosine similarity, Euclidean distance, Extended Jaccard) to all objects in that cluster. Finally, the elements belonging to one cluster are close to each other and dissimilar to the elements belonging to the other cluster.

The detailed algorithm of k-means is presented in Section 4.2.3.

2.7 Motivation

The recent growth of technology generates a massive amount of information every second in unstructured or semi-structured format which can be retrieved and converted into significant knowledge. The resources stored in web pages are accessed by millions of people every day. Moreover, content on different topics is published on same document and makes it an ambiguous textual document. As a result, to get an overall idea about the document has become a boring and time-consuming task. Different automatic techniques are available for this purpose. Text mining is one of them that provide the relevant information on the basis of user requirement. Topic extraction is one of the beneficial approaches to know the title or the topic of a text. But, sometimes, ambiguity of words makes it difficult to understand on which topic the document is written. The availability of the similar words and their corresponding semantic relationship under a single roof would be a very effective and sophisticated option to explore the concept extraction.

This paper provides a better, easier, and a clearer understanding of the overall process involved in grouping words into clusters based on the similarity of their ontology and displaying them under a single platform for better understanding of the concept of an ambiguous document.

2.8 Limitations of existing system

Text mining field is a massive and complex area to deal with. How to explore and utilize the huge amount of text documents is a major question in the areas of information retrieval and text mining. Document clustering is one of the most important text mining methods that are developed to help users effectively navigate, summarize, and organize text documents [5]. By organizing a large amount of words into a number of meaningful clusters, word clustering can be used to identify the content of documents or organize the results returned by a search engine in response to a user's query. Most of the existing systems that tried to group the words or documents, have represented documents using vector space model or ontology based term analysis, applied fuzzy c-means algorithm for clustering the words of a document etc.

The limitations of existing system are –

- Vector space model treats a document as a bag of words which creates high dimensionality in the feature space. When dimensionality increases, the number of words also increases. So, applying algorithm on huge number of words becomes difficult.
- Most of the system do not consider the semantic relationship among the words. Similar words can make confusion and increases dimensionality. As a result, handling these data becomes challenging.

- VSM has more complexity with feature selection techniques.
- The existing system could not work efficiently in high-dimensional feature spaces due to the inherent sparseness of the data which imposes a big challenge to the performance of clustering algorithms.

Moreover VSM, Fuzzy C-Means (FCM) cannot deal with ambiguous documents that contain different contents with similar words. The algorithms never considers the context or semantic relationship of the words of the documents that may cause confusion in understanding the idea of the document.

2.9 Conclusion

In this section, we have discussed about text mining, document pre-processing, document representation, clustering process of words and its overall importance in our modern life. All topics have a common goal which is to discover a new idea from the existing resources. So we are going to develop a system that will extract new and interesting knowledge from ambiguous documents and help the users get an overall idea of that document.

3. METHODOLOGY

3.1 Introduction

In this section, the working principle and architecture of our implemented method will be discussed. We have implemented unsupervised clustering text based method for understanding the ambiguity of the documents. This section will give the methodology of concept extraction of a document using text mining algorithm. In section 3.2 and 3.3, architecture and the methodology of our implemented system has been illustrated respectively.

3.2 The Overall Research Process

In this section, the overall process implemented for the thesis is described. The process includes of step-by-step actions performed for the clustering of words from different articles or documents which have the similar words but dissimilar contents. The overall research process is illustrated in Figure 3.1. In the figure, the process starts with the collection of document. Then the document is processed with some pre-processing techniques. After having it pre-processed, the clustering process is applied. The detail research process is illustrated below:

3.2.1 Document Pre-processing

Document preprocessing is the most crucial part of text mining. It is an important part of any NLP system, since the words explored at this stage are the fundamental units and passed to all further processing stages. The processing method has been stated in Section 2.2. Because the sentences of a document always have some special character, specific parts-of-speech like prepositions, articles, and pronouns. These type of words have to be eliminated from the document before applying any algorithm. The document pre-processing includes the following steps:

3.2.1.1 Tokenization

The detail concept and explanation about Tokenization have been stated in Section 2.2.1.The text that we have used in this thesis is in English. Generally textual data is only a set of words in the data set. The document is considered as stream of strings and the strings are considered as stream of characters. They may contain characters like brackets, hyphens, punctuation marks, spaces etc. The tokenization process is carried by using these type of special characters to split the sequence of characters to meaningful units called token.

3.2.1.2 Stop Word Removal

A document consists of some words that appears frequently throughout the text but these words have less importance in concept extraction. The words are discarded entirely to reduce the vocabulary. These common words are stop words and the process is called stop word removal. Stop words accounts 20-30% of total word counts in a particular text documents. There is a list of stop words such as article, prepositions, who, which, how etc. The tokens fetched from the tokenization step are compared to the stop word list. If the token matches to the list, it is discarded from the document.

3.2.1.3 Stemming

The stemming process have been mentioned in detail in Section 2.2.2. As this thesis is carried out using English language, there appeared so much vagueness during implementation. So, to reduce this complexity, the stemming process is applied. It identifies words to their root form. The documents contain words having the same linguistic root, are replaced by the root. Stemming may reduce indexing size as much as 40- 50%. Thus, the process also reduces the dimensionality of the document. Finally, we can get a reduced number of features or words for further processing.

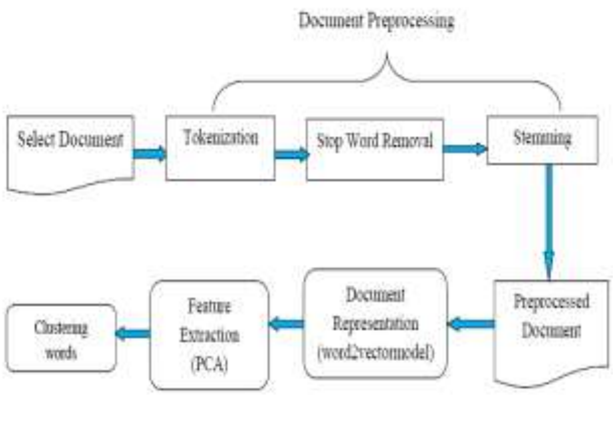


Fig - 3.1: The overall research process

3.2.2 Document Representation

Word2vec [5,6] is the one of the powerful and efficient tools for representing documents as vectors using efficient implementation of continuous-bag-of-words and skip-gram architecture as mentioned in Section 2.3. Each word is assigned to its vector values. Then, the entire document is represented as vectors in which similar words have the vector values close to each other.

