**CHAPTER ONE**

**INTRODUCTION**

With the dramatic growth of the Internet, people are overwhelmed by the tremendous amount of online information and documents. This expanding availability of documents has demanded exhaustive research in the area of automatic text summarization. Text document summarization is the process of reducing the size of documents while maintaining its relevant information. The aim of automatic text summarization is to reduce the source text into a compact version which will preserve contents and general meaning. A summary is a concise form of text that is composed from one or more texts that gives important information in the original text (Mullen **.**D. S, 2012). Summary reduces the reading time. Text summarization can be classified into extractive summarization and abstractive summarization. Extractive summarization means selecting the important sentence from the original documents. Abstractive summarization means express the meaning of the document in natural language. Although, technologies that can make a coherent summary must take into account variables such as length, writing style and syntax for any text document.

**1.1** **BACKGROUND OF THE STUDY**

Hybrid algorithms for text summarization comprise the using of optimization techniques in order to further summarize what the dimensional reduction algorithm had summarized. For the purpose of this project, we will be using Non-Negative matrix factorization algorithm as the dimension reduction with Artificial Bee Colony algorithm as the optimization techniques which shall be discuss further. Non-negative matrix factorization (NMF or NNMF), also non-negative matrix approximation on the other hand is a group of algorithms in multivariate analysis and linear algebra where a matrix V is factorized into (usually) two matrices W and H, with the property that all three matrices have no negative elements. This non-negativity makes the resulting matrices easier to inspect. Also, in applications such as processing of audio spectrograms or muscular activity, non-negativity is inherent to the data being considered. Since the problem is not exactly solvable in general, it is commonly approximated numerically (Inderjit S. Dhillon; SuvritSra, 2015)

NMF finds applications in such fields as computer vision, document clustering, chemometrics, audio signal processing and recommender systems (Yannis Sismanis, 2011). Many standard NMF algorithms analyze all the data together; i.e., the whole matrix is available from the start.

This may be unsatisfactory in applications where there are too many data to fit into memory or where the data are provided in streaming fashion. One such use is for collaborative filtering in recommendation systems, where there may be many users and many items to recommend, and it would be inefficient to recalculate everything when one user or one item is added to the system. The cost function for optimization in these cases may or may not be the same as for standard NMF, but the algorithms need to be rather different. The propose project “Hybrid Algorithm for Text summarization using Non negativity matrix factorization (NMF) and Artificial Bee Colony (ABC)” is a software that is aim at helping to optimizing text to a better but minimum useful context provision that carries the sense convey to manipulate the text and still give same result, without diminishing the grammar and lexical quality nor the semantic relationship between each terms in a giving string.

Optimization has been expanding in all directions at an astonishing rate during the last few decades. New algorithmic and theoretical techniques have been developed, the diffusion into other disciplines has proceeded at a rapid pace, and our knowledge of all aspects of the field has grown even more profound. At the same time, one of the most striking trends in optimization is the constantly increasing emphasis on the interdisciplinary nature of the field. Optimization has been a basic tool in all areas of applied mathematics, engineering, medicine, economics and other sciences. Optimization forms an important part of our day-to-day life. The task of optimization is that of determining the values of a set of parameters so that some measure of optimality is satisfied, subject to certain constraints. This task is of great importance to many professionals (Van den Bergh, 2006). Optimization problems arise in a variety of fields, including engineering design, operational research, information science and related areas. Effective and efficient optimization algorithms are always needed to tackle increasingly complex real world optimization problems. Stochastic optimization algorithms, such as genetic algorithm (GA) particle swarm optimization (PSO) ant colony optimization (ACO), biogeography-based optimization (BBO), harmony search (HS), and artificial bee colony (ABC) algorithm, have been shown to be successful in dealing with many optimization problems. Artificial Bee Colony algorithm, developed by Karaboga based on simulating the foraging behavior of honey bee swarm. Numerical comparisons demonstrated that the performance of ABC algorithm is competitive to other population-based algorithms with an advantage of employing fewer control parameters (D. Karaboga, B. Basturk, 2016). Due to its simplicity and ease of implementation, ABC algorithm has captured much attention and has been applied to solve many practical optimization problems since its invention in 2005. In practice, the exploration and exploitation contradicts to each other.

In order to achieve good performances on problem optimizations, the two abilities should be well balanced. Therefore, accelerating convergence speed and avoiding the local optima have become two most important and appealing goals in ABC research. A number of variant ABC algorithms have, hence, been proposed to achieve these two goals (B.Alatas, 2014). However, so far, it is seen to be difficult to simultaneously achieve both goals. For example, the chaotic ABC algorithm (CABC3) in (S. Rahnamayan, et al., 2008) focuses on avoiding the local optima, but brings in a more extra function evaluations in chaotic search as a result. To achieve both goals, inspired by DE (Differential Evoluton), ABC algorithm called ABC/best, which is based on that each bee searches only around the best solution of the previous iteration to improve the exploitation. In addition, to enhance the global convergence, when producing the initial population and scout bees, both chaotic systems and opposition-based learning method are employed.

* 1. **STATEMENT OF THE PROBLEM**

In the era of internet, online information is freely available for readers in the form of e-Newspapers, journal articles, technical reports, transcription dialogues etc. There is huge number of documents available in above digital media. Therefore it is very serious issue to get data fast and efficiently and extracting only relevant information from all these media is a tiresome for the individuals in stipulated time. There is a need for an automated text summarization that can extract only relevant information from the original document.

**1.3 AIM AND OBJECTIVES OF THE STUDY**

The core aim of this system: Text summarization using Non negativity matrix factorization (NMF) and Artificial Bee Colony (ABC) is to help in optimizing text to a better but minimum useful context and still give same result, without diminishing neither the grammar, lexical quality nor the semantic relationship between each term in a giving document. The Text summarization software is a desktop based application which has been targeted to adopt the following objectives.

1. To develop an application that seeks to provide quality and accurate summary of text document.
2. To provide a better means of clustering document with the intention of getting the major points from the original document.
3. To develop an interactive and user friendly interface application for document summarization.

**1.4 SIGNIFICANCE OF THE STUDY**

Extracting relevant information from the original document is very challenging, because when we as humans tend to summarize a piece of text, we usually read entire document to develop our understanding, and then write a summary by highlighting its main points. Doing this as human is tiresome for the individuals in stipulated time. Text summarization application helps to reduce the reading time and energy required to read the entire text document and as well help in getting data very fast**.**

**1.5 SCOPE OF THE STUDY**

The scope of study is to design an application that will achieve an optimal dimension reduction in a text document. This system may be integrated with third party application either for application development or research work.

**1.6 LIMITATION OF STUDY**

1. This project is a Text Summarization using NMF and ABC i.e. this project does not automatically address any error from the used or referenced libraries), plug-ins sources.
2. This project is not the solution to every problem on text optimization, and it does not promise total security except frequent improvement on it.
3. This project is limited to the information gathered during the development of the project.
4. The project can only be use for text extraction; therefore it cannot be use for text abstraction.

**1.7 DEFINITION OF TERMS**

* **Document summarization:** is the process of shortening a text document with software, in order to create a summary with the major points of the original document.
* **Artificial Bee Colony techniques:** In computer science and operations research, the **Artificial Bee Colony algorithm** (ABC) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs.
* **E-Assessment:** Electronic assessment is the use of technology to manage and deliver assessment
* **Optimization:** This is the process of minimizing or maximizing opportunities as best suit to the situation.
* **Pheromones**: are chemical substances which attract other bee searching for food.
* **Computer system:** These is an electronic device that is capable of accepting data, process it following a preset logic and generate output as require by the user.
* **Program, software or application:** A set of logical instruction combined together to get data and perform a specific task on a given data.
* **Input:** Data supplied to the computer for processing.
* **Output:** The result of a processed data.
* **Data:** A raw fact yet to be processed.
* **Information:** Data that has been proceed.

**CHAPTER TWO**

**LITERATURE REVIEW**

**2.1 HISTORICAL BACKGROUND**

The theory of optimization has its roots in the isoperimetric problem faced by Queen Dido in 1000 BC. She procured for the founding of Carthage the largest area of land that could be surrounded by the hide of a bull. From the hide she made a rope, which she arranged in a semicircle with the ends against the sea. Queen Dido’s intuitive solution was correct. But it was many centuries before a formal proof was presented, and the mathematical and systematic solution to this problem proved to be a very difficult problem in the calculus of variations. The calculus of variations essentially handles problems where the decision variable is a vector. The calculus of variations, or the first systematic theory of optimization, was born on June 4, 1694, when John Bernoulli posed the Brachistochrone (Greek for “shortest time”) problem, and publicly challenged the mathematical world to solve it. The problem posed was, “What is a slide path down which a frictionless object would slip in the least possible time?” Earlier attempts to solve this problem were made by many well-known scientists including Galileo who proposed the solution to be a circular arc, an incorrect solution, and Leibnitz who presented ordinary differential equations without solving them. Then John Bernoulli proved the optimal path to be a cycloid. From that point, efforts continued in the area of the calculus of variations, leading to the study of multiple integrals, differential equations, control theory, problem transformation and so on. Although this research mainly involved theory and analytical solutions, it formed the basis for numerical optimization developed during and after World War II. World War II made scientists aware of numerical optimization and solutions to physics and engineering problems. In 1947 Dantzig proposed the simplex algorithm for linear programming problems. Necessary conditions were presented by Kuhn and Tucker in the early 1950s, which formed a focal point for non-linear programming research. Now, numerical optimization techniques constitute a fundamental part of theoretical and practical science and engineering (Diwekar, 1995).Optimization has pervaded all spheres of human endeavor. Although optimization has been practiced in some form or other from the early prehistoric era, this area has seen progressive growth during the last five decades. Modern society lives not only in an environment of intense competition but also constrained to plan its growth in a sustainable manner with due concern for conservation of resources.

Thus, it has become imperative to plan, design, operate, and manage resources and assets in an optimal manner. Early approaches have been to optimize individual activities in a standalone manner, however, the current trend is towards an integrated approach: integrating synthesis and design, design and control, production planning, scheduling and control. The functioning of a system may be governed by multiple performance objectives. Optimization of such systems will call for special strategies for handling the multiple objectives to provide solutions closer to the systems requirement. Uncertainty and variability are two issues which render optimal decision making difficult. Optimization under uncertainty would become increasingly important if one is to get the best out of a system plagued by uncertain components. These issues have thrown up a large number of challenging optimization problems which need to be resolved with a set of existing and newly evolving optimization tools (Daniel D. Lee & H. Sebastian Seung, 2011).

Optimization theory had evolved initially to provide generic solutions to optimization problems in linear, non-linear, unconstrained, and constrained domains. These optimization problems were often called mathematical programming problems with two distinctive classifications, namely linear and non-linear programming problems. Although the early generation of programming problems were based on continuous variables, various classes of assignment and design problems required handling of both integer and continuous variables leading to mixed integer linear and non-linear programming problems (MILP and MINLP). The quest to seek global optima has prompted researchers to develop new optimization approaches which do not get stuck at a local optimum, a failing of many of the mathematical programming methods. Genetic algorithms derived from biology and simulated annealing inspired by optimality of the annealing process are two such potent methods which have emerged in recent years. The developments in computing technology have placed at the disposal of the user a wide array of optimization codes with varying degrees of rigor and sophistication. The challenges to the user are many fold. How to set up an optimization problem? What is the most suitable optimization method to use? How to perform sensitivity analysis? An intrepid user may also want to extend the capabilities of an existing optimization method or integrate the features of two or more optimization methods to come up with more efficient optimization methodologies.

