Extractive Text Summarization using Deep Learning

Nikhil S. Shirwandkar
Student, Mtech Electronics Engineering
K. J. Somaiya College of Engineering, Vidyavihar
Mumbai, India
nikhilshirwandkar@gmail.com

Dr. Samidha Kulkarni
Associate Professor, Dept. of Electronics Engineering
K. J. Somaiya College of Engineering, Vidyavihar
Mumbai, India
samidhakulkarni@somaiya.edu

***Abstract*— An approach for generating short and precise summaries for long text documents is proposed. Lately, the size of information on the internet is increasing. It has become tough for the users to dig into the loads of information to analyze it and draw conclusions. Text summarization solves this problem by generating a summary, selecting sentences which are most important from the document without losing the information. In this work, an approach for Extractive text summarization is designed and implemented for single document summarization. It uses a combination of Restricted Boltzmann Machine and Fuzzy Logic to select important sentences from the text still keeping the summary meaningful and lossless. The text documents used for summarization are in English language. Various sentence and word level features are used to provide meaningful sentences. Two summaries for each document are generated using Restricted Boltzmann Machine and Fuzzy logic. Both summaries are then combined and processed using a set of operations to get the final summary of the document. The results show that the designed approach overcomes the problem of text overloading by generating an effective summary.**

Keywords*— *Deep Learning, Extractive, Fuzzy logic, RBM, Sentence Features, Single document, Summarization, Unsupervised.

I. INTRODUCTION

Earlier, humans used to summarize the text by their own, but today due to increasing data, it is difficult for the human beings to cope up with the huge data. To overcome this issue, text summarization is needed. Text summarization generates succinct form of large documents keeping most of the original info. It is carried out on single or multiple documents of similar kind. Based on the nature of summary, it is classified as Extractive, wherein the generated summary consists of the most important sentences of the document put together and Abstractive, wherein new sentences are formed from the scratch to produce the summary. Text summarization has increased the attention of the researchers since 1950. H. P. Luhn was the first to make an automatic text summarization system. It was grounded on the frequency of terms [1]. In 1958, he stated significance of words based of their frequency measures. According to him, words which have a medium frequency i.e. which neither occur much frequently nor less frequently are important. He also removed the stop-words from the text. In 1958, Baxendale proposed salient features based on the sentence position [2]. He stated that the beginning 7% of the document and the last sentences have most of the information. In 1969, Edmundson suggested new method of automatic text summarization which included two fresh features in addition to the position and term which were pragmatic words (cue words), title and heading words [3]. In 1995, Kupiec, Pedersen, and Chen developed Naïve Bayes classifier which classified the sentences of the document to add in the summary [4]. They used sentence scoring features like Sentence length cutoff, fixed-phrase, paragraph feature, thematic words, and uppercase words to decide the probability of the sentence. In 1995, SUMMONS (Summarizing online news articles) a natural language based summarizer was developed by McKeown and Radev [5]. This was the first multi-document summarization system. In 1997, Inderjeet Mani and Eric Bloedorn developed a graph based method in which the text was represented in the form of graphs [6]. Spreading activation technique was used to find semantic nodes which are related. It was used to identify the similarities and differences between the documents. In 2001, Conroy and O'leary proposed a Hidden Markov Model (HMM) for text summarization [7]. Features like position, sentence terms and their respective frequencies were incorporated for HMM development. In 2004, Radev, Jing, Stys, and Tam developed a centroid based multi document text summarizer named MEAD [8]. Term Frequency- Inverse Document Frequency (TF-IDF) feature was used for centroid based summarization. MEAD was used to summarize clusters of news articles. In 2005, Evans and Klavans proposed a new method for summarizing the clusters of documents on the same events based on text similarity [9]. English and Arabic language documents were used. In 2015, S. A. Babar and P. D. Patil proposed an approach to improve the performance of text summarization using Singular value Decomposition and Fuzzy inference system [10]. In 2016, S.P Singh, A. Kumar, A. Mangal, S. Singhal proposed a method for bilingual text summarization using Restricted Boltzmann Machine (RBM) [11]. Eleven sentence scoring features were used and given to RBM for feature enhancement. In 2017, an approach for auto text summarization was developed by H. A. Chopade and M. Narvekar using Deep network and Fuzzy logic which provided significant increase in the accuracy of the summary [12].