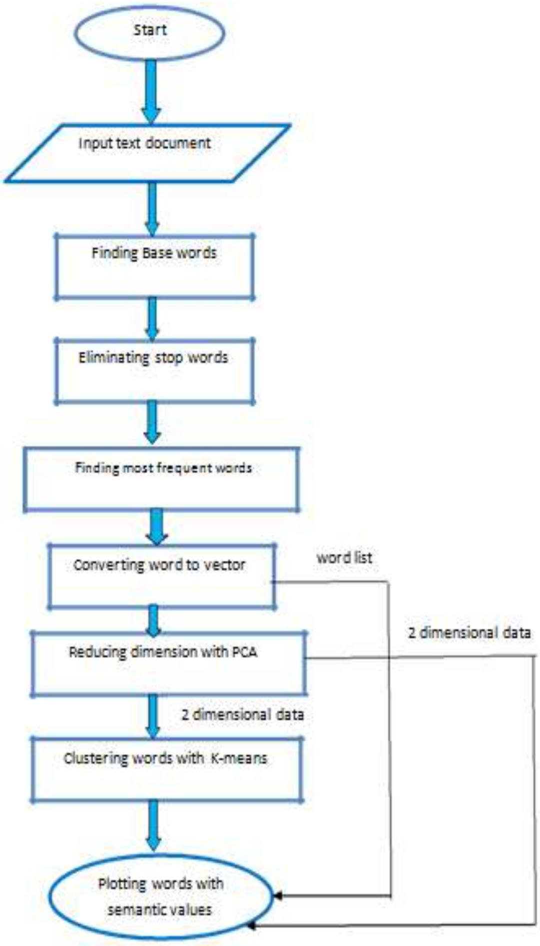

Start
Input text document
Finding Base words
Eliminating stop words
Finding most frequent words
Converting word to vector
word list
Reducing dimension with PCA
2 dimensional data
2 dimensional data
Clustering words with K-means
Plotting words with semantic values

Fig - 3.2: Flowchart of the proposed method

3.2.3 Dimensionality Reduction using PCA Algorithm

The detail concept and explanation about feature extraction have already been mentioned in Section 2.4. After each base word is assigned by its vector values, the process of dimension reduction is carried out. Synonym expansion is carried out by searching each token in the dictionary and transforming each word to the base words. The dictionary consists of a list of words and all of their synonyms.

3.2.4 Clustering Words using K-means Algorithm

After completing feature extraction, clustering process is applied on the words of the document. The K-means clustering algorithm is used to meet the purpose of this research.

The basic algorithm of K-means used for the project is as following:

3.2.4.1 K-means Algorithm

For partitioning where each cluster's center is represented by the mean value of the objects in the cluster.

```
Input:
k: the number of clusters
n: the number of words
Output:
A set of k clusters.
Method:
Step 1: Choose k numbers of clusters to be determined.
Step 2: Initialize $C_k$ center randomly for k clusters.
Step 3: Repeat
        3.1: Assign each element to its nearest cluster center using Euclidean distance.
        3.2: Calculate new cluster centric by computing mean points.
Step 4: Until
        4.1: Cluster centroid is not changed OR
        4.2: No element changes its cluster
```

3.2.5 Mapping/Visualization of the results

The output achieved from the entire process, which, in fact are the clusters of similar words from ambiguous documents, is represented in the console on the two-dimensional space. The result shows clusters of words which have the similar content. Moreover, it also shows the largest cluster among all the clusters to give an idea/abstract of the ambiguous document.

3.2.6 Challenges of this research

- One of the challenges of the system is generating the vector values to identify the semantic relationship among the words of the document.
- Another challenge of the implementation is clustering the words which give us insight of the topic or the general idea of the document.

3.3 Conclusion

In this section we discussed our proposed system and how it will work. We also briefly discussed the challenges we are going to face in building the system.

4. IMPLEMENTATION AND RESULT ANALYSIS

4.1 Introduction

In this section, we have discussed the total implementation procedure of our system. The implementation involves the extraction of large document in a textual format, fetching them in the system to go through the document pre-processing techniques, forwarding the pre-processed documents to the clustering system and obtaining clusters of similar words with similar semantic values as a final output.

4.2 Implementation model

The Implementation model is involved during the implementation phases are described in detail in Section 3.2. The model is divided into three phases. They are –

- Document preprocessing
- Document representation
- Clustering words

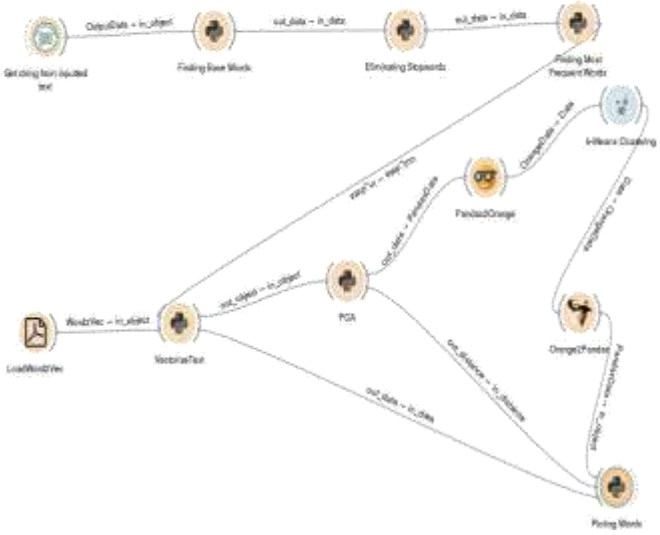
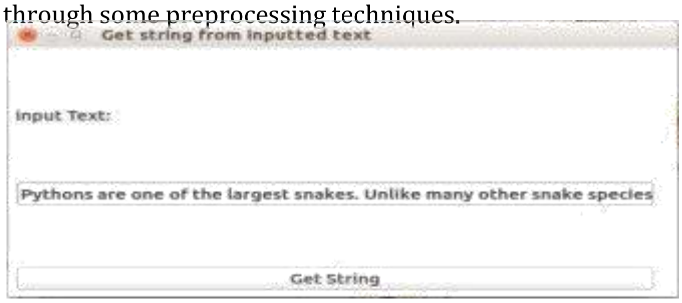


Fig - 4.1: Implementation model

The model is implemented for finding those words of a document that are helpful for further document classification, summarization and other text mining techniques. It finds the key terms of a document which can give insight of the topic of it help to get an overall idea. The above figure represent the whole process of clustering semantic valued words of a text document.

4.2.1 Input Text

Our thesis implementation begins with the collection of text documents. The text to be mined is fetched in this step. Document which is used in our system is written in English. Users can give any large document, text, paragraph, article etc. written in a textual format. There is no limitation in the size of the text. It can be as large as the user wants. Traditional input in orange wizard is a file. But we have created a python wizard to get input in a text format from the user. This text is further processed by the analyzing phase through some preprocessing techniques.



