Extractive Text Summarization Using Feature-Based Unsupervised RBM Method

Grishma Sharma, Subhashini Gupta, and Deepak Sharma

Abstract A methodology for creating shorter and meaningful summaries for single documents is provided. With a lot of content to be had on the web, it's far simply no longer possible to go through each information source in complete detail. Consequently, a great mechanism is needed to extract relevant information. To overcome these challenges, information in the form of text is summarized with the objective to get relevant knowledge without loss of any information. A methodology for extractive text summarization for single-document summary is devised and developed in this work. It uses a restricted Boltzmann machine to choose essential phrases from the text. The text documents used for summarization are in the English language. Various aspects are used to generate meaningful phrases, and the restricted Boltzmann machine is being utilized to enrich and abstract those features to improve the consequent accuracy without sacrificing any significant information. The sentences are scored, and an extracted summary is created based on those enhanced features. The result indicates that the presented methodology tackles the problem of text overload by producing an appropriate summary. The result of RBM has been compared with the Text Rank, Lex Rank, LSA, and Luhn algorithm. The experimentation is carried out, and the summary is generated for eight different document sets and the result is evaluated using the ROUGE-1 score.

Keywords Extractive text summarization · RBM · Feature extraction

G. Sharma (✉) · S. Gupta · D. Sharma
Department of Computer Science, K. J. Somaiya College of Engineering, Mumbai, India
e-mail: neelammotwani@somaiya.edu

S. Gupta
e-mail: subhashini.g@somaiya.edu

D. Sharma
e-mail: deepaksharma@somaiya.edu

D. P. Agrawal et al. (eds.), *Cyber Security, Privacy and Networking*, Lecture Notes in Networks and Systems 370, https://doi.org/10.1007/978-981-16-8664-1_10

1 Introduction

The problem of summarizing text is very trivial. In this fast-paced life, everyone wants shorthand information to save time. People read the news by only reading the headlines and the first few lines. A popular app that is exploiting this nature of humans is "Inshorts", which provides a news summary in just 60 words. Students want a summary of class notes just one night before the exams. They also want a summary of YouTube lectures that may be hours long. NLP has proven to be very useful to solve this problem. While extractive summarization is popular currently, abstractive summarization is picking up pace. With the ever-rising amount of data in the globe, the demand for automatic summary generation from the text document is growing considerably to reduce the manual work of a person. In today's world, data generation and consumption are exploding at an exponential rate. Text summarization finds its applications in various NLP-related tasks such as question answering, text classification, and other related fields. Summaries are generated as an intermediary step in these systems, which help to reduce the length of documents. This, in turn, leads to faster access for information searching. News summarization and headline generation is another important application. Most of the search engines use machine-generated headlines for displaying news articles in feeds.

The focus of this research is on extractive text summarization that aims to implement a single-document summarizer system that achieves a quick, concise extractive summary of any textual document. The structure of paper is as follows: Sect. 1 is introduction, Sect. 2 presents literature survey, the proposed methodology is given in Sect. 3, the results and discussion are in Sect. 4, and finally, the conclusion and future aspects are in Sect. 5.

2 Literature Survey

Madhuri and Ganesh Kumar [1], a unique statistical methodology is proposed and proven for extractive text summarization on a single document. The approach is described to extract sentences of the input text in a concise fashion. Weights are assigned to sentences to rank them. From the input text, highly ranked sentences are extracted. They created a summarizer application that accepts text files as input. After that, the input file is pre-processing. The system then calculates the sum of weighted frequencies by dividing the frequency of the keywords by their greatest frequency. Finally, the summarizer extracts high-weighted-frequency sentences and converts them to audio.

Krishnaveni and Balasundaram [2] proposed strategy for improving the summary coherence, based on local scoring and ranking. The author creates two types of summaries in this paper: heading-wise and main. Features are usually employed for the score of phrases. The original text is separated heading by heading, and each heading is treated as a separate text.