This work is focused on extractive text summarization. Combining some of the significant works an approach is designed which uses RBM as an unsupervised deep learning algorithm and Fuzzy logic, along with some sentence features to improve the connectivity and the relevance in the sentences.

A. Organisation of the paper

In this paper, section I gives an introduction to the topic and its literature survey, section II describes the proposed method for extractive text summarization, section III produces experiments

978-1-5386-5257-2/18/$31.00 ©2018 IEEE

and results, and section IV states conclusions and the future scope.

II. Proposed Method

A. Problem Statement

Due to increasing text data on the web, the time required for the users to summarize and analyze the huge data is increased. The solution to reduce reading time of the user is producing a succinct document summary.

B. Method

Fig.1 shows the proposed method for Extractive text summarization using Deep Learning:

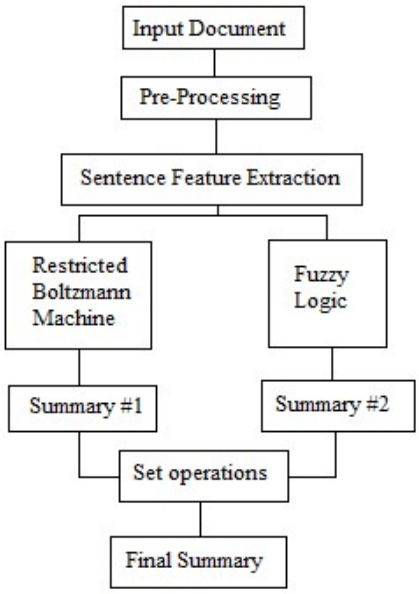

Fig. 1 Proposed Method

1) Input Document: The first step of the process for generating the summary is to input a text document with format as .txt. Text Documents used in this work are in English language. The text files are imported using tkinter python library.

2) Pre-Processing: The imported text is processed as shown in Fig. 2.

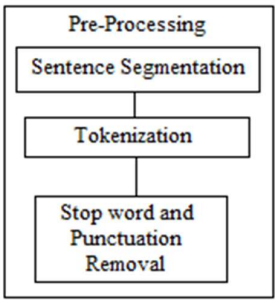

Fig. 2 Pre-Processing

In Sentence Segmentation, the whole text is broken down into sentences and is stored in an array with their respective sentence positions. In Tokenization, the sentences obtained are further broken down into words for some feature calculations. In Stop word and punctuation removal, the commonly occurring words such as the, an, a, but, and, or etc. are removed along with all the punctuation.

All the pre-processing steps are carried out using a library in python called as Natural Language Toolkit (NLTK) for Natural Language Processing.

3) Sentence Feature Extraction: After Pre-processing the text, sentence features are calculated to find the sentence score. Following sentence features contribute in deciding the sentence score:

- Sentence Position: On the basis of sentence location, its relevance is known. In [2] author states that the first and the last sentence of the document is always important and has maximum information. Position feature is calculated using (1).

$$\text{Sent. Pos.} = \begin{cases} 1, & \text{if first or last sentence} \\ \frac{N-P}{N}, & \text{if others} \end{cases} \quad (1)$$

where, N is the total sentences, P is the location of the sentence.