**2.2 Related work**

Various works has been done by different researchers on a variety of research topics using NMF. ABC is also a favorite choice for researchers for solving different kind of optimization problems. These two topics are relatively new than the CREDIT scoring problem. For a long time people are dealing with credit scoring. High investor risk and low consumer satisfaction was the root cause for required improvement in this area. Now researchers are using different techniques for reducing risk and improving satisfaction and they are also successful up to some extent. One of the most interesting and successful applications of NMF is to cluster data such as text, image or biology data, i.e. discovering patterns automatically from data. In many cases, some background information concerning the pair wise relations of some samples are known and we can add them into the clustering model in order to guide the clustering process. The resulting constrained problem is called semi-supervised clustering. The incorporated user provided constraints in data clustering (Chen and Rege, 2015). In the stock market, it has been observed that the stock price fluctuations does not behave independently of each other but are mainly dominated by several underlying and unobserved factors. Hence try to identify the underlying trends from the stock market data is an interesting problem, which can be solved by NMF (Konstantinos, 2008).

Building appropriate financial distress prediction model based on the extracted discriminative features is more and more important under the background of financial crisis. Ref presents a new prediction model which is indeed a combination of K-means, NMF and Support Vector Machine (SVM). The basic idea is to train a SVM classifier in the reduced dimensional space which is spanned by the discriminative features extracted by NMF, the algorithm of which is initialized by K-means (Bernardete, Catarina, Armando, &Joao, 2009).

**2.3 ARTIFICIAL BEE COLONY (ABC)**

ABC is an optimization algorithm, which imitates the real acts of honey bees (Karaboga&Dervis, 2005). The most important components of ABC algorithm are its food source, employed and unemployed bees. The main theme of this algorithm is to arrive at the best food source. The standard pseudo code of the ABC algorithm is presented in algorithm (Das, Swagatam, and Amit Konar, 2009). The most basic ABC algorithm consists of three phases. They are initialization, employed, onlooker and scout bees phase. Each phase is replayed until the maximum count of

iterations are reached. In the initial phase, the count of solutions and the control parameters are fixed. The employed bees phase deals with the search of new high quality food sources in the nearby locality of old food source. The new food source is then evaluated for its fitness, which is then followed by the comparison of the old and the new food source by means of greedy selection. The collected knowledge about the food source is distributed among the onlooker bees present in the beehive. In the next phase, the onlooker bees follow a probabilistic approach to select the food sources with respect to the information provided by the employed bees. This is followed by the calculation of the fitness function of the food source, which is located nearby the selected food source. Finally, the old and the new food sources are compared by the greedy selection. In the final phase, the employed bees turn to scout bees, when their solutions cannot be enhanced within a predefined count of iterations. The solutions so found by the bees are dropped out. At this point, the scout bees search for new food source again. Using this functionality, the poor solutions are dropped out. These three phases continue its process until the stopping point is reached (Karaboga, Dervis and CelalOzturk, 2014).

The below is the ABC algorithm

1:Input: Training data;

2: Produce initial population

3: Calculate the fitness function of the population

4: Fix counter=1

5: Do

// Employed bees phase

6: Search for the food source;

7: Calculate the fitness function;

8: Employ greedy selection process;

9: Compute the probability for the food source;

// Onlooker bees phase

10: Select food source based on the probability values;

11: Generate new food source;

12: Calculate the fitness function;

13: Apply greedy selection process;

// Scout bees phase

14: If food source drops out then swap it with new food source;

15: Save the best food source;

16: Counter + =1;

17: While counter=MC;



Figure.1 ABC Food Source Diagram

(Abiyot.B, 2010)

**2.4 NON NEGATIVE MATRIX FACTORIZATION (NMF)**

Non-negative Matrix Factorization (NMF) is recent development for document clustering. Initial work on NMF (Lee &Seung 1999; 2001) emphasizes contain coherent parts of the original data (images). Later work (Xu, Liu, & Gong 2003; Pauca et al. 2004) show the usefulness of NMF for clustering with in experiments on documents collections, and a recent theoretical analysis (Ding, He, & Simon 2005) shows the equivalence between NMF and K-means / spectral clustering.

In many settings in science and engineering the observed data are admixtures of multiple latent sources. We would typically want to infer the latent sources as well as the admixture distribution given the observations. Non-negative matrix factorization (NMF) is a natural mathematical framework to model many admixture problems. In NMF we are given an observation matrix M 2 Rnm, where each row of M corresponds to a data-point in Rm. It is assume that there are r latent sources, modeled by the unobserved matrix W 2 Rrm, where each row of M characterizes one source. Each observed data-point is a linear combination of the r sources and the combination weights are encoded in a matrix A 2 Rnr. Moreover, in many natural settings, the sources are non-negative and the combinations are additive. The computational problem is then is to factor a given matrix M as M = AW, where all the entries of M;A and W are non-negative. We call r the inner-dimension of the factorization, and the smallest possible r is usually called the non-negative rank of M. NMF was first purposed by (Lee &Seung, 2009), and has been widely applied in computer vision (Lee &Seung, 2000), document clustering (Xu et al., 2003), hyperspectralunmixing (Nascimento& Dias, 2004; Gomez et al., 2007), computational biology (Devarajan, 2009), etc. Non-negative Matrix Factorization has been proved to be valuable in many fields of data mining, especially in unsupervised learning. The special point on NMF is its ability to recover the hidden patterns or trends behind the observed data automatically, which makes it suitable for image processing, feature extraction, dimensional reduction and unsupervised learning. Intuitively there are three ideas on disguising sensitive data. One is to transform original data into protected, publishable data by data perturbation. An alternative to data perturbation is to generate a new dataset (synthetic dataset), not from the original data, but from random values that are adjusted in order to have the same feature pattern as the original data. A third possibility is to build a hybrid dataset as a mixture of a distorted one and a synthetic one. The idea of positive matrix factorization is developed by P. Paatero at the University of Helsinki, and to be popular in the computational science community. Interest in positive matrix factorization increased when a fast algorithm for Non-negative Matrix Factorization (NNMF), based on iterative update, was developed by (Lee and Seung, 2010), particularly as they were able to show that it produced intuitively reasonable factorizations for a face recognition problem. NNMF has recently been shown to be very useful technique in approximating high dimensional data where the data are comprised of non-negative components. NNMF is a vector space method to obtain a representation of data using non-negative constraints. These constraints can lead to a parts-based representation because they allow only additive, not subtractive, combinations of the original data. This is in contrast to techniques for finding a reduced dimensional representation based on SVD (single value decomposition).

**2.5. THE FRAMEWORK FOR THE PROPOSED HYBRID SYSTEM NMF AND ABC FOR TEXT SUMMARIZATION**

The proposed hybrid system combines the structural and semantic features based on the weights calculated by either the NMF or the ABC algorithm. Using the combined sentence score function, it then ranks the sentences in the document and extracts the highest scored sentences to generate a summary. A detailed description of the 2 types of sentence features is presented below.

**2.5.1.** Fi**-Structural features**

The structural features included in our model principally depend on the structural analysis of the sentences in the document. These features are the ‘f n: Length’, ‘fi2: Position’, ‘fi3: Title’,

fi4: Frequency’, and ‘ f *i5* : Class relevance’ features.

A detailed explanation of these features is given below.

**F i i - *Length***: The use of this feature is motivated by the idea that sentences are important if the number of words in them is within a certain range. After the stop words are eliminated and the stemming is applied, each sentence is given a length score, which is the number of words contained in the sentence

**F12 *-* Position**: Sentences at the beginning of the documents always introduce the main topics that the documents describe. To capture the significances of different sentence positions, each sentence in a document is given a rank.

**F i 3 *-*Title**: This feature is based on the assumption that the sentences are important if they contain the title words of a document. After the stop words are eliminated and the stemming is applied, each sentence is given a title score by summing the number of overlapping words between the title and the sentence.

**F i 4 *-*Frequency**: This feature depends on the intuition that words occurring frequently within a document usually has salient information and that sentences with a higher number of such words are important. After the stop words are eliminated and the stemming is applied, each sentence is given a frequency score by summing the frequencies of the constituent words.

**F i 5 *-*Class relevance**: This sentence feature is a novel sentence feature that applies the text classification task for summary generation. In order to obtain this feature, first of all, each document to be summarized is classified using the multinomial naive Bayes algorithm. The classifier is trained with the 1150 documents obtained from the study. This data set contains 5 different classes (economy, magazine, health, political, and sports) of documents and there are

230 documents in each class. Additionally, from this data set, the most frequent unigram, bigram, and trigram word combinations in each class are stored in 5 separate dictionaries. For the document to be summarized, we count the matching unigram, bigram, and trigram words in each sentence using the N-gram word dictionary of its class. If a sentence in the document to be summarized contains these frequent N-gram words in the dictionary for the class of the document, we assume that this sentence is important for the summary generation. Therefore, we assign a sentence score to each sentence according to the number of matches between its N-gram words and the related frequent N-gram word dictionary. The above procedure is applied after the stop words are eliminated and the stemming is performed on both the text categorization and the summarization data sets.

**2.5.2 Creation of input matrix**

In order to extract the above features for text summarization, a document is represented as an *m x n* term sentence matrix *A* = (a1j*, a* 2j...a nj), where each entry aijis obtained by multiplying a local and a global weighting factor as follows: aij= L(tij) *G* (tij*)*. Here, *L* (tij) is defined as *L* (tij) = log (1 + *t* f *(*tij)) and *G* (tij) is defined as *G* (tij) = log (N-) + 1, where *t f* (tij) is the number of times that term *t j* occurs in the sentence, *N* is the total number of sentences in the document, and Niis the number of sentences that contain term tij*.* In this study, when the matrix *A* is created, instead of considering the words individually, we detect syntactically related words in the documents and treat them as a single word. For example, Mustafa Kemal Ataturk (the founder of the Turkish Republic), is considered as a single term. In order to find these syntactically related words, we use Turkish Wikipedia (Vikipedi). The main purpose of mining Vikipedi is to extract information by analyzing web links. There are several successful studies that use Wikipedia as an external knowledge resource to enrich text mining applications. In a novel method called explicit semantic analysis (ESA) was presented to get better performance for text classification systems with Wikipedia. In their approach, they use a semantic interpreter to represent each text document as a weighted vector of Wikipedia concepts. They then add these Wikipedia concepts to a traditional bag of words approach as new features.

Their results show that the ESA with Wikipedia improves the correlation of the computed semantic relatedness score with humans. This study presented a single-document summarization method that maps document sentences to semantic concepts in Wikipedia and selects sentences based on the frequency of the mapped-to concepts. Their results indicate that the Wikipedia-based summarization method is competitive with the state of single document summarization. The study worked on categorization through syntactically related word associations and the study in [34] used syntactically related words for topic segmentation and link detection. The underlying motivation of these approaches comes from the observation that syntactically related word associations may be used to represent the gist of the semantic content of a document. Although there are numerous studies using the English Wikipedia in semantic analysis, there are a limited number of studies using Vikipedi. The employed Vikipedi to discover missing links in a Vikipedi article. The study integrated semantic information into the suffix tree clustering algorithm using Vikipedi. Knowledge-based word sense disambiguation methods were compared for Turkish texts, using Turkish WordNet as a primary knowledge base and Vikipedi as an enrichment resource. In another study an automatic Turkish document summarization system was built. In that study, the NMF-based summarization algorithm was used with syntactically related word associations. Wikipedia contains many different types of semantic relationships, such as synonymy, polysemy, categorical information, and hyperlinks, between articles. In our study, we only use the semantic relationship of words that occur literally. Wikipedia has two important characteristics: the dense web link structure and the concept identification by the web links, called uniform resource locaters (URLs). Articles are strongly connected to each other by this dense structure of web links. Almost every concept (article/page) has its own URL as an identifier (i.e. consecutive words that occur in a single URL represent a single concept or entity). In order to find these concepts or entities, all URLs are searched in Vikipedi and the syntactically related words in the links, such as Recep Akdag (name of a person), Anayasa Mahkemesi (Constitutional Court), Saglik Bakanligi (Ministry of Health), and Domuz Gribi (swine flu), are selected. This modification provides semantic integration between consecutive words. In this work, all of the semantic features are extracted after the syntactically related word detection phase and we show that the performance of this modification shows promising results.

**2.5.3 Similarity Measure**

The process of clustering completely depends on the similarity of terms or documents, as similar entities can alone be grouped or clustered. This step is the predecessor of the clustering process. The similarity measure determines the level of association between the documents. There are several performance similarity measures such as Euclidean distance, cosine similarity, Jaccard coefficient and Pearson correlation coefficient (T Li, C Ding, 2008). This work proposes to incorporate cosine similarity measure because of its wider range of applications in text mining.