Pythons are one of the largest snakes. Unlike many other snake species

Fig - 4.2: Getting text input from user

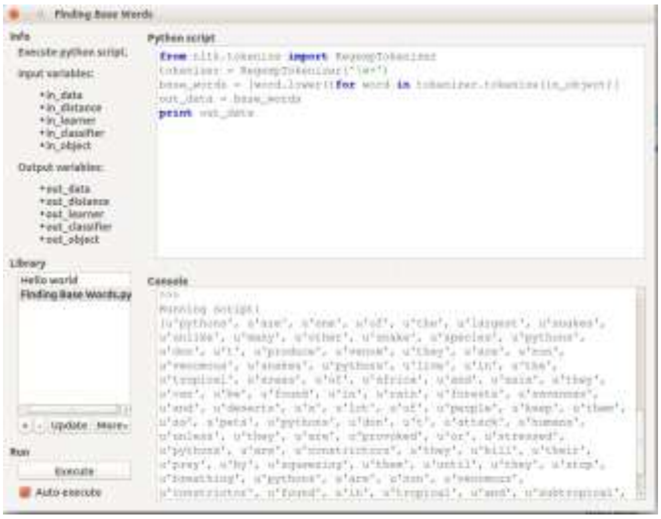
4.2.2 Document Preprocessing

After getting input in a textual format, the document is preprocessed with document preprocessing technique. The detail concept is described in section 2.3. This step represents the text document with a set of index terms. Pre-processing significantly reduces the size of the input text documents by tokenization, stop word removal and stemming. All the steps of Document pre-processing are described in the following section:

4.2.2.1 Tokenization

The input text is a collection of stream of sentence. But we have to deal with the base words only. This step slices up the stream of text into words, phrases and meaningful elements. It removes the special character and other elements from the sentence described in section 2.3.1. All the words are converted into Unicode.

To begin with tokenization, we have used a tool called Natural Language tool kit (NLTK). We have imported the tokenization package which is included in NLTK tool. The only function of this tool is to collect the words from stream of sentences.



```
from nltk.tokenize import RegexpTokenizer
tokenizer = RegexpTokenizer(r'\w+')
base_words = [word.lower() for word in tokenizer.tokenize(in_object)]
out_data = base_words
print out_data
```

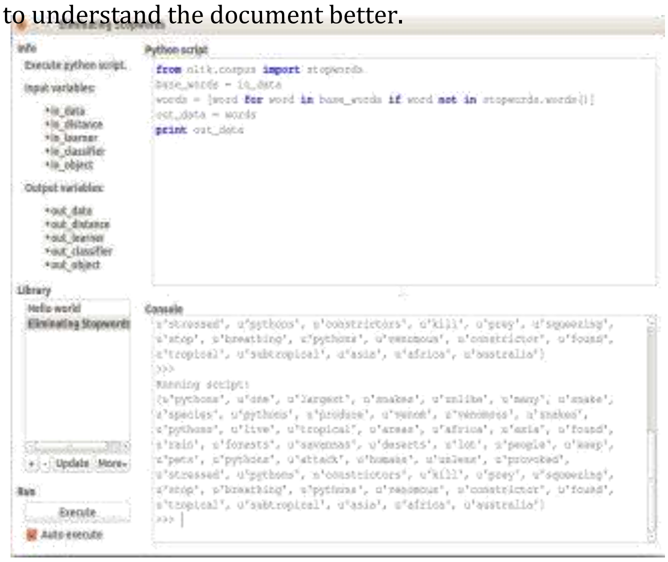
```
>>>
Running script:
[u'pythons', u'are', u'one', u'of', u'the', u'largest', u'snakes',
u'unlike', u'many', u'other', u'snake', u'species', u'pythons',
u'don', u't', u'produce', u'venom', u'they', u'are', u'non',
u'venomous', u'snakes', u'pythons', u'live', u'in', u'the',
u'tropical', u'areas', u'of', u'africa', u'and', u'asia', u'they',
u'can', u'be', u'found', u'in', u'rain', u'forests', u'savannas',
u'and', u'deserts', u'a', u'lot', u'of', u'people', u'keep', u'them',
u'as', u'pets', u'pythons', u'don', u't', u'attack', u'humans',
u'unless', u'they', u'are', u'provoked', u'or', u'stressed',
u'pythons', u'are', u'constrictors', u'they', u'kill', u'their',
u'prey', u'by', u'squeezing', u'them', u'until', u'they', u'stop',
u'breathing', u'pythons', u'are', u'non', u'venomous',
u'constrictors', u'found', u'in', u'tropical', u'and', u'subtropical',
```

Fig – 4.3: Getting words from input text

Usually the word begins with a capital letter at the beginning of a sentence. So, there may present some words hold capital letter in the entire document. But all of them are turned into lower case by word. Lower () method is used after finishing the tokenization process.

4.2.2.2 Stop word removal

Some words appear frequently in a document but have little value in analyzing the text document. These words are called stop words. So, it is obvious to eliminate those words to understand the document better.



```
from nltk.corpus import stopwords
base_words = in_data
words = [word for word in base_words if word not in stopwords.words()]
out_data = words
print out_data
```

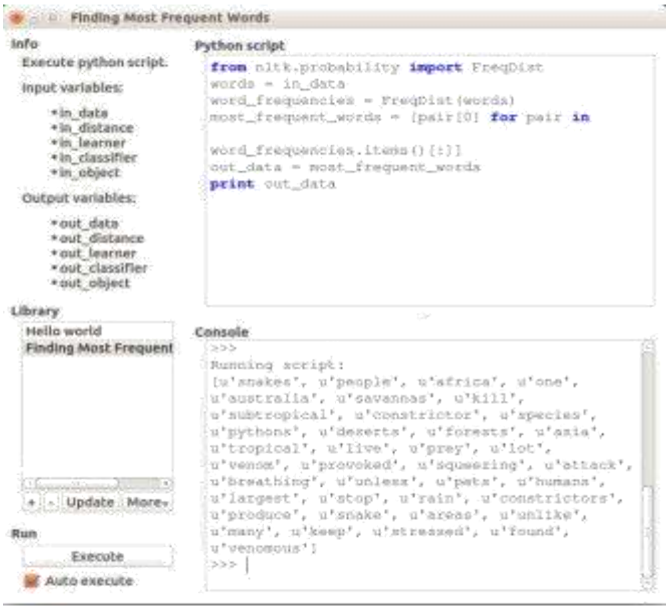
```
u'stressed', u'pythons', u'constrictors', u'kill', u'prey', u'squeezing',
u'stop', u'breathing', u'pythons', u'venomous', u'constrictor', u'found',
u'tropical', u'subtropical', u'asia', u'africa', u'australia']
>>>
Running script:
[u'pythons', u'one', u'largest', u'snakes', u'unlike', u'many', u'snake',
u'species', u'pythons', u'produce', u'venom', u'venomous', u'snakes',
u'pythons', u'live', u'tropical', u'areas', u'africa', u'asia', u'found',
u'rain', u'forests', u'savannas', u'deserts', u'lot', u'people', u'keep',
u'pets', u'pythons', u'attack', u'humans', u'unless', u'provoked',
u'stressed', u'pythons', u'constrictors', u'kill', u'prey', u'squeezing',
u'stop', u'breathing', u'pythons', u'venomous', u'constrictor', u'found',
u'tropical', u'subtropical', u'asia', u'africa', u'australia']
>>> |
```

Fig- 4.4: Eliminating Stop words

To remove stop words, Natural Language Processing tool kit (NLTK) is used again. We have imported the stop word package which is inherited from NLTK tool. The only function of this tool is to collect the words from stream of sentences. The package contains a list of pre-defined stop words and those are removed from the document during this process.