- Sentence Length: According to the author, in [2] very short sentences do not contain much information. To find the important sentence based on its length, the feature score is calculated using (2).

$$\text{Sent. Len}_{\,i} = \frac{\text{words in sentence}_i}{\text{words in largest sentence}} \quad (2)$$

- Numerical Token: Numerical token is the total numerical values in a sentence. It is calculated using (3).

$$\text{Numerical token Score}_i = \frac{\text{num\_numeric}_i}{\text{len}} \quad (3)$$

where, num_numeric_i is the numerical tokens in i^{th} sentence and len is the total words in i^{th} sentence.

- TF-ISF: Term frequency - Inverse document frequency (TF-IDF) is necessary for systems retrieving information. In this work, text summarization is carried out for a single document. So, Term frequency - Inverse Sentence Frequency (TF-ISF) is calculated using (4) [11].

$$\text{TFISF score} = \frac{(\log(\text{isf}) * (\text{tf}))}{\text{len}} \quad (4)$$

Where, isf is the total occurrences of the each term of i^{th} sentence in all other sentences, tf is term frequency of each term in i^{th} sentence and len is total words in i^{th} sentence.

- Cosine similarity between Sentence and Centroid: Centroid is the sentence with largest TF-ISF [11]. Cosine similarity with the centroid is calculated for each sentence as given in (5).

$$\text{Cos}_{\text{sim}_i} = \cos(\text{sentence}_i, \text{centroid}) = \frac{\text{sentence}_i \cdot \text{centroid}}{||\text{sentence}_i|| \, ||\text{centroid}||} \quad (5)$$

- Bi-Gram: Bi-grams are pair of two adjacent words formed for each sentence in the document. NLTK library is used for finding the total no. of bi-grams in a sentence. The feature score is normalized to restrict the feature value between 0 and 1.

- Tri-Gram: Tri-grams are a triple of three adjacent words formed each sentence in the document. NLTK library is used for finding the total no. of tri-grams in a sentence. The feature score is normalized to restrict the value between 0 and 1.

- Proper Noun: Proper noun is a noun which refers to a unique identity like name, place, etc. In this feature, the total no. of proper nouns in each sentence is calculated using (6) [10]. To find the proper nouns, the sentences are POS tagged using NLTK.

$$\text{Proper noun score}_i = \frac{\text{No.of proper nouns}_i}{\text{Sentence length}_i} \quad (6)$$

  where, No. of proper nouns$_i$ is the no. of proper nouns in ith sentence and sentence length$_i$ is the no. of total words in ith sentence.

- Thematic Words: Thematic words are keywords in each sentence which are domain specific. The no. of thematic words in a sentence is calculated using (7) [10].

$$\text{Thematic word}_i = \frac{\text{No.of thematic words}_i}{\text{Total no.of thematic words}} \quad (7)$$

  where, No. of thematic words$_i$ is the no. of thematic words in ith sentence.

After calculating all the sentence features, a sentence feature matrix is formed. In this work, each sentence has nine feature values. The number of feature values is variable based on the number of features used.

4) Restricted Boltzmann Machine (RBM): RBM is a neural network with random probability distributions. The network contains a visible layer of visible neurons (input nodes) and hidden layers of hidden neurons (hidden nodes). Every input node has a bi-directional connection with every hidden node. The bias node has a connection with every hidden node. The input nodes are not interconnected in the visible layer. Also the hidden nodes are not interconnected in the hidden layers. Because of these restricted connections, the name of the network is Restricted Boltzmann machine. Fig.3 shows the RBM Network architecture.

   After the formation of the sentence matrix, it is normalized by dividing each of its elements with the highest element and is given as an input to the RBM to enhance the values for obtaining the sentence score. In this work, RBM consists of three layers: One Visible layer, and two Hidden layers.