The main objective of a similarity measure is to obtain the degree of association between two documents.

**2.5.4 Clustering algorithm**

A combination of Artificial Bee Colony and k-means algorithm is proposed for clustering the web documents. ABC colony algorithm is an efficient population based optimization algorithm and it imitates the behaviour of real bees. The k-means algorithm is efficient and fast, however the problem is on finding initial cluster point. This work proposes to locate the initial cluster point with the help of bees and these clusters are refined by the k-means algorithm. We propose to combine both ABC and k-means algorithm, so as to inherit the merits of both the algorithms. ABC is efficient but consumes more time for convergence. The k-means algorithm is also known for its faster convergence but struggles in locating the initial cluster point. Thus, a new algorithm is presented for improving the efficiency and reducing the execution time. The steps involved in the proposed algorithm are explained below.

**Step 1:** Parameters initialization

The parameters that need to be initialized are the maximum count of iterations or the maximum time can be given in milliseconds, position of the food source (cluster center) and occurrence frequency threshold of the terms for labeling clusters. The fitness value for this algorithm is the degree of relevance between two documents. Thus, the initial populations of food sources are distributed randomly.

**Step 2:** Document pre-processing

This step is concerned with the removal of articles, connectors, prepositions and pronouns. This step is to weed out the unwanted terms, so as to make the clustering process effective. The stop words are eliminated. Some of the sample stop words which were removed from the documents are presented in table 1 below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sample stop words** | | | |
| a | an | the | about |
| above | According | despite | along |
| again | among | apart | around |
| before | after | between | but |
| by | because | with | without |
| over | under | near | on |
| of | till | until | in |
| below | behind | beside | beyond |
| you | me | I | we |

Table 1. Sample stop words (Mesfin. G, 2015)

Let Doc = (Doc1, Doc2, Doc3 … Doc i) be the accumulation of documents and be the term set of the documents Doc, where TS= (T1, T2, TK) in this work, every document is defined as a point in a dimensional vector space. Thus, the dimension is based on the term set of the document. Every dimension denotes a different term and it is Doc M= (wtM1, wtM2, wtM3, wtMk) where the value of *m* ranges between 1 to *i*. Each element of DocM is the weight of the terms present in the document. Thus, DocM is based on the level of association between the term and the document.

**Step 3:** The execution of the proposed algorithm depends on the source of food, which are the solutions. Let food source Fs*=* 1*,* 2..i is a k dimensional vector, where kis the multiplicative result of documents and the cluster size. The initial count of documents in a cluster is 2 and the maximum number of documents to be in a cluster is 8.

**Step 4:** In this step, the fitness of the population is calculated by the Equation (6) and is given below.



Where, c is the cluster, CC is the cluster center and doc is the document. The above equation determines the distance between the document and the cluster center. The main objective is to have minimal fitness value.

**Step 5:** After finding the fitness of the population, the employed bees search for the new food source in its neighborhood and provides a new food source from its locality. This new food

source is tested for its fitness by the K-means algorithm and the greedy selection is applied. In case, if the degree of similarity (fitness) of the new document with the cluster center is more than

the similarity between the old document and the cluster centre, then its memory is loaded with the new document and is computed by Equation (7). By this way, the employed bee search for the better documents with respect to the cluster centre. This is followed by the calculation of probability of the food source and is calculated by Equation (8).



The employed bees search for the new documents with better degree of relevance with respect to the cluster centre, in its neighborhood. In (7), ai*,*jis the location of the initial document and is stored in memory of the employed bee, ai*,*j is the new document and *d* is in the range of [1, -1]. The new document is found by changing a dimension over *a*. By this way, the employed bee moves in its neighborhood and the location bound is reset.



Where, fi is the fitness of the ithdocument and n is the total number of cluster centers. Thus, the probability is calculated.

**Step 6:** The onlooker bee, which is waiting on the bee-hive, chooses the document based on the so calculated probability value and tries to find a document in the neighborhood. If a document is found in its neighborhood, then the degree of relevance is found. This is followed by the application of K-means and the greedy selection, as in step 6. This notifies that any number of onlooker bees can probabilistically select a single document with high fitness value. Finally, the obtained best solution is saved. This process continues till the stopping point for execution is reached.

**2.6 OTHER TECHNIQUES USED IN SUMMARIZING A FREE TEXT.**

In this part, we will outline new and existing approaches to automatic summarization of free text. Note also that most of these approaches are generic and apply to other languages especially European language.

**2.6.1 Topic based approaches.**

Moreover (Teng, et al; 2009) propose an approach which combines the automatic topics identification technique with the terms frequency method. This methodology consists of calculating initially the similarity between the sentences, then carry out the identification of the subject covered by gathering similar sentences in clusters. In a second stage, and based on terms frequency, the projecting sentences are selected starting from the local topics already identified. (Kuo and Chen; 2010) use not only the frequency of terms to detect relevant information in a text, the authors also use informative words and event-driven. This type of words indicates concepts and the important relations which can be used to detect important sentences in the text.

**2.6.2 Graphs Based Technique.**

LexRank and TextRank are the most important algorithms used in automatic summarization system based on graph method. In the same context, proposed a method based on graphs algorithm for automatic texts summarization. This method consists in building a graph from the text. Nodes of the graph are represented by the text sentences, for each sentence there is a node. The edge of the graph represent connection (lexical or semantic) between the sentences, this connection is evaluated by calculating the similarity between the sentences. The weight of each node is calculated by using the function COS (Cosine). After that the summary is made up by taking the shortest way which starts with the first sentence of the original text and finishes with the last sentence.

**2.6.3 Latent Semantic Analysis (LSA) Based Approaches**

LSA is an algebraic-statistical method that extracts and represents semantic knowledge of the text based on the observation of the co-occurrence of words. This technique aims to builds a semantic space with very large dimension from the statistical analysis of the whole co-occurrences in a body of texts. The starting point of LSA consists of a lexical table which contains the number of occurrences of each word in each document. (Gong & Liu; 2012) proposed an automatic text summarization system of news text with the use of LSA as a way to identify the important topics in the documents without using lexical resources like WordNet. In this way, the SVD (Singular Value Decomposition) is applied to matrix *A* to decompose into three new matrices as follows: *A = UWVT*. The suggested that the row of the matrix *VT* can be considered as various topics covered in the original text, while each column represents a sentence in the document. And finally, in order to produce an extractive summary, they consider each row of matrix *VT* consecutively, and select the sentence with the highest value

**2.6.4 Approach Based on Fuzzy Logic**

In (Farshad, et al. 2011), another approach to automatic summarization has been proposed, this time it is based on a fuzzy logic. This method takes into account every feature of the text such as word frequency, similarity to keywords, similarity to the title words, sentences position, statistics of co-occurrence of lexical chain, indicative expression etc. After extracting these features and depending on the results, a value of 0-1 is assigned to each sentence of the text according to the characteristics of sentences and rules available in the knowledge base. The value obtained at the output determines the degree of importance of the sentence in the final summary. In (Esther Hannah, et al. 2012) , different characteristics of each sentence were taken into account, such as title words, sentence length, term weight, Sentence to sentence similarity, etc. the values of these features are used by the inference engine to generate the score of each sentence of the text.

|  |
| --- |
|  |

**CHAPTER THREE**

**SYSTEM ANALYSIS AND DESIGN**

**3.1 RESEARCH APPROACH**

This research adopts a Text summarization using NMF and ABC approach to optimize documents for proper processing and accurate mining of text within documents. The project will take different documents as input, processes the documents in other to extract stop words from it and the output of the project will be important word space that quantifies and qualifies the meaning of the words in each documents.

The various activities carried out and different modules implemented to ensure application feature behaviors are intercepted for the use of Text summarization are

1. Input Document
2. Extract Text
3. Extract Text stop words
4. Check for Repeated Text
5. Intercept on a table to generate output
6. Display summarized text

Load Document from specified location

Hybrid Algorithm For Text Summarization system

Display the summarized and compressed document output

Load summarized output for further analysis

Remove repeated word using stemming

Manipulate loaded document to produce summarized output

**Figure 3.1: Architecture of Hybrid Text summarization using NMF and ABC**

(Kamil. N, 2012)

**3.1.1 Materials**

The application of the Hybrid Text summarization using NMF and ABC will be test on a laptop machine with an intel-centrino-dual2-pro processor, 4GB of available memory and 500 GB Hard Disk Drive (HDD). This machine runs windows 8 Operating System while NetBeans 7.2 Integrated Development Environment (IDE) will be used as the Java Developer Kit (JDK). The server used to test runs the project is Xampp that contains apache for rendering the application.

**3.1.2 Methods**

The approach toward which this project will take is a desktop application that will mainly be developed to address the issue of text summarization. This project which will be develop using java technology such as word processor, text extractor or abstractor and some other related tools for text summarization.The application will be tailor toward desktop text summarization in other to provide Automatic summarization by processing shortening a text document in order to create a summary with the major points of the original document. Technologies that can make a coherent summary take into account variables such as length, writing style and syntax.

**3.2 PROCEDURAL MODEL OF NMF**

Stop words removal

Lexical analysis of text

Obtaining optimal dimension using ABC

Term weighting

Term-document matrix construction

stemming

Dimension reduction using NMF

Ranking/similarity measurement using cosine rules

Document collection

**Figure 3.2: System Procedural Diagram (**Jones K. S,2017)

**3.2.1 Document Collection**

The source of the document shall be a document containing text which can either be course material and the students’ free text response to each question. The answer to each question by the lecturer and the student will constitute a document of its own. That is each student will generate five documents for a set of five questions. The responses shall be supplied as a soft copy through an interface designed to capture the data. These documents will serve as the input data to the system. Obviously, the responses are not expected to be the same in spelling but some will be closer in relational concept. This perfectly fit into the description of Non-negative matrix factorization which discovers the semantic relationship between keywords and documents in the document set in order to achieve the goal of concept based assessment by eliminating the influence of different word usage.

**3.2.2 Document processing and Term Extraction**

Document processing comprises of three stages which are as follows:

**3.2.2.1 Lexical Analysis of text**: this is the process of converting a sequence of characters into a sequence of tokens (strings with an assigned and thus identified meaning). It breaks down streams of text up into words, phrases, symbols or other meaningful element called tokens. The list of tokens becomes input for further processing.

**3.2.2.2 Stop words removal:** This is the elimination of the most common words such as: the, as, a, in, to etc. in a language. These words are usually excluded from the analysis as they do not contribute much (if any) meaning to a context. The filtered words become input for further processing.

**3.2.2.3 Stemming:** This is the process of reducing inflected (or sometimes derived) words to their word stem, base on root form. The stem needs not to be identical to the morphological root of the word. A stemming algorithm reduces the word “fishing”, “fished” and “fisher” to the root word “fish”.

The purpose of term extraction is to generate list of terms that are relevant to the input domain. A well generated or extracted term will facilitate the assessment process. It is made up of two stages which are the training stage and the extraction stage. At the training stage, a model is created for identifying terms using training documents. The extraction stage chooses term from a test document using the model that was created at the training stage.

The training stage makes use of the following procedure:

1. Read the input document which is the training document
2. Extract noun phrase from each sentence in the training document using syntactic parser. The parser will analyze each sentence and generate a list of syntactic information such as Noun, Noun-Phrase etc.
3. The extracted noun and noun phrase are preprocessed and stem to remove stopwords and produce a list of clean noun phrase as term.
4. A set of five features(domain relevance, domain consensus, term cohesion, first occurrence and length of noun phrase) are calculated for each candidate term which are subsequently used to calculate the score and rank the term based on their score.