4.2.2.3 Finding Most of the Frequent Words

We have not counted the words that has used only once as they have no role in the concept extraction. We paid heed to the words that have appeared repeatedly. In this step, the method freqDist is utilized to locate the most frequent words in the document. The method finds the word that has appeared one or more times in the document. The word that appears once has less importance in further process. We have only chosen the words that can make difference in distinguishing an ambiguous document.



```
from nltk.probability import FreqDist
words = in_data
word_frequancies = FreqDist(words)
most_frequent_words = [pair[0] for pair in

word_frequancies.items()[:]]
out_data = most_frequent_words
print out_data
```

```
>>>
Running script:
[u'snakes', u'people', u'africa', u'one',
u'australia', u'savannas', u'kill',
u'subtropical', u'constrictor', u'species',
u'pythons', u'deserts', u'forests', u'asia',
u'tropical', u'live', u'prey', u'lot',
u'venom', u'provoked', u'squeezing', u'attack',
u'breathing', u'unless', u'pets', u'humans',
u'largest', u'stop', u'rain', u'constrictors',
u'produce', u'snake', u'areas', u'unlike',
u'many', u'keep', u'stressed', u'found',
u'venomous']
>>> |
```

Fig- 4.5: Finding Most Frequent Word

4.2.3 Document Representation

Document representation is the most crucial and challenging part of our implementation model. The remaining words of the document are to be represented by vector value. Each vector value of a word also represents the semantic relationship among them. Thus, it helps to extract knowledge from an ambiguous document very easily. The following steps are applied to represent the index term of a document into vector forms:

4.2.3.1 Load Word to Vector

This tool provides an efficient implementation for computing vector representations of words. Using the word2vec tool, it is possible to train models on huge data sets. We use google word 2 vector tool here.

After finding most frequent word, it is given to word2vec tool. To use the tool, first we have to load word2vec from PC directory. For this we have to create an orange widget. Here we use tkFileDialog which is a module with open and save dialog functions. The askopenfile function is used to create a file dialog object. Tkinter is Python's de-facto standard GUI (Graphical User Interface) package which is used to pop up a small window with two buttons (Open File & Quit). The tkMessageBox module is used to display message boxes. This module provides a number of functions that we can use to display an appropriate message. Gensim.models is a free python library which is used to load word to vector.

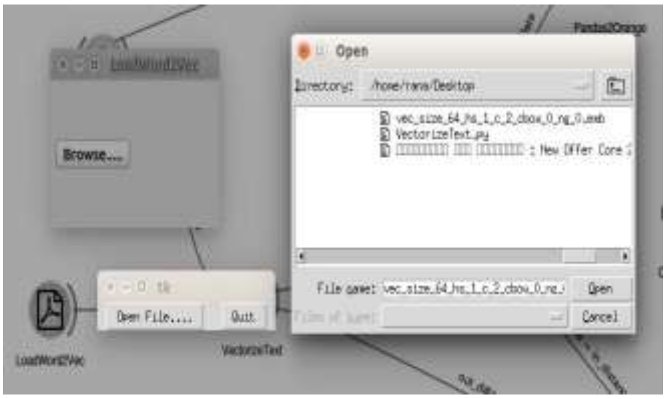
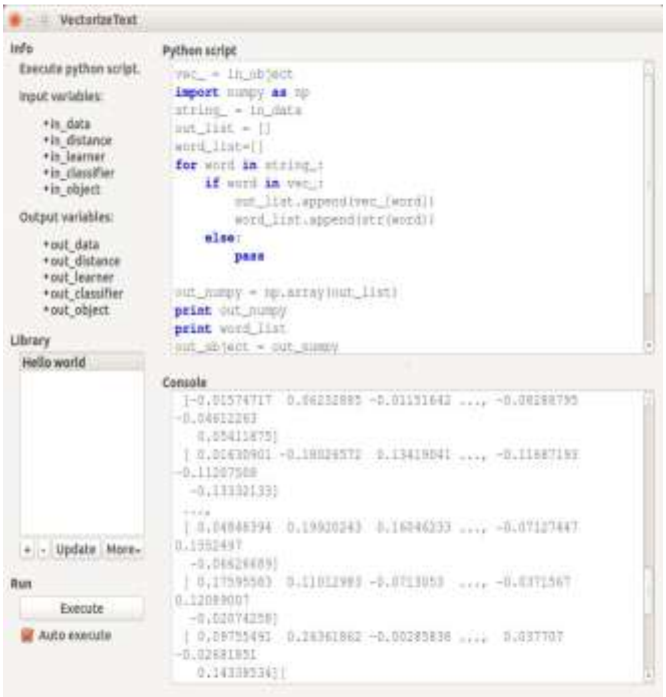


Fig- 4.6: Load Word to Vector Tool

4.2.3.2 Vectorize Text

After loading Word to Vector tool, the words are converted into vector forms. The tool is applied to get vector value for each word of the document. The vector values are fetched from the word2vec package where a vector is predefined for each word. After applying this tool we have got a vector of 64 dimensions for each individual word.