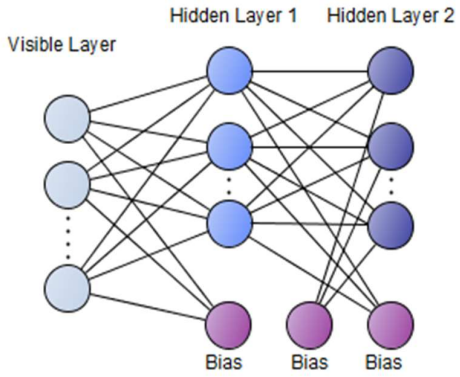

Fig. 3 RBM architecture

Visible layer contains nine nodes corresponding to nine features.

As shown in Fig.4, in the Forward pass, features of every sentence are multiplied by random weights when going through Hidden layer 1.

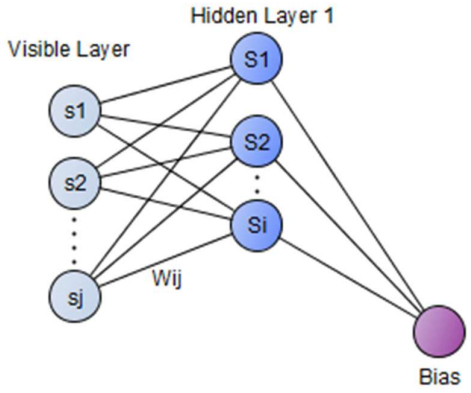

Fig. 4 RBM: Forward Pass- Visible layer to Hidden layer 1

A randomly generated bias is then added to the product. The probability of activation of hidden unit S_i given visible unit s_j is given in (8).

$$p(S_i|s_j) = \sigma(\sum_{j=1}^{n} s_j \times w_{ij} + b_i) \quad (8)$$

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (9)$$

where, $\sigma(x)$ is the sigmoid function given in (9), w_{ij} are weights which are randomly generated and b_i is the bias.

The process is same for hidden layer 2 where in hidden layer 1 output is input for hidden layer 2 as shown in Fig. 5.

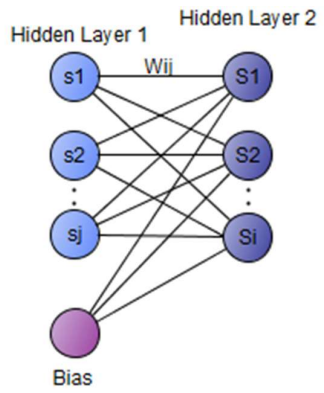

Fig. 5 RBM: Forward Pass- Hidden layer 1 to Hidden layer 2

In the backward pass, the output value is multiplied by the same weights and added to a bias as done in the forward pass and output is obtained at the visible layer as shown in Fig. 6.

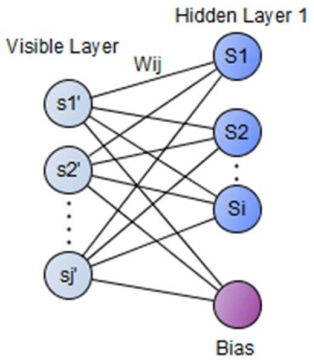

Fig. 6 RBM: Backward Pass- Reconstruction

The probability of activation of visible unit s_j given hidden unit S_i is given in (10).

$$p(s_j|S_i) = \sigma(\sum_{i=1}^{m} S_i \times w_{ij} + b_j) \quad (10)$$

where, $\sigma(x)$ is the sigmoid function given in (9), w_{ij} are weights and b_i is the bias.

Thus, while training, the values of hidden units are predicted, and after knowing the values of hidden units, the new values of input are predicted. This process is called Gibbs sampling. Difference between the input values s_j and the new input values s_j' is calculated which states the training loss. This process is carried out for certain epochs and the weights are learned by Contrastive Divergence given by (11).

$$w_{ij\,new} = w_{ij\,old} + (LR \times \Delta w) \quad (11)$$

$$\Delta w = (s_j \otimes p(S_i|s_j) - s_j' \otimes p(S_i'|s_j')) \quad (12)$$

where, LR is the learning rate, Δw is change in weights given in (12) as difference of outer products of probabilities with original input values s_j and new input values s_j'. An enhanced feature matrix is obtained after the epochs which has enhanced feature values for each sentence.