The extraction stage consists of the following procedures:

1. Read the test documents
2. Perform preprocessing operation
3. Perform feature generation
4. Conduct term ranking
5. Generate list of terms

**3.2.3 Term document matrix construction:** In this stage, each row represents documents in the collection and each column a term, and respective cells of the matrix contain the frequencies with which the term occurs in the document (Each cell contains the number of times that index word occurs in the document). In general, the matrices built during NMF tend to be very large, but also very sparse (most cells contain 0). That is because each document usually contains only a small number of all the possible words. This sparseness can be taken advantage of in both memory and time by more sophisticated NMF implementations.

**3.2.4 Term weighting generation:** In Latent Semantic Analysis systems, the raw matrix counts are usually modified so that rare words are weighted more heavily than common words. For example, a word that occurs in only 5% of the documents should probably be weighted more heavily than a word that occurs in 90% of the documents. The reason for this because rare words reveal better similarity features among documents. The most popular weighting is TFIDF (Term Frequency - Inverse Document Frequency). Under this method, the count in each cell is replaced by the following formula.

For the purpose of term generation, a set of five features are used to characterize the noun phrases in the document. These features are calculated for each term and are used at both stages (i.e. training and extraction stages). The features are:

1. **Domain Relevance:** It is a measure of the amount of information captured in the target document with respect to contrastive documents. If Di is a set of relevant document in a domain of interest Di and [Di…Dn] is the sets of documents in another domain.
2. **Domain Consensus**:-It measures the distributed use of a term in a Domain Dk.
3. **Term Cohesion:** It is used to calculate the cohesion of the multi-word terms. This measure is proportional to the co-occurrence frequency and the length of the term.
4. **First Occurrence**: It is calculated as the number of words that precede the phrase’s first appearance, divided by the number of words in the document. The result is a number between 0 and 1 that represents how much of the document precedes the phrase’s first appearance.
5. **Length of noun phrase**: candidate length is also a useful feature in extraction as well as in candidate selection, because the majority of terms are one or two words in length. Length of noun phrase score is calculated as its frequency times its length (in words).

**3.2.5 Dimension reduction using NMF**

Dimension reduction using NMF has been observed in literatures to have the following setback: It does not lead to proper storage management and conservation, the presence of negative value in the cell of term-document matrix makes it un-interpretable. This research intends to solve these problems by hybridizing NMF with ABC a particle swarm optimization techniques.

**3.2.6 The Mechanism for Application Design**

Application Designed Interface(s) with Java codes

NMF, ABC document processing system

Xampp (MySQL) server database design holding the application data as a repository

**Figure 3.3: Application Design and Program**

**3.3 HYBRID TEXT SUMMARIZATION USING NMF AND ABC ACTIVITIES**

**Figure 3.4: Data flow Diagram (** Lloret. E, 2010)

Document location

Summarized Text

NMF processor

Text Extraction

ABC Optimizer

Document Term weight output

Get provided document

The following are the systems activities for the Hypermedia Optimization Using Adaptive Compressor

Step 1: Supply document location.

Step 2: Extract Text from each documents.

Step 3: Load the extracted text for optimization using NMF.

Step 4: Load the optimization output for factorization using ABC.

Step 5: Generate the output of the summarized text.

**CHAPTER FOUR**

**SYSTEM IMPLEMENTATION AND**

**DOCUMENTATION**

**4.1 SYSTEM IMPLEMENTATION**

The implementation phase of this project is more concerned with the integrating of non-negative matrix factorization (NMF) and artificial bee colony (ABC) optimization techniques on free text document for text summarization service with the use of formatted free text document, by installing the Text Mining Language (TML) and it applications from the appropriate respective online/offline repository. The application required Java Virtual Machine (JVM) which is available on most recent Operating Systems. In case the JVM is not available on the machine, then, one has install Java Developer Kit (JDK), the application also required the use of text mining language (TML). It invariably means that the coding perspective of building the system provides the blue print for the system and helps provide the platform for the user.

**4.2 CHOICE OF PROGRAMMING LANGUAGE**

The technologies used in building the code are Java, structured query language (SQL) server and text mining language (TML).

1. Java is an object-oriented programming language. It provides support for software engineering principles such as checking array sounds checking, detection of attempts to use uninitialized variables and automatic garbage collection. It also provides software justness, durability, and programmers' productivity. The language is used in developing software component suitable for deployment in distributed environments.
2. SQL (Structured query language) server is a relational database management system first developed by Microsoft in 1989. As a database, it is a software product whose primary function is to store and retrieve data as requested before other software application be it those on the same computer or those running on another computer across a network including the internet.

**3**. TML (Text mining language) is a standard plugging for text factorization and optimization having various kind factorization techniques such as NMF (Non- negative matrix factorization), LSA (Latent semantic analysis), etc. this technology is built on java framework to mine data of huge.

**4.3 HARDWARE AND SOFTWARE SPECIFICATION**

**This is the Section where we discuss the two-basic software division of computer.**

**4.3.1 SOFTWARE SPECIFICATION**

The following are the software requirements for this application

* NetBeans 7.2 Integrated Development Environment (IDE)
* Structured query language (SQL) server 2005
* Windows operating system (at least windows vista installed)
* Anti-virus package to prevent that application from virus attack

**4.3.2 HARDWARE SPECIFICATION**

The above listed software will work perfectly with the under listed specification, as a computer is not complete without either the software or the hardware

* 1 gigabyte RAM
* 1.5GH2 processor
* 200GB Hard Disk or higher (recommended)
* Uninterruptible Power Supply (UPS)
* Mouse and enhanced keyboard

**4.4 TESTING**

After the implementation, the following tests are going to take place:

**i Alpha Test:** This means self or in-house test for the application for any error or exceptions.

**ii Beta Test:** at this stage the virtual data for the application is going to be release for user to test the application.

**4.5 IMPLEMENTATION PHASE**

The implementation phase of this project is divided into two which are the NMF factorized result and the ABC (Artificial bee colony) summary. These phases are explained below:

**i.** The NMF factorization result is generated using a plug-in called Text mining language (TML), TML implementation analysis each paragraph, sentence and word. Nonnegative Matrix Factorization has been proved to be valuable in many fields of data mining, especially in unsupervised learning. The special point on NMF is its ability to recover the hidden patterns or trends behind the observed data automatically, which makes it suitable for image processing, feature extraction, dimensional reduction and unsupervised learning.

**ii** The ABC summary technique as ABC algorithm consists of three phases. They are initialization, employed, onlooker and scout bees phase. Each phase is replayed until the maximum count of iterations is reached. In the initial phase, the count of solutions and the control parameters are fixed. The employed bees phase deals with the search of new high quality food sources in the nearby locality of old food source. The new food source is then evaluated for its fitness, which is then followed by the comparison of the old and the new food source by means of greedy selection. The collected knowledge about the food source is distributed among the onlooker bees present in the beehive.

**4.6 SYSTEM DOCUMENTATION**

**Installation Procedure**

This program is already packaged to some extence having all requirement to function with the database and tml mining. some computer programs can be executed by simply copying them into a [folder](http://en.wikipedia.org/wiki/Folder_%28computing%29) stored on a computer and executing but this is quit advanced in nature because of the advancement in technology Other programs are supplied in a form not suitable for immediate execution and therefore need an installation procedure. Once installed, the program can be executed again and again, without the need to reinstall before each execution. But it is important to note that users must have an SQL server and TML package in the C: directory installed and running before execution as the program is based on local server. The following are the step involve in installing this application.

1. Install Java on your system by visiting oracle.com
2. Download and install TML ( Text mining language) from online
3. Move TML ( Text mining language) to the C dir
4. Copy the Free text summary to a dir on your system and launch the jar
5. You are set to go.

**4.6.1 System Maintenance**

The program may be maintained on the ground that the system requires an upgrade. Though it is compiled as standalone software the database can be tempered with but it’s advisable that the admin put a password on the file to secure the database from intrusion.

**The following precaution should be done**

* Ensure that the computer is kept in clean areas.
* System should be kept in cool places.
* Air conditioner is important to reduce room temperature and keep it constant.
* Backup of data is important

**4.6.2 SYSTEM EVALUATION**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S/N | ORIGINAL DOCUMENTS | NUMBER OF WORDS | SUMMARY DOCUMENTS | NUMBER OF WORDS | PERCENTAGE |
| 1 | With the dramatic growth of the Internet, people are overwhelmed by the tremendous amount of online information and documents. This expanding availability of documents has demanded exhaustive research in the area of automatic text summarization. Text document summarization is the process of reducing the size of documents while maintaining its relevant information.  The aim of automatic text summarization is to reduce the source text into a compact version which will preserve contents and general meaning. A summary is a concise form of text that is composed from one or more texts that gives important information in the original text (Mullen .D. S, 2012). Summary reduces the reading time. Text summarization can be classified into extractive summarization and abstractive summarization. Extractive summarization means selecting the important sentence from the original documents. Abstractive summarization means express the meaning of the document in natural language.  Although, technologies that can make a coherent summary must take into account variables such as length, writing style and syntax for any text document. | 165 | With the dramatic growth of the Internet, people are overwhelmed by the tremendous amount of online information and documents  The aim of automatic text summarization is to reduce the source text into a compact version which will preserve contents and general meaning  Although, technologies that can make a coherent summary must take into account variables such as length, writing style and syntax for any text document | 66 | 40% |
| 2 | The theory of optimization has its roots in the isoperimetric problem faced by Queen Dido in 1000 BC. She procured for the founding of Carthage the largest area of land that could be surrounded by the hide of a bull. From the hide she made a rope, which she arranged in a semicircle with the ends against the sea. Queen Dido’s intuitive solution was correct. But it was many centuries before a formal proof was presented, and the mathematical and systematic solution to this problem proved to be a very difficult problem in the calculus of variations. The calculus of variations essentially handles problems where the decision variable is a vector.  The calculus of variations, or the first systematic theory of optimization, was born on June 4, 1694, when John Bernoulli posed the Brachistochrone (Greek for “shortest time”) problem, and publicly challenged the mathematical world to solve it. The problem posed was, “What is a slide path down which a frictionless object would slip in the least possible time?” Earlier attempts to solve this problem were made by many well-known scientists including Galileo who proposed the solution to be a circular arc, an incorrect solution, and Leibnitz who presented ordinary differential equations without solving them. Then John Bernoulli proved the optimal path to be a cycloid. From that point, efforts continued in the area of the calculus of variations, leading to the study of multiple integrals, differential equations, control theory, problem transformation and so on. | 244 | \* The theory of optimization has its roots in the isoperimetric problem faced by Queen Dido in 1000 BC She procured for the founding of Carthage the largest area of land that could be surrounded by the hide of a bull  \* The calculus of variations, or the first systematic theory of optimization, was born on June 4, 1694, when John Bernoulli posed the Brachistochrone (Greek for “shortest time”) problem, and publicly challenged the mathematical world to solve it  \* The problem posed was, “What is a slide path down which a frictionless object would slip in the least possible time?” Earlier attempts to solve this problem were made by many well-known scientists including Galileo who proposed the solution to be a circular arc, an incorrect solution, and Leibnitz who presented ordinary differential equations without solving them | 136 | 55.74% |
| 3 | Non-negative Matrix Factorization has been proved to be valuable in many fields of data mining, especially in unsupervised learning. The special point on NMF is its ability to recover the hidden patterns or trends behind the observed data automatically, which makes it suitable for image processing, feature extraction, dimensional reduction and unsupervised learning.  Intuitively there are three ideas on disguising sensitive data. One is to transform original data into protected, publishable data by data perturbation. An alternative to data perturbation is to generate a new dataset (synthetic dataset), not from the original data, but from random values that are adjusted in order to have the same feature pattern as the original data. A third possibility is to build a hybrid dataset as a mixture of a distorted one and a synthetic one. The idea of positive matrix factorization is developed by P. Paatero at the University of Helsinki, and to be popular in the computational science community. Interest in positive matrix factorization increased when a fast algorithm for Non-negative Matrix Factorization (NNMF), based on iterative update, was developed by (Lee and Seung, 2010), particularly as they were able to show that it produced intuitively reasonable factorizations for a face recognition problem. NNMF has recently been shown to be very useful technique in approximating high dimensional data where the data are comprised of non-negative components. NNMF is a vector space method to obtain a representation of data using non-negative constraints. These constraints can lead to a parts-based representation because they allow only additive, not subtractive, combinations of the original data. This is in contrast to techniques for finding a reduced dimensional representation based on SVD. | 278 | \* Non-negative Matrix Factorization has been proved to be valuable in many fields of data mining, especially in unsupervised learning  \* The special point on NMF is its ability to recover the hidden patterns or trends behind the observed data automatically, which makes it suitable for image processing, feature extraction, dimensional reduction and unsupervised learning | 54 | 19.4% |
| 4 | NMF finds applications in such fields as computer vision, document clustering, chemometrics, audio signal processing and recommender systems. Many standard NMF algorithms analyze all the data together; i.e., the whole matrix is available from the start. This may be unsatisfactory in applications where there are too many data to fit into memory or where the data are provided in streaming fashion. One such use is for collaborative filtering in recommendation systems, where there may be many users and many items to recommend, and it would be inefficient to recalculate everything when one user or one item is added to the system. The cost function for optimization in these cases may or may not be the same as for standard NMF, but the algorithms need to be rather different.  A combination of Artificial Bee Colony and k-means algorithm is proposed for clustering the web documents. ABC colony algorithm is an efficient population based optimization algorithm and it imitates the behaviour of real bees. The k-means algorithm is efficient and fast, however the problem is on finding initial cluster point. This work proposes to locate the initial cluster point with the help of bees and these clusters are refined by the k-means algorithm. We propose to combine both ABC and k-means algorithm, so as to inherit the merits of both the algorithms. ABC is efficient but consumes more time for convergence. The k-means algorithm is also known for its faster convergence but struggles in locating the initial cluster point. Thus, a new algorithm is presented for improving the efficiency and reducing the execution time. The steps involved in the proposed algorithm are explained below | 278 | \* NMF finds applications in such fields as computer vision, document clustering, chemometrics, audio signal processing and recommender systems  \* One such use is for collaborative filtering in recommendation systems, where there may be many users and many items to recommend, and it would be inefficient to recalculate everything when one user or one item is added to the system | 58 | 20.86% |
| 5 | Interaction between insects contribute to their collective intelligence of the social insect colony which have been adapted to scientific problem optimization.one of the examples of such interactive behavior is the waggle dance of bees during food procuring.by this dance By performing this dance, successful foragers share the information about the direction and distance to patches of flower and the amount of nectar within this flower with their hive mates. So this is a successful mechanism which foragers can recruit other bees in their colony to productive locations to collect various resources.  Bee colony can quickly and precisely adjust its searching pattern in time and space according to changing nectar sources. The information exchange among individual insects is the most important part of the collective knowledge. Communication among bees about the quality of food sources is being achieved in the dancing area by performing waggle dance | 147 | Interaction between insects contribute to their collective intelligence of the social insect colony which have been adapted to scientific problem optimization  \* by this dance By performing this dance, successful foragers share the information about the direction and distance to patches of flower and the amount of nectar within this flower with their hive mates | 54 | 36.7% |