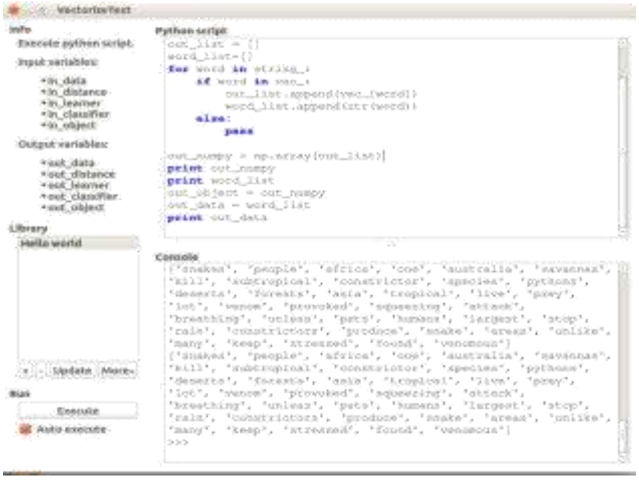
We shall receive two different outputs after the termination of the step. The first output is the list of vector values of each word and the other output is the list of those words those are converted into vector forms. The vector values for each word and the respective word list are shown in the following figures respectively:



```
vec_ = in_object
import numpy as np
string_ = in_data
out_list = []
word_list=[]
for word in string_:
    if word in vec_:
        out_list.append(vec_[word])
        word_list.append(str(word))
    else:
        pass

out_numpy = np.array(out_list)
print out_numpy
print word_list
out_object = out_numpy
```

```
[-0.01574717  0.06232885 -0.01131642 ..., -0.08288795
-0.04612263
  0.05411875]
 [ 0.01630901 -0.18024572  0.13419041 ..., -0.11687153
-0.11207508
  -0.13332133]
 ...,
 [ 0.04848394  0.19920243  0.16846233 ..., -0.07127447
0.1552497
  -0.06626489]
 [ 0.17595503  0.11012985 -0.0713053  ..., -0.0371567
0.12089007
  -0.02074258]
 [ 0.09755491  0.24361862 -0.00285838 ...,  0.037707
-0.02681851
  0.14339534]]
```

Fig- 4.7: Getting vector values for each word



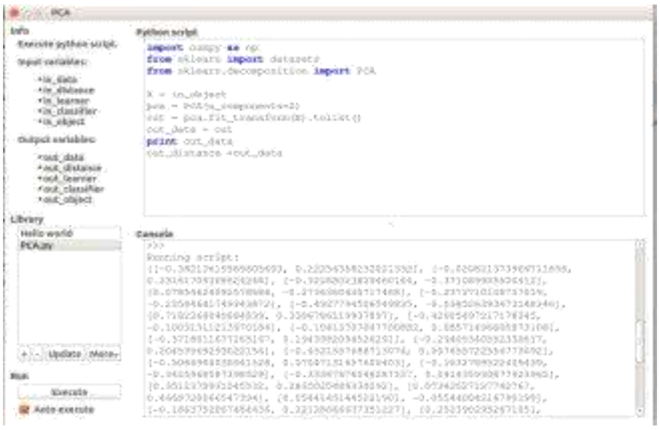
```
out_list = []
word_list=[]
for word in string_:
    if word in vec_:
        out_list.append(vec_[word])
        word_list.append(str(word))
    else:
        pass

out_numpy = np.array(out_list)
print out_numpy
print word_list
out_object = out_numpy
out_data = word_list
print out_data
```

```
['snakes', 'people', 'africa', 'one', 'australia', 'savannas',
'kill', 'subtropical', 'constrictor', 'species', 'pythons',
'deserts', 'forests', 'asia', 'tropical', 'live', 'prey',
'lot', 'venom', 'provoked', 'squeezing', 'attack',
'breathing', 'unless', 'pets', 'humans', 'largest', 'stop',
'rain', 'constrictors', 'produce', 'snake', 'areas', 'unlike',
'many', 'keep', 'stressed', 'found', 'venomous']
['snakes', 'people', 'africa', 'one', 'australia', 'savannas',
'kill', 'subtropical', 'constrictor', 'species', 'pythons',
'deserts', 'forests', 'asia', 'tropical', 'live', 'prey',
'lot', 'venom', 'provoked', 'squeezing', 'attack',
'breathing', 'unless', 'pets', 'humans', 'largest', 'stop',
'rain', 'constrictors', 'produce', 'snake', 'areas', 'unlike',
'many', 'keep', 'stressed', 'found', 'venomous']
>>>
```

Fig- 4.8: Getting word list

4.2.3.3 Apply PCA Algorithm

When we have the vector values of all words, it is processed by the Principal Component Algorithm (PCA). This algorithm transforms the data in the high dimensional space to a lower dimensional space. After vectorize the text, we have a vector value of 64 dimensions which is very complex to deal with. Moreover, if we want to plot the words, it is not visually possible. So, we have applied PCA algorithm to reduce this 64 dimensionality into two or three dimensional vector values.



```
import numpy as np
from sklearn import datasets
from sklearn.decomposition import PCA

X = in_object
pca = PCA(n_components=2)
out = pca.fit_transform(X).tolist()
out_data = out
print out_data
out_distance =out_data
```

```
>>>
Running script:
[[-0.38212415586805693, 0.22254358232821332], [-0.02682137396711656,
0.23141709319962042684], [-0.32580023829660164, -0.37108993559264412],
[0.07855424589251898, -0.27360804457171468], [-0.23737101087373034,
-0.22584681749943872], [-0.49277945265498095, -0.53450629347214834],
[0.71822688456604509, 0.33967061199378971], [-0.42685497217178345,
-0.10031311212397015], [-0.15813537847700882, 0.56571496988873108],
[-0.37188116771265147, 0.29439920345262911], [-0.29469540592338637,
0.20457969292502235], [-0.45218876886113074, 0.93745872235477744921],
[-0.30689685388041528, 0.37047131487488403], [-0.19327895224484439,
-0.04259468597139852], [-0.33867874546281327, 0.24143599068779239651],
[0.35112799813465232, 0.26650206885438592], [0.05242527157742767,
0.44697208464547394], [0.05441451445221901, -0.05544004216789299],
[-0.18627520674586436, 0.22120696817351227], [0.25239029352671851,
```

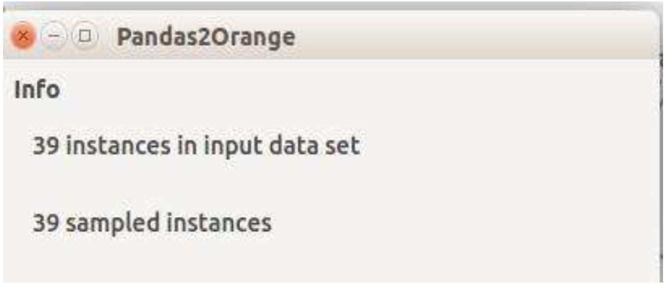
Fig- 4.9: Applying PCA algorithm

To decrease this high dimensionality, we have imported sykitlearn package and used PCA algorithm available under this package. As our goal is to plot the similar words in the console, we have defined to reduce the dimensionality into 2- dimensional vector values. We have used a method which generates a general value within the high dimensional vector values and reduces the curse of high dimension.

4.2.3.4 Convert Pandas Data to Orange Data

After applying PCA, we get two dimensional vector values. Then we pass the data set to K-Means for clustering. But we use here the Orange K-Means widget and it works only with Orange data. We get Pandas data after applying PCA, so we have to convert the Pandas data to Orange data.

For converting the data we create an orange widget, where we can see the number of pandas input data and orange output data. We use here a python file named conversion for changing pandas data to orange data.