5) Summary #1: Fig. 7 shows the process of generating the first summary. The sum of all enhanced feature values for each sentence in the document is calculated and stored in a list.

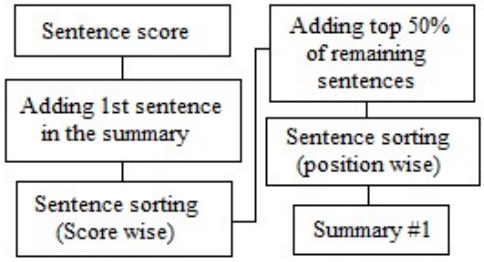

Fig. 7 Generating Summary #1

Thus, for each sentence one value is generated which is the score of that sentence. On the basis of scores, sentences are arranged in descending manner. Summary always includes the first sentence as it is the most important sentence and then top 50% of the remaining sentence based on their descending scores are added in the first summary and sorted according to their original position in the document.

6) Fuzzy Logic: The feature scores calculated earlier are converted into percentage. Triangular membership functions are used to fuzzify each score into three levels HIGH, MEDIUM and LOW as shown in Fig. 8. IF-THEN fuzzy rules are then applied for de-fuzzification to determine whether the sentence is Important, Average or Unimportant e.g. IF (Feature1 is HIGH, Feature2 is HIGH, Feature3 is MEDIUM, Feature4 is MEDIUM), THEN (Sentence is Important). Python library used for fuzzy logic system is skfuzz.

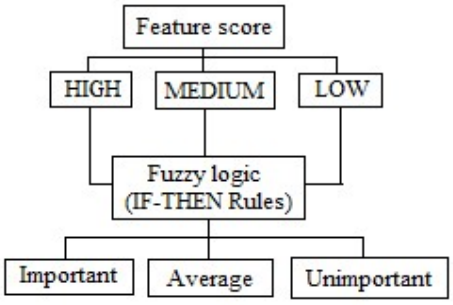

Fig. 8 Fuzzy logic system

7) Summary #2: After getting the computation results from the fuzzy logic system, the sentences which fit in "Important" category are added in the second summary according to their original position in the document.

8) Set operations for Final Summary: Procedure for generating the final summary is shown in Fig. 9. From the first and second summary, a common and uncommon set of sentences are found. Inclusion of common set is made in the Final summary.

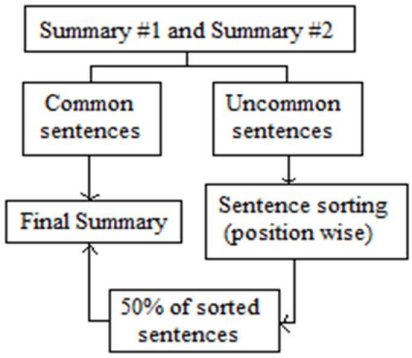

Fig. 9 Generating final summary

The uncommon sentences are sorted position wise and half part of uncommon set of sentences are added in final summary. After adding, the sentences are arranged according to their original position in the text document.

III. Experimentations and Results

Experimentations are performed using Intel Core i3 2.0GHz processor, 4GB RAM. The programming language used is Python 3.6 with Jupyter Notebook as a platform.

In the proposed method, nine feature values are calculated for a sentence to get better relevance. RBM is trained for fifteen epochs for generating the first summary of the document. Fuzzy logic IF-THEN rules are used for generating the second summary of the same document and then using set operations a final summary is obtained.

The proposed method for Extractive text summarization is implemented for single document along with the original method which uses RBM only. The generated summaries from both the methods are compared. Recall Oriented Understudy for Gisting Evaluation (ROUGE) is used for evaluating generated summaries. The parameters for performance evaluation are Precision, Recall and F measure. The values are found out for ten English documents from the News articles dataset from Kaggle using both the methods. Average values with both the methods are shown in TABLE 1.