Based on the system evaluation stated above, it is discovered that the number of words in summary document is smaller compare to the number of words in the original document. That’s to say that the aim of this project has been achieved.

**CHAPTER FIVE**

**SUMMARY, CONCLUSION AND RECOMMENDATION**

**5.1 SUMMARY**

This chapter summarizes the research done in this thesis, together with directions for future work.

An hybrid algorithms for text summarization system using Non-negative matrix factorization technique and Artificial bee colony optimization is a system that summarizes an original text document by achieving an optimal dimension reduction of the source document , removing the stop words , clustering the text document in matrix form and summarizing the text document based on the weighted words. This application is an improvement to the previous techniques used in summarizing a text document. The approach of summarizing text manually and extracting information online has been in existence for ages. In the era of internet, online information is freely available for readers in the form of e-Newspapers, journal articles, technical reports, transcription dialogues etc. There is huge number of documents available in above digital media. Therefore it is very serious issue to get data fast and efficiently and extracting only relevant information from all these media is a tiresome for the individuals in stipulated time. There is a need for an automated text summarization that can extract only relevant information from the original document.

**5.2 CONCLUSION**

After integrating the two algorithms NMF (Non-negative matrix factorization) and ABC ( Artificial bee colony) techniques to meet up with the task of developing an automated text summarization application, following strict specification has been reached, we can then say that an automated text summarization application with standard capacity of summarizing an original document and displaying a summary output has been achieved. Following all the steps and processes involved in putting this together, it is now clear for us to see that the result of hybrid algorithm for text summarization using NMF (Non-negative matrix factorization) and ABC (Artificial bee colony) is more interpretable, fast in convergence and ability to achieve its aim within a short term period compare to other related techniques proposed in achieving same result of summarizing an original document.

**5.3 RECOMMENDATION**

The following recommendations are suggested in order to and efficiently fasten the summarization process on free text documents.

1. This hybrid text summarization application used to summarize free text document should not be limited to a student environment alone.
2. The application should be adopt by every assessment organization because it is considered accurate efficient.
3. This project can be enhanced for future works to authenticate in a dark place.

In future works, it could be also implemented on a special device.

**REFERENCES**

**Abiyot, B**. (2010). Developing automatic word parser for Amharic verbs and their derivation. Masteres thesis, Addis Ababa University, Addis Ababa.

**Adane, L**. (2011). Feature Extraction and Matching in Amharic Document Image Collections.

**Ager, S.** (2013). Omniglot - The Online Encyclopedia of Writing Systems and Languages Retrieved April 28,, 2013, from http://www.omniglot.com/

**Anbessa, T., & Hudson, G**. (2010). Essentials of Amharic (Vol. 18): Rüdiger Köppe.

**Asker, L., Atelach, A. A., Gambäck, B., & Sahlgren, M.** (2010). Applying Machine Learning to Amharic Text Classification. Paper presented at the Proceedings of the 5th World Congress of African Linguistics.

**Asker, L., Atelach, A. A., Gambäck, B., Samuel, E. A., & Lemma, N. H.** (2012). Classifying Amharic webnews. Information retrieval, 12(3), 416-435.

**Atelach, A. A.** (2016). Automatic Sentence Parsing for Amharic Text: An Experiment using Probabilistic Context Free Grammars. Master Thesis, Addis Ababa University.

**Atelach, A. A., & Asker, L.** (2012). An Amharic stemmer: Reducing words to their citation forms. Paper presented at the Proceedings of the 2012 workshop on computational approaches to semitic languages: Common issues and resources.

**Atelach, A. A., Asker, L., Cöster, R., & Karlgren, J**. (2013). Dictionary-based Amharic–English information retrieval Multilingual Information Access for Text, Speech and Images (pp. 14149): Springer.

**Basagic, R., Krupic, D., & Suzic, B**. (2012). Automatic Text Summarization. Graz University of Technology.

**Baye, Y.** (2010). Root Reduction and Extension in Amharic'. Paper presented at the Tenth Annual Conference of the Institute of Language Studies, Addis Ababa University.

**Bender, M. L**. (2013). Language in Ethiopia (illustrated ed.). London: Oxford University Press.

**Bender, M. L., & Hailu, F**. (2014). Amharic verb morphology: A generative approach: African Studies Center, Michigan State University.

**Bethlehem, M**. (2014). N-Gram-based Automatic Indexing for Amharic Text. Masters Thesis. Addis Ababa University: Addis Ababa. 110

**Boudin, F., Torres-Moreno, J.-M., & Velázquez-Morales, P.** (2010). An efficient statistical approach for automatic organic chemistry summarization *Advances in Natural Language Processing* (pp. 89-99): Springer.

**Devarajan, (**2009). An Integrated Approach to Automatic Complex Sentence Parsing for Amharic Text. Unpublished Master‟ s Thesis, School of Graduate Studies, Addis Ababa University.

**Ding, He, & Simon** (2005). A Survey on Automatic Text Summarization.

**Dom Lachowicz, M. G., & Rotem, N.** (2010). Open Text Summarizer, version 0.4. 2, libots group.

**Gohr, A**. (2013). Web interface for Open Text Summarizer, April 11, 2013, from http://www.splitbrain.org/services/ots

**Gupta, V., & Lehal, G. S**. (2010). A survey of text summarization extractive techniques. JournalofEmergingTechnologies *in* WebIntelligence*, 2*(3), 258-268.

**Hahn, U., & Mani, I**. (2015). The challenges of automatic summarization. *Computer, 33*(11), 29-36.

**Hassel, M.** (2012). Evaluation of Automatic Text Summarization. Licentiatethesis*,* SchoolofComputer Science and Communication*,* Royal Institute of Technology *(*KTH*),* published.

**Hassen, R., Tessema, M., & Solomon, A**. (2013). SearchEngine for Amharic Web Content*.* Paper presented at the IEEE AFRICON, Nairobi, Kenya.

**Helen, A**. (2014). AutomaticText Summrizer for Amharic Legal Judgment*.* Addis Ababa University, School of Graduate Studies,, Addis Ababa.

**Hovy, E., & Marcus, D**. (2014). Automated text summarization. The Oxford Handbook of computational linguistics, 583-598.

**Kamil, N. (**2012). Automatic Amharic News Text Summarizer. Master's Thesis, Addis Ababa University, Faculity of Informatics, Department Information Science, Addis Ababa.

**Karel, J., & Josef, S**. (2012). Automatic Text Summarization (The State of the art 2012 and new challenges).

**Lee & Seung (**1999**;** 2001). Automatic Summarizing Factors and Directions Advances in Automatic Text Summarization. Cambridge MA: MIT Press, (2016), 675-685.

**Lin, C.-Y., & Hovy, E**. (2012). Automated multi-document summarization in neats. Paper presented at the Proceedings of the second international conference on Human Language Technology Research.

**Lloret, E.** (2010). Text Summarization: An Overview: Dept. Lenguajes y Sistemas Informaticos Universidad de Alicante Alicante, Spain.

**Luhn, H. P.** (2015). The automatic creation of literature abstracts. IBM Journal of research and development.

**Martha, Y. T**. (2010). Morphology-based language modeling for Amharic. Doctoral Thesis, University of Hamburg, Hamburg, Germany.

**Massih, R. A., Nicolas, U., & Gallinari, P.** (2016). Automatic Text Summarization Based on Word - Clusters and Ranking Algorithms.

**Melese, T**. (2011). Automatic Amharic Text Summarization using Latent Semantic Analysis. Master Thesis, Addis Ababa University.

**Mersehazen, W**. (2017). Yäamargna Sawasaw. Addis Ababa: Birhanena Selam Printing Press.

**Mesfin, G.** (2015). Automatic part of speech tagging for Amharic language: An experiment using stochastic HMM. Master Thesis, Addis Ababa University.

**Mullen, D. S.** (2012). Issues in the morphology and phonology of Amharic: The lexical generation of pronominal clitics: University of Ottawa.

**Nega, A., & Willett, P.** (2013). Stemming of Amharic words for information retrieval. Literary and Linguistic computing.