Pandas2Orange

Info

39 instances in input data set

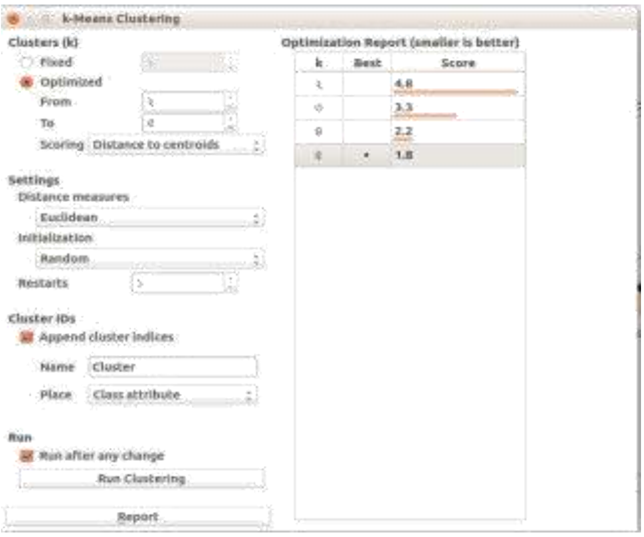
39 sampled instances

Fig- 4.10: Converting Pandas Data to Orange Data

In our example, we send 39 instances of pandas data and get 39 instances of orange data.

4.2.4 Apply K-Means Clustering Algorithm

After completing document representation, we have applied K-Means clustering algorithm to the data and this gives output of a new data set. Here cluster index is used as a class attribute. The original class attribute, if it existed, is moved to meta attributes. For various k, the scores of clustering results are also shown in the widget.



k-Means Clustering

Clusters (k)

○ Fixed

● Optimized

From 2

To 5

Scoring Distance to centroids

Settings

Distance measures

Euclidean

Initialization

Random

Restarts [illegible]

Cluster IDs

☒ Append cluster indices

Name Cluster

Place Class attribute

Run

☒ Run after any change

Run Clustering

Report

Optimization Report (smaller is better)

k	Best	Score
2		4.8
3		3.3
4		2.2
5	•	1.8

Fig- 4.11: K-Means Clustering Algorithm

1. Select the number of clusters.

- Fixed: algorithm clusters data in a specified number of clusters.
- Optimized: widget shows clustering scores for the selected cluster range.
- Silhouette (contrasts average distance to elements in the same cluster with the average distance to elements in other clusters)
- Inter-cluster distance (measures distances between clusters, normally between centroids)
- Distance to centroids (measures distances to the arithmetic means of clusters).

2. Select initialization method (the way the algorithm begins clustering):

- k-Means++ (first center is selected randomly, subsequent are chosen from the remaining points with probability proportioned to squared distance from the closest center)
- Random initialization (clusters are assigned randomly at first and then updated with further iterations)

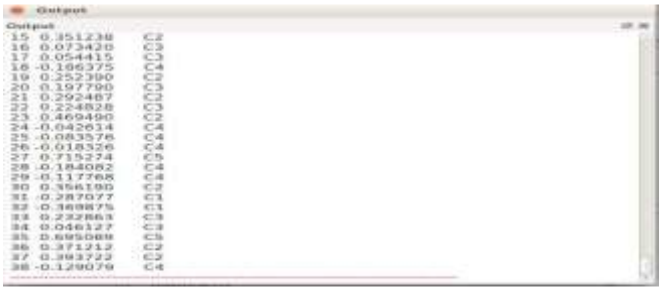
Re-runs (how many times the algorithm is run) and maximal iterations (the maximum number of iteration within each algorithm run) can be set manually.

3. The widget gives output of a new data set with appended cluster information. Select how to append cluster information (as class, feature or meta attribute) and name the column.

4. If Run on every change is ticked, the widget will commit changes automatically. Alternatively, click Run.

These are the descriptions of the User Interface section.

When we execute, then we get the final cluster as output which is given below.



Output

Output

```
15  0.351238    C2
16  0.073420    C3
17  0.054415    C3
18 -0.186375    C4
19  0.252390    C2
20  0.197790    C3
21  0.292487    C2
22  0.224828    C3
23  0.469490    C2
24 -0.042614    C4
25 -0.083576    C4
26 -0.018326    C4
27  0.715274    C5
28 -0.184082    C4
29 -0.117768    C4
30  0.356190    C2
31 -0.287077    C1
32 -0.369875    C1
33  0.232863    C3
34  0.046127    C3
35  0.695088    C5
36  0.371212    C2
37  0.383722    C2
38 -0.129079    C4
```

Fig- 4.12: Getting Cluster Value

4.2.4.1 Convert Orange Data to Pandas Data

After applying K-Means algorithm, we get Orange data. For plotting the word we need Pandas data. So we have to convert the Orange data to Pandas data again.

For converting the data we create an orange widget, where we can see the number of orange input data and pandas output data. We use here a python file named conversion for changing pandas data to orange data.

Orange2Pandas

Info

39 instances in input data set

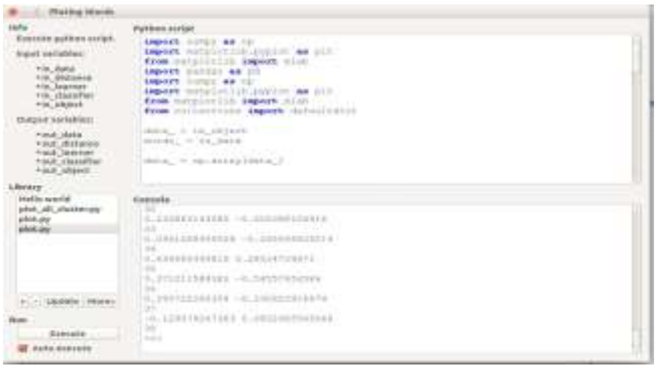
39 sampled instances

Fig- 4.13: Converting Orange Data to Pandas Data

In our example, we have sent 39 instances of orange data and get 39 instances of pandas data.

4.2.4.2 Plotting Words

The inputs of plotting words are the list of those words that are converted into vector forms, two-dimensional vector value of words and clustered words. We use here several package for plotting the words. Numpy is the fundamental package for scientific computing with Python. We use numpy as a powerful N-dimensional array object. *Matplotlib.pyplot* is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In *matplotlib.pyplot* various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes (please note that “axes” here and in most places in the documentation refers to the axes part of a figure and not the strict mathematical term for more than one axis). Collection is a high-performance container data type. This module implements specialized container data types providing alternatives to Python’s general purpose built-in containers, dict, list, set, and tuple.



Plotting Words

Info

Execute python script.