TABLE 1. COMPARISON RESULTS OF TEXT SUMMARIZATION

	RBM method			Proposed method		
Doc	***Precision***	***Recall***	***F measure***	***Precision***	***Recall***	***F measure***
1	0.78	0.7	0.74	0.8	0.88	0.84
2	0.85	0.76	0.80	0.86	0.75	0.80
3	0.75	0.7	0.72	0.87	0.77	0.82
4	0.82	0.78	0.80	0.91	0.79	0.85
5	0.84	0.8	0.82	0.89	0.80	0.84
6	0.77	0.76	0.76	0.92	0.86	0.89
7	0.87	0.83	0.85	0.88	0.78	0.83
8	0.79	0.76	0.77	0.89	0.72	0.80
9	0.85	0.82	0.83	0.87	0.84	0.85
10	0.83	0.79	0.80	0.86	0.81	0.83
Avg	**0.82**	**0.77**	**0.79**	**0.88**	**0.80**	**0.84**

IV. Conclusions and Future Scope

In this work, RBM is used as an unsupervised learning algorithm along with fuzzy logic for improving the accuracy of the summary. It is observed that the proposed approach generates short and precise summaries without any irrelevant text. Using features like Sentence-Centroid similarity and thematic words has improved the connectivity of the sentences. Using the proposed method, on an average 88% precision, 80% recall and 84% F measure is obtained. The results produced using the proposed method give better evaluation parameters in comparison with prevailing RBM method.

The proposed method can be extended for multi document summarization. Documents in different languages can be summarized. Various other features can be used and the method can be combined with other methods for improving the nature of the summary. Also it can be used in Abstractive text summarization.

References

[1] H. P. Luhn, "The automatic creation of literature abstracts," IBM Journal of Research Development, vol. 2, no. 2, pp. 159-165, 1958.

[2] P. Baxendale, "Machine-made index for technical literature - an experiment." IBM Journal of Research Development, vol. 2, no. 4, pp. 354-361, 1958.

[3] H. P. Edmundson, "New methods in automatic extracting," Journal of the ACM, vol. 16, no. 2, pp. 264-285, 1969.

[4] J. Kupiec, J. Pedersen, and F. Chen, "A trainable document summarizer," In Proceedings SIGIR '95, pp. 68-73, New York, NY, USA, 1995

[5] K. R. McKeown, and D. R. Radev, "Generating summaries of multiple news articles," In Proceedings of SIGIR '95, pp. 74-82, Seattle, Washington, 1995.

[6] Inderjeet Mani and Eric Bloedorn, "Multi-document summarization by graph search and matching," AAAI/IAAI, vol. cmplg/ 9712004, pp. 622-628, 1997.

[7] J. M. Conroy, and D. P. O'leary, "Text summarization via hidden markov models," In Proceedings of SIGIR '01, pp. 406-407, New York, NY, USA, 2001.

[8] D. R. Radev, H. Jing, M. Stys, and D. Tam, "Centroid-based summarization of multiple documents," Information Processing and Management, vol. 40 , pp. 919-938, 2004.

[9] D. K. Evans, "Similarity-based multilingual multidocument summarization," Technical Report CUCS-014- 05, Columbia University, 2005.

[10] S. A. Babar, and P. D. Patil, "Improving performance of text summarization," Procedia Computer Science, vol. 46, pp. 354-363, 2015.

[11] S. P. Singh, A. Kumar, A. Mangal, and S Singhal, "Bilingual Automatic Text Summarization Using Unsupervised Deep Learning." International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)-2016, pp. 1195-1200, 2016.

[12] H. A. Chopade and M. Narvekar, "Hybrid auto text summarization using deep neural network and fuzzy logic system," 2017 International Conference on Inventive Computing and Informatics (ICICI), COIMBATORE, India, pp. 52-56, 2017.