**APPENDIX A**

**NMFALGORITHM**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package NMF;

importjava.util.Iterator;

importtml.corpus.CorpusParameters;

importtml.vectorspace.TermWeighting.GlobalWeight;

importtml.vectorspace.TermWeighting.LocalWeight;

importtml.corpus.SearchResultsCorpus;

importtml.storage.Repository;

importtml.utils.Stats;

importtml.vectorspace.operations.PassagesSimilarity;

/\*\*

\*

\*

\*

\* @author Holus

\*/

public class NMF {

Variables var;

public NMF() throws Exception {

var = new Variables();

// trainData("data");

}

/\*\*

\* @paramargs the command line arguments

\*/

public String trainData(String ans) throws Exception {

Repository repository = new Repository("C:/tml-3.2/tml/corpora/myrepo");

repository.cleanStorage("C:/tml-3.2/tml/corpora/myrepo");

repository.addDocumentsInFolder(ans);

var.setCollection(testData());

System.out.println(testData());

returntestData();

// TODO code application logic here

}

public String testData() throws Exception {

Repository repository = new Repository("C:/tml-3.2/tml/corpora/myrepo");

SearchResultsCorpus corpus = new SearchResultsCorpus("type:document");

corpus.getParameters().setTermSelectionThreshold(0);

corpus.getParameters().setDimensionalityReduction(CorpusParameters.DimensionalityReduction.NO);

corpus.getParameters().setDimensionalityReductionThreshold(50);

corpus.getParameters().setTermWeightGlobal(GlobalWeight.None);

corpus.getParameters().setTermWeightLocal(LocalWeight.TF);

corpus.load(repository);

// System.out.println("Corpus Loaded and Semantic space Calculated");

// System.out.println("Total documents: "+corpus.getPassages().length);

System.out.println(corpus.parametersSummary());

PassagesSimilarity distances = new PassagesSimilarity();

distances.setCorpus(corpus);

distances.start();

String a = "";

for (String str : corpus.getPassages()) {

a += str + "\n";

}

var.setAnswer(a);

var.setWordFreq(corpus.printFrequencies());

a = "U\n";

for (double dob[] : corpus.getTermDocMatrix().svd().getU().getArray()) {

for (double dob1 : dob) {

a += dob1 + " ";

}

a += "\n";

}

var.setU(a);

a = "V\n";

for (double dob[] : corpus.getTermDocMatrix().svd().getV().transpose().getArray()) {

for (double dob1 : dob) {

a += dob1 + " ";

}

a += "\n";

}

// return null;

returnvar.getAnswer() + "\n" + "\n" + var.getWordFreq() + "\n" + var.getU() + "\n" + var.getA() + "\n" + var.getRel() + "\n" + corpus.parametersSummary() + "\n" + "\n";

}

// public String rep() throws Exception {

// Repository repository = new Repository("C:/tml-3.2/tml/corpora/myrepo");

// SearchResultsCorpus corpus = new SearchResultsCorpus("type:document");

// corpus.load(repository);

// String a = "";

// for (String str :corpus.getPassages()) {

// a += str + "\n";

// }

// var.setAnswer(a);

// return a;

// }

public static void main(String[] args) throws Exception {

new NMF();

}

}

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package ABC;

import NMF.\*;

importjava.util.Iterator;

importtml.corpus.CorpusParameters;

importtml.vectorspace.TermWeighting.GlobalWeight;

importtml.vectorspace.TermWeighting.LocalWeight;

importtml.corpus.SearchResultsCorpus;

importtml.storage.Repository;

importtml.utils.Stats;

importtml.vectorspace.operations.PassagesSimilarity;

/\*\*

\*

\*

\*

\* @author Holus

\*/

public class ABC {

Variables var;

public ABC() throws Exception {

var = new Variables();

// trainData("data");

}

/\*\*

\* @paramargs the command line arguments

\*/

public String trainData(String ans) throws Exception {

Repository repository = new Repository("C:/tml-3.2/tml/corpora/myrepo");

repository.cleanStorage("C:/tml-3.2/tml/corpora/myrepo");

repository.addDocumentsInFolder(ans);

var.setCollection(testData());

System.out.println(testData());

returntestData();

// TODO code application logic here

}

public String testData() throws Exception {

Repository repository = new Repository("C:/tml-3.2/tml/corpora/myrepo");

SearchResultsCorpus corpus = new SearchResultsCorpus("type:document");

corpus.getParameters().setTermSelectionThreshold(0);

corpus.getParameters().setDimensionalityReduction(CorpusParameters.DimensionalityReduction.NO);

corpus.getParameters().setDimensionalityReductionThreshold(50);

corpus.getParameters().setTermWeightGlobal(GlobalWeight.None);

corpus.getParameters().setTermWeightLocal(LocalWeight.TF);

corpus.load(repository);

// System.out.println("Corpus Loaded and Semantic space Calculated");

// System.out.println("Total documents: "+corpus.getPassages().length);

System.out.println(corpus.parametersSummary());

PassagesSimilarity distances = new PassagesSimilarity();

distances.setCorpus(corpus);

distances.start();

String a = "";

for (String str : corpus.getPassages()) {

a += str + "\n";

}

var.setAnswer(a);

var.setWordFreq(corpus.printFrequencies());

a = "U\n";

for (double dob[] : corpus.getTermDocMatrix().svd().getU().getArray()) {

for (double dob1 : dob) {

a += dob1 + " ";

}

a += "\n";

}

var.setU(a);

a = "V\n";

for (double dob[] : corpus.getTermDocMatrix().svd().getV().transpose().getArray()) {

for (double dob1 : dob) {

a += dob1 + " ";

}

a += "\n";

}

var.setV(a);

a = "S\n";

for (double dob[] : corpus.getTermDocMatrix().svd().getS().transpose().getArray()) {

for (double dob1 : dob) {

a += dob1 + " ";

}

a += "\n";

}

var.setE(a);

a = "A\n";

for (double dob[] : corpus.getTermDocMatrix().getArray()) {

for (double dob1 : dob) {

a += dob1 + " ";

}

a += "\n";

}

var.setA(a);

a = "R\n" + distances.getResultsCSVString();

String a1[] = a.split("\n");

String y = "";

for (String string : a1) {

if (string.startsWith("lecturer")) {

y += string + "\n";

}

}

var.setRel("R\n" + y);

//distances.printResults();

inti = 1;

System.out.println(corpus.getName());

for (String string : corpus.getTerms()) {

System.out.println(i++ + string);

}

i = 1;

for (Stats string : corpus.getDocStats()) {

System.out.println(i++ + string.toString());

}

i = 1;

for (Stats string : corpus.getTermStats()) {

System.out.println(corpus.getTerms()[i-1]);

System.out.println(i++ + string.toString());

}

for (String string[] : distances.getResultsStringTable()) {

System.out.println();

for (String string1 : string) {

}

}

i = 1;

System.out.println("===>" + corpus.parametersSummary());

// return null;

returnvar.getAnswer() + "\n"+ "\n" + var.getWordFreq() + "\n" + "\n" + corpus.parametersSummary() + "\n" + "\n";

}

// public String rep() throws Exception {

// Repository repository = new Repository("C:/tml-3.2/tml/corpora/myrepo");

// SearchResultsCorpus corpus = new SearchResultsCorpus("type:document");

// corpus.load(repository);

// String a = "";

// for (String str :corpus.getPassages()) {

// a += str + "\n";

// }

// var.setAnswer(a);

// return a;

// }

**ABC ALGORITHM**

public static void main(String[] args) throws Exception {

new ABC();

}

}

packagedocsum.ui;

importdocsum.summarizer.DocumentSummarizer;

importdocsum.summarizer.KeywordExtractor;

importdocsum.summarizer.SentencePreprocessor;

importdocsum.summarizer.SentenceSegmenter;

import NMF.NMF;

/\*\*

\* Graphical user interface for summarizer program.

\*

\* @author Evan Dempsey

\*/

public class GraphicalInterface extends JFrame {

final Charset ENCODING = StandardCharsets.UTF\_8;

private static final long serialVersionUID = 6253527329314698074L;

DocumentSummarizer summarizer;

KeywordExtractor extractor;

JPanel panel;

JTextAreasourceTextArea;

JTextAreakeywordTextArea;

JTextAreasummaryTextArea;

JLabelpercentLabel;

JSliderpercentSlider;

JLabelsourceCharsLabel;

JLabelsourceWordsLabel;

JLabelsourceLinesLabel;

JLabelsummaryCharsLabel;

JLabelsummaryWordsLabel;

JLabelsummaryLinesLabel;

JRadioButton jRadioButton1;

JRadioButton jRadioButton2;

ButtonGroup buttonGroup1;

NMF lsa;

/\*\*

\* Constructor.

\*

\* @param summarizer DocumentSummarizer instance.

\* @param extractor KeywordExtractor instance.

\*/

publicGraphicalInterface(DocumentSummarizer summarizer,

KeywordExtractor extractor) {

this.summarizer = summarizer;

this.extractor = extractor;

initUI();

try {

lsa = new NMF();

} catch (Exception ex) {

Logger.getLogger(GraphicalInterface.class.getName()).log(Level.SEVERE, null, ex);

}

}

/\*\*

\* Sets up the user interface.

\*/

public void initUI() {

// Set up the menu bar

JMenuBarmenuBar = new JMenuBar();

// Set up the file menu

JMenufileMenu = new JMenu("File");

JMenuItemopenMenuItem = new JMenuItem("Open");

openMenuItem.setToolTipText("Open a text document.");

openMenuItem.addActionListener(new OpenActionListener());

JMenuItemsaveMenuItem = new JMenuItem("Save");

saveMenuItem.setToolTipText("Save the summarys.");

saveMenuItem.addActionListener(new SaveActionListener());

JMenuItemexitMenuItem = new JMenuItem("Exit");

exitMenuItem.setToolTipText("Exit application.");

exitMenuItem.addActionListener(new QuitActionListener());

fileMenu.add(openMenuItem);

fileMenu.add(saveMenuItem);

fileMenu.add(exitMenuItem);

// Set up the edit menu

JMenueditMenu = new JMenu("Edit");

JMenuItemcutMenuItem = new JMenuItem(new DefaultEditorKit.CutAction());

cutMenuItem.setText("Cut");

cutMenuItem.setToolTipText("Cut the current selection.");

JMenuItemcopyMenuItem = new JMenuItem(new DefaultEditorKit.CopyAction());

copyMenuItem.setText("Copy");

copyMenuItem.setToolTipText("Copy the current selection.");

JMenuItempasteMenuItem = new JMenuItem(new DefaultEditorKit.PasteAction());

pasteMenuItem.setText("Paste");

pasteMenuItem.setToolTipText("Paste the contents of the clipboard.");

editMenu.add(cutMenuItem);

editMenu.add(copyMenuItem)

// Make a JScrollPane for the summary text and put a JTextArea in it.

// jSlider1.setFont(new java.awt.Font("Century Gothic", 1, 12)); // NOI18N

// jSlider1.setMajorTickSpacing(10);

// jSlider1.setMinorTickSpacing(5);

// jSlider1.setPaintLabels(true);

// jSlider1.setPaintTicks(true);

// jSlider1.setSnapToTicks(true);

// jSlider1.setToolTipText("range");

// jSlider1.setValue(0);

// jSlider1.setBorder(javax.swing.BorderFactory.createTitledBorder("Range of Summarization"));

// Set up the percentage TextField

percentLabel = new JLabel("50%");

percentLabel.setText("50%");

//Create Radio Button

jRadioButton1 = new JRadioButton("NMF Summarization");

jRadioButton1.setFont(new java.awt.Font("Century Gothic", 1, 12)); // NOI18N

jRadioButton1.setBounds(450, 510, 150, 25);

jRadioButton2 = new JRadioButton("NMF ABC Summarization");

jRadioButton2.setFont(new java.awt.Font("Century Gothic", 1, 12)); // NOI18N

jRadioButton2.setBounds(600, 510, 190, 25);

buttonGroup1 = new ButtonGroup();

buttonGroup1.add(jRadioButton1);

buttonGroup1.add(jRadioButton2);

// Create the summarize button

JButtonsummarizeButton = new JButton("Summarize");

summarizeButton.setBounds(50, 60, 80, 30);

summarizeButton.setToolTipText("Summarize the document.");

summarizeButton.addActionListener(new SummarizeActionListener());

// Set up the bottom JPanel

JPanelbottomPanel = new JPanel();

bottomPanel.setLayout(new BoxLayout(bottomPanel, BoxLayout.X\_AXIS));

bottomPanel.setBorder(BorderFactory.createEmptyBorder(20, 0, 0, 0));

bottomPanel.add(Box.createHorizontalGlue());

// bottomPanel.add(jRadioButton1);

bottomPanel.add(jRadioButton2);

bottomPanel.add(summarizeButton);

// Set up the main JPanel

panel = new JPanel();

panel.setLayout(new BorderLayout());

panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

// Add widgets to main panel

panel.add(centerPanel, BorderLayout.CENTER);

panel.add(bottomPanel, BorderLayout.SOUTH);

setJMenuBar(menuBar);

add(panel);

setTitle("Document Summarizer");

setSize(800, 600);

setLocationRelativeTo(null);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

}

/\*\*

\* Reads a specified file and return its contents as a string

\*

\* @param file File object.