Naik and Gaonkar [3], the rule-based concept was proposed to extract features from sentences. The author pre-processed the input data first and then retrieved keywords from the document, which were subsequently pruned based on the computed threshold. Then the feature value is calculated for each document. Certain rules have been written by the author. Finally, the sentence is sorted based on sentence score to form an extractive summary.

Jafari and Shahabi [4], presented a fuzzy logic method to calculate and encode feature values as feature attribute vectors for each text. The fuzzy system assigns a score between 0 and 1. Input values for membership function are calculated. The knowledge base also contains the rules needed for summarization. Finally, top n scored sentences are selected to form a summary.

One of the most well-known extractive text summarizing algorithms is stated by Luhn [5], and it is determined by the number of times words appear in the text, as well as the distance between relevant words, which is determined by the number of non-relevant words among relevant ones.

The text rank algorithm is generally based on the graph algorithm which is developed by Mihalcea and Tarau [6]. The text rank algorithm works in two steps, very first it calculates the similarity between two sentences, and in the second step, it calculates the overall significance of sentences.

3 Proposed Methodology

3.1 Data Pre-processing

1. *Sentence Segmentation*: Segmentation breaks down the entire text into sentences, once the sentences are broken then each sentence with their respective position will be stored in the array.
2. *Tokenization*: This data pre-processing step sentences are divided into tokens so that they can be further used for feature extraction tasks.
3. *Stop word and punctuation*: In this data pre-processing task, we will remove punctuation as well as often occurring words such as punctuation, but, the, a, and so on. Proposed methodology for extractive text summarization is shown in Fig. 1.

3.2 Feature Extraction

1. *Sentence Position*: The position of the sentence can determine the importance of the sentence for the summary. The sentences at the beginning and end of the document are usually the most significant. So, based on this the sentence score is

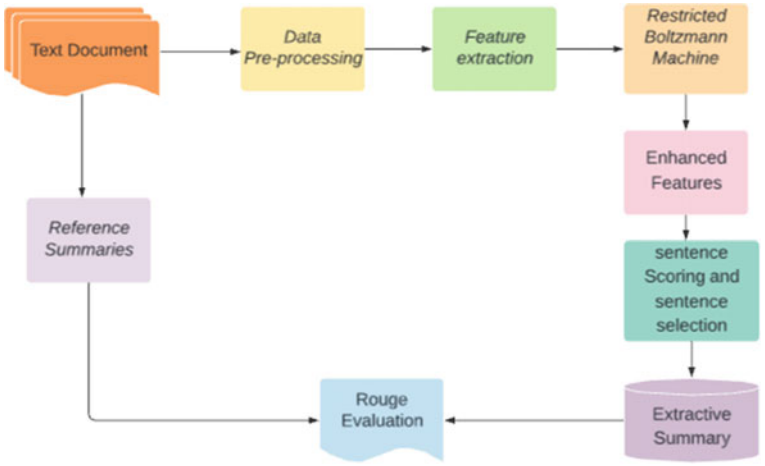


Fig. 1 Proposed methodology

calculated. In our case, the positional score is determined by taking into account the following factors:

$$\text{Sentence\_Position} = \begin{cases} 0, \text{ if it's first or last sentence of text} \\ \text{else cosine}(\text{Sen\_pos} - \min)((1/\max) - \min) \end{cases}$$

where,
Sen_pos: Position of sentence in text. Min: th * total no. of sentence in the text. Max: th*2* total no. of sentence in the text. Th: threshold which to be considered as 0.2.

2. *Tokenization*: This feature helps to filter out too short sentences, which typically don't convey much information.

$$\text{Sentence\_Length} = \begin{cases} 0, \text{if number of words is less than 3} \\ \text{else number of words in the sentence} \end{cases}$$

3. *Proper nouns*: A proper noun refers to something with a particular identity, such as a name or a location. In this first sentences are tagged using POS tagger using NLTK Library. Then we count the proper noun in tagged sentence.

$$\text{proper\_noun} = \begin{cases} \dfrac{\text{numberofpropernounintagged } S_i}{\text{totallengthoftagged } S_i} \end{cases}$$

where
s_i: ith sentence. Number of proper noun in tagged s_i: number of proper noun in ith sentence. Total length of s_i: number of words in ith sentence.