Input variables:

- in_data
- in_distance
- in_learner
- in_classifier
- in_object

Output variables:

- out_data
- out_distance
- out_learner
- out_classifier
- out_object

Library

Hello world
plot_all_clusters.py
plot.py
plot.py

Python script

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import mlab
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import mlab
from collections import defaultdict

data_ = in_object
words_ = in_data

data_ = np.array(data_)
```

Console

```
[illegible]
```

Fig- 4.14: Getting Plotting Value

We define here 30 Hexadecimal color for clustering. Here is the output of all clusters of words.

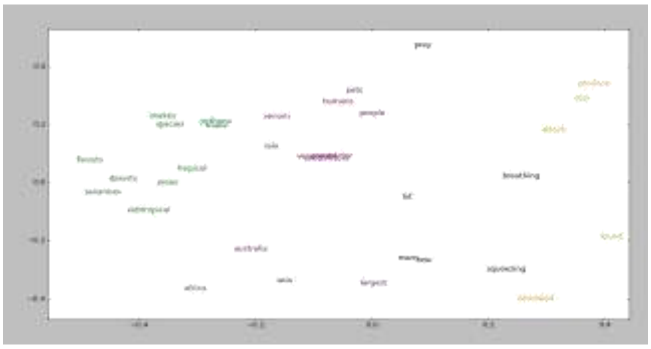


Fig- 4.15: Getting all Clusters of words

When we get all the cluster of words, it is difficult to understand what types of document it is. So finally, we plot the biggest cluster of words so that we can easily extract the concept of any document. The final output is given below.

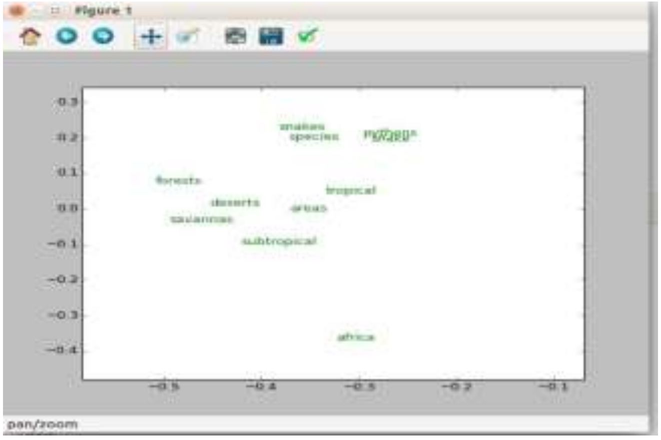


Fig- 4.16: Getting largest Cluster of words

From the above figures, we can say that the desired output of our system is acquired. Seeing these words, we can have an overview of the document at a glance. The system transforms a time-consuming task to an automatic one without reading the whole document.

4.3 Result Analysis

We have tested on some text document to make results. K-means is a heuristic method of constructing clusters of the documents. Execution time of the process is really fast but there is no guaranteed output. The output depends on optimal calculation of centroids of the clusters.

4.3.1 Dataset

To test and compare the approaches, we have collected a small corpus of Dutch Wikipedia articles consisting of 758 documents. After analyzing the documents, we have found 118099 terms occurred and among them 26373 were unique terms. The articles were taken from 8 Wikipedia categories: spaceflight, painting, architecture, trees, monocots, aviation, pop music and charadriiformes. All the articles are equally distributed over the categories. Different articles have different size among the categories.

4.3.2 Experimental Result and Evaluation

The methodology described in this paper is an extraction-based word clustering. If the largest cluster has smaller number of words in comparison with the original number of words, it will become difficult to gather a general idea of that document. On the other hand, if the number of words become larger, the concept extraction will also become harder. So, it is required for the resulting words in the largest cluster to be around 8% - 20% of the original words of the document. For evaluating our system, we ran simulation on 8 selected Wikipedia text samples. The result generated by our approach is not similar to the existing technique for concept extraction. For this, we select extraction-based summarization approach by autosummarization.com and compared it with our system. The existing approach shows the summarized sentence for concept extraction but we have generated a number of words for it. So, we have assumed that the existing system generating on an average of 10-12 words for each sentence. Then we have the words generated by our approach and compared it with summarizer extraction-based technique.

Table- 4.1: Results of Evaluation of our approach on various sample text

Text Number	Data Set	Number of words in original passage	Number of words extracted by our approach	% of words extracted by autosummerizer.com	% of words extracted by our approach
1	Spaceflight	179	20	27.77	11.17
2	Painting	234	32	16.66	13.68
3	Architecture	237	31	16.66	13.08
4	Trees	357	45	18.18	12.61
5	Monocots	257	29	26.66	11.28
6	Aviation	51	10	21.43	19.60
7	Pop music	168	15	27.77	8.92
8	Charadriiformes	180	20	27.77	11.11

From the above table we can see that the percentage of words extracted by our approach is smaller than that of the existing approaches. A large number of words can give concise meaning in summary but it is not helpful for understanding the concept seeing only words. So, our approach can give a meaningful understanding of concept at a glance without reading it line by line.

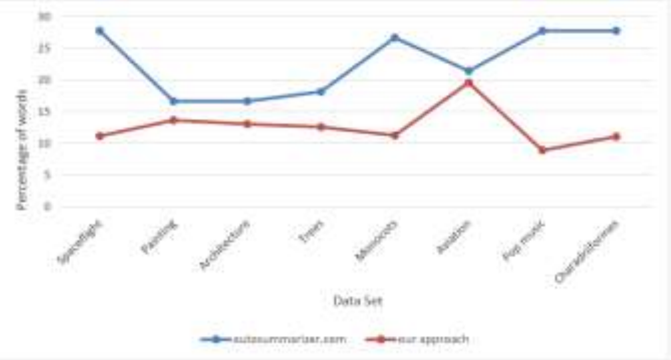


Fig- 4.17: Comparison of the word percentage for autosummarization.com and proposed approach

The clustered words generated by our technique can be further used in auto summarization or categorization. The words in each cluster is varied with different distance measurement techniques with respect to cluster centroid. So, the numbers of words can also be controlled for required text mining techniques.

4.4 Summary

Implementing and analyzing with different method of creating cluster, we can say that this thesis will be helpful and effective for user to extract a general perception of a text document. The key words can also be helpful for document categorization and summarization.

5. CONCLUSIONS

Revolution in technology makes remarkable change on web. A huge amount of textual information is published online on billions number of web pages. With the increasing amount of textual data every day, knowledge extraction and trend prediction have become a challenging task. The paper investigated the technique of text mining and word clustering to make a document understandable to the user. The system is implemented for finding the similarities among the words of a text focused on various methods for document pre-processing. Then, the k-means clustering algorithm was applied for grouping of similar words with similar sematic values.

Text mining is the most powerful and efficient technique for organizing the text corpus. The thesis based on this technique to discover an idea of a document. The thesis was accomplished by finding the similarities among the words to extract a general concept of the document. Word clustering technique is easier, less time consuming and will give a better concept of an ambiguous text document. The real world application of this paper would help people differentiate between documents having similar words but different contents.

LIMITATION