\* @return String with file contents.

\*/

public String readFile(File file) {

StringBufferfileBuffer = null;

String fileString = null;

String line = null;

try {

FileReaderfileReader = new FileReader(file);

BufferedReaderbufferedReader = new BufferedReader(fileReader);

fileBuffer = new StringBuffer();

while ((line = bufferedReader.readLine()) != null) {

fileBuffer.append(line).append(

System.getProperty("line.separator"));

}

fileReader.close();

fileString = fileBuffer.toString();

} catch (IOException e) {

return null;

}

returnfileString;

}

/\*\*

\* Change listener: updates percentage figure in text field in response to

\* slider changes.

\*

\* @author Evan Dempsey

\*/

public class SliderChangeListener implements ChangeListener {

public void stateChanged(ChangeEvent e) {

JSlider source = (JSlider) e.getSource();

intval = source.getValue();

percentLabel.setText(Integer.toString(val) + "%");

}

}

// Action listeners: executed in response to

// graphical user interface events.

/\*\*

\* Takes text from the sourceTextArea and the percentage from the

\* percentSlider and generates a summary and keyword list. Puts the summary

\* into the summaryTextArea and the keyword list into the keywordTextArea.

\*

\* @author Evan Dempsey

\*/

public class SummarizeActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

try {

if (sourceTextArea.getText().isEmpty()) {

JOptionPane.showMessageDialog(rootPane, "Nothing to summarize");

return;

}

int percentage = percentSlider.getValue();

intlc = Integer.parseInt(sourceLinesLabel.getText().split(": ")[1]);

intlin = (int) ((percentage / 100.0) \* lc);

System.out.println(lin);

String summary = summarizer.summarize(sourceTextArea.getText(), percentage);

Summarizer sm = new Summarizer(sourceTextArea.getText(), lin);

String sumup = sm.Summarize(sourceTextArea.getText(), lin);

// String keywords = extractor.extract(sumup.trim());

writeFile("data/document.txt", sourceTextArea.getText());

String okdata = lsa.trainData("data");

if (jRadioButton1.isSelected()) {

String keywords = extractor.extract(summary.trim());

// summaryTextArea.setText(okdata);

summaryTextArea.setText(summary);

keywordTextArea.setText(okdata);

// keywordTextArea.setText(keywords);

} else if (jRadioButton2.isSelected()) {

String keywords = extractor.extract(sumup.trim());

summaryTextArea.setText(sumup);

// summaryTextArea.append(sumup);

keywordTextArea.setText(okdata);

// keywordTextArea.setText(keywords);

// sumup = sm.Summarize(sumup, lin);

// keywords = sumup;

// keywords = stripeSpecialChar(keywords);

// keywords = StripRepeatedWord(keywords);

// sumup = sm.Summarize(sumup, lin);

// keywords = sm.coolected;

} else {

JOptionPane.showConfirmDialog(rootPane, "Please Select and algorithm");

}

} catch (Exception ex) {

Logger.getLogger(GraphicalInterface.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

/\*\*

\* Listens for the Open menu item, gets a file from the file chooser dialog,

\* reads its contents, and puts it in the sourceTextArea.

\*

\* @author Evan Dempsey

\*/

public class OpenActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

JFileChooserfileOpen = new JFileChooser();

FileNameExtensionFilter filter = new FileNameExtensionFilter("Text Files", "txt");

fileOpen.setFileFilter(filter);

intreturnValue = fileOpen.showDialog(panel, "Open File");

if (returnValue == JFileChooser.APPROVE\_OPTION) {

// Read the file

File file = fileOpen.getSelectedFile();

String text = readFile(file);

// Put the file contents into the text area

sourceTextArea.setText(text);

}

}

}

/\*\*

\* Listens for the Save menu item and saves the summary.

\*

\* @author Evan Dempsey

\*/

public class SaveActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

JFileChooserfileSave = new JFileChooser();

intreturnValue = fileSave.showDialog(panel, "Save Summary");

if (returnValue == JFileChooser.APPROVE\_OPTION) {

File file = fileSave.getSelectedFile();

try {

FileWriterfileWriter = new FileWriter(file);

fileWriter.write(summaryTextArea.getText());

fileWriter.close();

} catch (IOException ex) {

ex.printStackTrace();

}

}

}

}

/\*\*

\* Listens for the Exit menu item and exits the application.

\*

\* @author Evan Dempsey

\*/

public class QuitActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

System.exit(0);

}

}

/\*\*

\* Listens for the About menu item and displays dialog.

\*

\* @author Evan Dempsey

\*/

public class AboutActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

// Make information string.

StringBuilder info = new StringBuilder();

info.append("<html><body style='width: 200px; text-align: center'>");

info.append("Document Summarizer<br><br>");

info.append("Automatic document summarization program by Evan Dempsey.<br><br>");

info.append("Penn Treebank tokenizer provided by Stanford NLP toolkit.");

info.append("</body></html>");

String infoString = info.toString();

// CreateJOptionPane with no icon and custom title.

JOptionPane.showMessageDialog(panel,

infoString,

"About Document Summarizer",

JOptionPane.PLAIN\_MESSAGE);

}

}

/\*\*

\* Listens for changes in documents, recalculates document statistics and

\* updates stats widgets.

\*

\* @author Evan Dempsey

\*/

public class TextChangeListener implements DocumentListener {

public void changedUpdate(DocumentEvent e) {

update

}

private String StripRepeatedWord(String n1) {

for (String str : n1.split(" ")) {

n1 = n1.replaceAll(str, "") + str + " ";

}

return n1.trim();

}

floattestCorpus(String aFileName, String wordA, String wordB) {

Path path = Paths.get(aFileName);

String eachLine = "";

float count = 0;

try {

Scanner scanner = new Scanner(path, ENCODING.name());

while (scanner.hasNextLine()) {

eachLine = scanner.nextLine() + ":";

if (eachLine.contains(wordA + ":") &&eachLine.contains(wordB + ":")) {

//corpusList += eachLine + "\n";

count++;

}

}

} catch (Exception e) {

e.printStackTrace();

}

return count;

}

String stripeSpecialChar(String stripeStr) {

String specialChar = "\*.,!?;:%(){}[]\n";

for (char str : specialChar.toCharArray()) {

if (stripeStr.contains(str + "")) {

// punt += str + " ";

stripeStr = stripeStr.replace(str + "", "");

}

}

returnstripeStr;

}

private String compileCompound(String aFileName, String n1) throws IOException {

Path path = Paths.get(aFileName);

String eachLine = "";

String WordN1[] = n1.split(" ");

String compoundList = "";

try {

Scanner scanner = new Scanner(path, ENCODING.name());

while (scanner.hasNextLine()) {

eachLine = scanner.nextLine();

for (inti = 0; i< n1.split(" ").length - 1; i++) {

if (eachLine.equalsIgnoreCase(WordN1[i] + WordN1[i + 1])) {

n1 = n1.replaceAll(WordN1[i] + " " + WordN1[i + 1], eachLine);

compoundList += eachLine + "\t";

}

}

}

} catch (Exception e) {

e.printStackTrace();

}

return n1;

}

voidwriteFile(String aFileName, String text) throws IOException {

try {

PrintWriter writer = new PrintWriter(aFileName, "UTF-8");

writer.println(text);

writer.close();

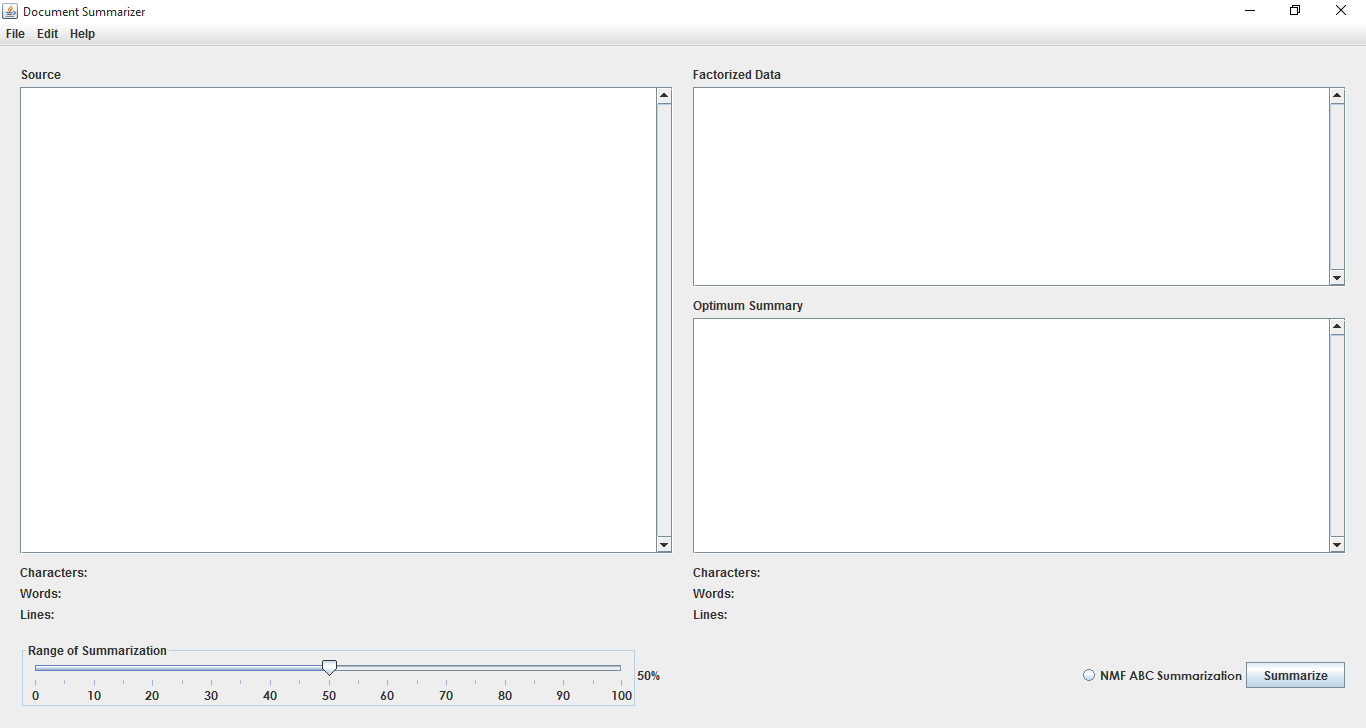
} catch (Exception e) {

e.printStackTrace();

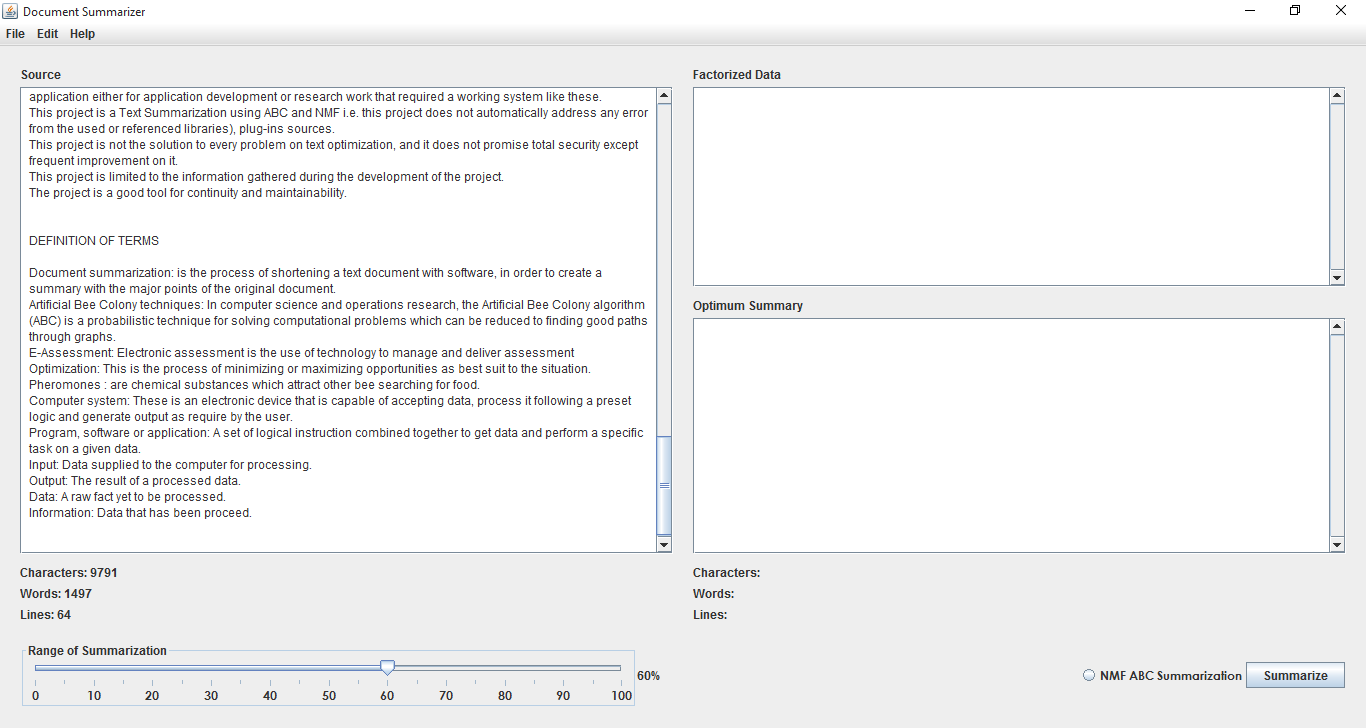
}

**APPENDIX B**

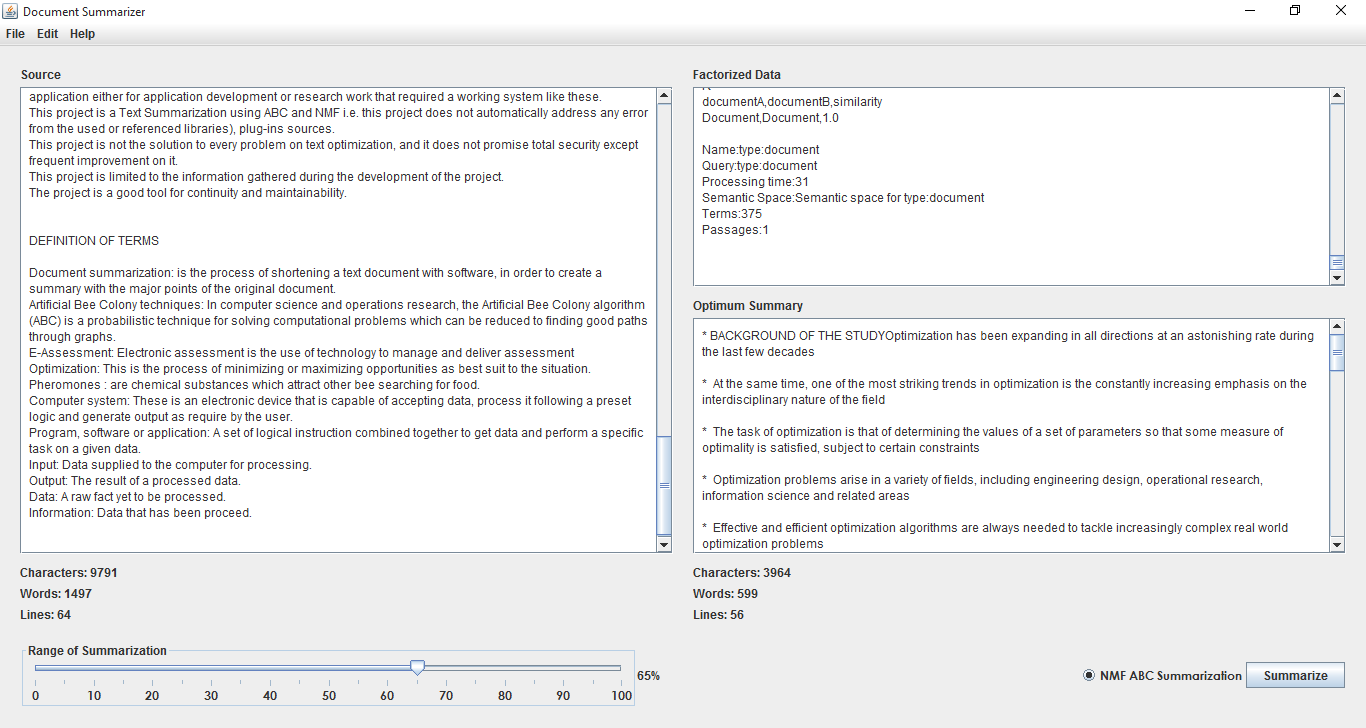
**The Initial Interface**



**The Input Interface**



**The Output Interface**



**APPENDIX C**

**RESULT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S/N | ORIGINAL DOCUMENTS | NUMBER OF WORDS | SUMMARY DOCUMENTS | NUMBER OF WORDS |
| 1 | With the dramatic growth of the Internet, people are overwhelmed by the tremendous amount of online information and documents. This expanding availability of documents has demanded exhaustive research in the area of automatic text summarization. Text document summarization is the process of reducing the size of documents while maintaining its relevant information.  The aim of automatic text summarization is to reduce the source text into a compact version which will preserve contents and general meaning. A summary is a concise form of text that is composed from one or more texts that gives important information in the original text (Mullen .D. S, 2012). Summary reduces the reading time. Text summarization can be classified into extractive summarization and abstractive summarization. Extractive summarization means selecting the important sentence from the original documents. Abstractive summarization means express the meaning of the document in natural language.  Although, technologies that can make a coherent summary must take into account variables such as length, writing style and syntax for any text document. | 165 | With the dramatic growth of the Internet, people are overwhelmed by the tremendous amount of online information and documents  The aim of automatic text summarization is to reduce the source text into a compact version which will preserve contents and general meaning  Although, technologies that can make a coherent summary must take into account variables such as length, writing style and syntax for any text document | 66 |
| 2 | The theory of optimization has its roots in the isoperimetric problem faced by Queen Dido in 1000 BC. She procured for the founding of Carthage the largest area of land that could be surrounded by the hide of a bull. From the hide she made a rope, which she arranged in a semicircle with the ends against the sea. Queen Dido’s intuitive solution was correct. But it was many centuries before a formal proof was presented, and the mathematical and systematic solution to this problem proved to be a very difficult problem in the calculus of variations. The calculus of variations essentially handles problems where the decision variable is a vector.  The calculus of variations, or the first systematic theory of optimization, was born on June 4, 1694, when John Bernoulli posed the Brachistochrone (Greek for “shortest time”) problem, and publicly challenged the mathematical world to solve it. The problem posed was, “What is a slide path down which a frictionless object would slip in the least possible time?” Earlier attempts to solve this problem were made by many well-known scientists including Galileo who proposed the solution to be a circular arc, an incorrect solution, and Leibnitz who presented ordinary differential equations without solving them. Then John Bernoulli proved the optimal path to be a cycloid. From that point, efforts continued in the area of the calculus of variations, leading to the study of multiple integrals, differential equations, control theory, problem transformation and so on. | 244 | \* The theory of optimization has its roots in the isoperimetric problem faced by Queen Dido in 1000 BC She procured for the founding of Carthage the largest area of land that could be surrounded by the hide of a bull  \* The calculus of variations, or the first systematic theory of optimization, was born on June 4, 1694, when John Bernoulli posed the Brachistochrone (Greek for “shortest time”) problem, and publicly challenged the mathematical world to solve it  \* The problem posed was, “What is a slide path down which a frictionless object would slip in the least possible time?” Earlier attempts to solve this problem were made by many well-known scientists including Galileo who proposed the solution to be a circular arc, an incorrect solution, and Leibnitz who presented ordinary differential equations without solving them | 136 |
| 3 | Non-negative Matrix Factorization has been proved to be valuable in many fields of data mining, especially in unsupervised learning. The special point on NMF is its ability to recover the hidden patterns or trends behind the observed data automatically, which makes it suitable for image processing, feature extraction, dimensional reduction and unsupervised learning.  Intuitively there are three ideas on disguising sensitive data. One is to transform original data into protected, publishable data by data perturbation. An alternative to data perturbation is to generate a new dataset (synthetic dataset), not from the original data, but from random values that are adjusted in order to have the same feature pattern as the original data. A third possibility is to build a hybrid dataset as a mixture of a distorted one and a synthetic one. The idea of positive matrix factorization is developed by P. Paatero at the University of Helsinki, and to be popular in the computational science community. Interest in positive matrix factorization increased when a fast algorithm for Non-negative Matrix Factorization (NNMF), based on iterative update, was developed by (Lee and Seung, 2010), particularly as they were able to show that it produced intuitively reasonable factorizations for a face recognition problem. NNMF has recently been shown to be very useful technique in approximating high dimensional data where the data are comprised of non-negative components. NNMF is a vector space method to obtain a representation of data using non-negative constraints. These constraints can lead to a parts-based representation because they allow only additive, not subtractive, combinations of the original data. This is in contrast to techniques for finding a reduced dimensional representation based on SVD. | 278 | \* Non-negative Matrix Factorization has been proved to be valuable in many fields of data mining, especially in unsupervised learning  \* The special point on NMF is its ability to recover the hidden patterns or trends behind the observed data automatically, which makes it suitable for image processing, feature extraction, dimensional reduction and unsupervised learning | 54 |
| 4 | NMF finds applications in such fields as computer vision, document clustering, chemometrics, audio signal processing and recommender systems. Many standard NMF algorithms analyze all the data together; i.e., the whole matrix is available from the start. This may be unsatisfactory in applications where there are too many data to fit into memory or where the data are provided in streaming fashion. One such use is for collaborative filtering in recommendation systems, where there may be many users and many items to recommend, and it would be inefficient to recalculate everything when one user or one item is added to the system. The cost function for optimization in these cases may or may not be the same as for standard NMF, but the algorithms need to be rather different.  A combination of Artificial Bee Colony and k-means algorithm is proposed for clustering the web documents. ABC colony algorithm is an efficient population based optimization algorithm and it imitates the behaviour of real bees. The k-means algorithm is efficient and fast, however the problem is on finding initial cluster point. This work proposes to locate the initial cluster point with the help of bees and these clusters are refined by the k-means algorithm. We propose to combine both ABC and k-means algorithm, so as to inherit the merits of both the algorithms. ABC is efficient but consumes more time for convergence. The k-means algorithm is also known for its faster convergence but struggles in locating the initial cluster point. Thus, a new algorithm is presented for improving the efficiency and reducing the execution time. The steps involved in the proposed algorithm are explained below | 278 | \* NMF finds applications in such fields as computer vision, document clustering, chemometrics, audio signal processing and recommender systems  \* One such use is for collaborative filtering in recommendation systems, where there may be many users and many items to recommend, and it would be inefficient to recalculate everything when one user or one item is added to the system | 58 |
| 5 | Interaction between insects contribute to their collective intelligence of the social insect colony which have been adapted to scientific problem optimization.one of the examples of such interactive behavior is the waggle dance of bees during food procuring.by this dance By performing this dance, successful foragers share the information about the direction and distance to patches of flower and the amount of nectar within this flower with their hive mates. So this is a successful mechanism which foragers can recruit other bees in their colony to productive locations to collect various resources.  Bee colony can quickly and precisely adjust its searching pattern in time and space according to changing nectar sources. The information exchange among individual insects is the most important part of the collective knowledge. Communication among bees about the quality of food sources is being achieved in the dancing area by performing waggle dance | 147 | Interaction between insects contribute to their collective intelligence of the social insect colony which have been adapted to scientific problem optimization  \* by this dance By performing this dance, successful foragers share the information about the direction and distance to patches of flower and the amount of nectar within this flower with their hive mates | 54 |