The implemented method only clusters words of a document on the basis of their similarities. Clustering of words give the insight of a general concept of the document. Sometimes, the exact topic or concept cannot be retrieved by our system. Moreover, the same words in different cluster may create confusion. Thus, understanding the actual topic of the document is become complex.

REFERENCES

[1] Vishal Gupta, Gurpreet S. Lehal (2009), “A Survey of Text Mining Techniques and Applications”, Journal of Emerging Technologies in Web Intelligence, Vol.1, No.1.

[2] Berry Michael W. (2004), “Automatic Discovery of Similar Words”, in “Survey of Text Mining: Clustering, Classification and Retrieval”, Springer Verlag, New York, LLC, 24-43.

[3] Navathe, Shamkant B. and Elmasri Ramez, (2000), “Data Warehousing and Data Mining”, in “Fundamentals of Database Systems”, Pearson Education pvt Inc, Singapore, 841-872.

[4] Haralampos Karanikas, Babis Theodoulidis Manchester, (2001), “Knowledge Discovery in Text and Text Mining Software”, Centre for Research in Information Management, UK.

[5] Mikolov, Tomas; Sutskever, Ilya; Chen, Kai; Corrado, Greg S. ; Dean, Jeff (2013).“Distributed representations of words and phrases and their compositionality.” Advances in Neural Information Processing Systems.

[6] Mikolov, Tomas; "Efficient Estimation of Word Representations in Vector Space” (PDF).

[7] C. Ding, X. He, H. Zha, H.D. Simon (2002), “Adaptive Dimension Reduction for Clustering High Dimensional”, Data,Proceedings of International Conference on Data Mining.

[8] Ms. J.Sathya Priya and Ms. S.Priyadharshini (2012), “Clustering Technique in Data Mining for Text Documents”, International Journal of Computer Science and Information Technologies, Vol. 3 (1), 2943-2947.

[9] Michael W. Berry and Malu Castellanos (2007), Editors “Survey of Text Mining:Clustering, Classification, and Retrieval, Second Edition”

[10] Hotho, A. Staab, S. Stumme,G. - "Ontologiess Improve Text Document Clustering", IEEE International Conference on, p. 541, Third IEEE International Conference on Data Mining (ICDM'03), 2003 Implementation, Morgan Kaufmann Press, 2000.

[11] Hisham Al-Mubaid and Syed A. Umair (2006), “A New Text Categorization Technique Using Distributional Clustering and Learning Logic” IEEE.

[12] Chen Wenliang, Chang Xingzhi, and Wang Huizhen (2004), “Automatic Word Clustering for Text CategorizationUsing Global Information” Copyright ACM.

[13] Tao Liu and Shengping Liu (2003), “An Evaluation on Feature Selection for Text Clustering” Proceedings of theTwentieth International Conference on Machine Learning(ICML-2003), Washington DC.

[14] Yanjun Li Congnan Luo (2008), “Text Clustering with Feature Selection by Using Statistical Data” IEEE.

[15] Navathe, Shamkant B. and Elmasri Ramez, (2000), “Data Warehousing and Data Mining”, in “Fundamentals of Database Systems”, Pearson Education pvt Inc, Singapore, 841-872.

[16] XiQuan Yang, DiNa Guo, XueYa Cao and JianYuan Zhou (2008), “Research on Ontology-based Text Clustering”, Third International Workshop on Semantic Media Adaptation and Personalization, China, IEEE Computer Society, 141-146.

[17] Dino Ienco Rosa Meo “Exploration and Reduction of the Feature Space by Hierarchical Clustering”Dipartimento di Informatica, Universit`a di Torino, Italy.

[18] Maheshwari, P. & Agrawal, J. (2010), "Centroid Based Text Clustering", Retrieved October 3, 2013.

[19] Ning Zhong, Yuefeng Li, Sheng-Tang Wu (2010), “Effect ive P att ern Discovery for Text Mining”, IEEE Transact ions on Knowledge and Data Engineering, C Copyright IEEE.

[20] Rahul Patel and Gaurav Sharma, “A survey on text mining techniques”, International Journal of Engineering and Computer Science ISSN: 2319-7242 Volume 3 Issue 5 May, 2014, Page No. 5621-5625

[21] Qiaozhu Mei and ChengXiang Zhai (2005),” Discovering evolutionary theme patterns from text: an exploration of temporal text mining.” In KDD, pages 198–207.

[22] Shixia Liu, Michelle X. Zhou, Shimei Pan, Yangqiu Song (2012), Weihong Qian, Weijia Cai, and Xiaoxiao Lian. TIARA: Interactive, “Topic-Based Visual Text Summarization and Analysis.” ACM TIST, 3(2):25.

[23] Sumit Goswami, Mayank Singh Shishodia (2011), “A Fuzzy based approach to text mining and document clustering”, International Journal of Computer & Organization Trends – Volume 11 Issue3.

[24] Vishwanath Bijalwan, Vinay Kumar, Pinki Kumari, Jordan Pascual (2014), “ KNN based Machine Learning Approach for Text and Document Mining”, International Journal of Database Theory and Application Vol.7, No.1, pp.61-70.

[25] Chihli Hung, Stefan Wermter (2014), “Neural-Net based document clustering using wordnet ontology”, International Journal of Engineering and Computer Science ISSN: 2319-7242 Volume 3 Issue 5.

[26] Principal component analysis

https://en.wikipedia.org/wiki/Principal_component_analysis. Last visited date: 13th July, 2016

[27] Ayush Agrawal , Utsav Gupta (2014), “Extraction based approach for text summarization using k-means clustering”, International Journal of Scientific and Research Publications ISSN: 2250-3153 Volume 4 Issue 11.

BIOGRAPHIES

	Md. Mehedi Rahman Rana Completed Bachelor Degree in Computer Science from Khulna University, Khulna, Bangladesh. Currently working as lecturer at Northern University of Business & Technology Khulna, Bangladesh.
	Rubya Afrin Completed Bachelor Degree in Computer Science from Khulna University, Khulna, Bangladesh. Currently working as lecturer at Northern University of Business & Technology Khulna, Bangladesh.
	Md. Anisur Rahman Completed Masters and PhD in Computer Science from University of Ottawa. Currently working as professor at Khulna University, Khulna, Bangladesh.
	Azazul Haque Completed Bachelor and Master’s Degree in Mathematics & Applied Mathematics from Dhaka University. Currently working as lecturer at Northern University of Business & Technology Khulna.
	Md. Ashiqur Rahman Completed Bachelor Degree in Computer Science from Khulna University, Khulna, Bangladesh. Currently working as lecturer at Northern University of Business & Technology Khulna, Bangladesh.