4. *Number of numerals*: Since the numerals in a document represent facts, having sentences with specific figures is important. The number of numerical can be calculated by using the following formula.

$$\text{number\_of\_numericals} = \left\{ \frac{\text{numberofnumericalsin } S_i}{\text{totalnumberofwordsin } S_i} \right.$$

5. *No. of Thematic words*: Thematic terms are the top ten most commonly used words in the sentence. The no. of thematic terms can be calculated as follows.

$$\text{thematic\_words} = \frac{\text{numberofthematicwordsin } S_i}{\text{totalnumberofthematicwords}}$$

6. *Centroid similarity*: The centroid sentence is defined as the sentence with the maximum TF-ISF score. The cosine similarity of each sentence to the centroid sentence will then be computed.

$$\text{Centroid\_Similarity} = \{\cos ine\_\text{similarity(centroid, sentence)}$$

7. *TF-ISF*: TF-ISF represents Term frequency—Inverse document frequency that's work similar to TF-IDF works. TF_ISF is given as follow:

$$\text{TF\_ISF} = \left\{ \frac{\sum \log(\text{isf}) \times \text{tf}}{\text{Totalwords}} \right.$$

8. *Named entities*: In this feature extraction section, we calculate the number of the named entity in every sentence. Sentences include references to named individuals such as a corporation, a group of people, and so on are often required to understand a factual report in some way.

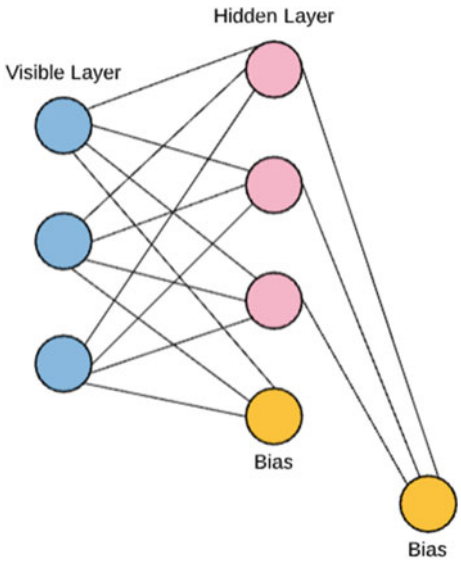
Once the feature value of each sentence is obtained, we will generate the sentence feature matrix of sentence. Sentence feature Matrix is a two-dimensional matrix as shown in Fig. 2. Sentence feature matrix $S = (s_1, s_2, s_3, s_4, \ldots, s_n)$where $s_i = (f_1, f_2, f_3, f_4, f_5, f_6,$
$f_7, f_8)$ is a sentence feature and $(i <= n)$. Where total no. of sentences in the document is indicated by n.

The sentence feature matrix produced by the preceding stages is as follows:

Fig. 2 Feature matrix for text summarization

$$\begin{array}{c} \\ s_1 \\ s_2 \\ . \\ . \\ s_n \end{array} \begin{bmatrix} f1 & f2 & f3 & f4 & f5 & f6 & f7 & f8 \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \end{bmatrix}$$

Fig. 3 RBM model



3.3 Restricted Boltzmann Machine

Our approach of extractive summarization is restricted Boltzmann machine (RBM), a stochastic and generative neural network that is capable of learning internal representations through probability distribution over its set of inputs [7]. They are a two-layered artificial neural network; the first layer is called the visible layer or the input layer (input nodes), and another one is called the hidden layer (hidden nodes). Every hidden node is connected to the bias node. The input nodes are not related to one another in the visible layer. Also, in the hidden layer, hidden nodes are not related to one another. The network is known as the restricted Boltzmann machine because of these restricted connections (Fig. 3).

To improve, the feature matrix S is fed into an RBM with one hidden layer. Each sentence passes through hidden layer 1 initially. Each sentence's feature values are multiplied by randomly produced weights, and one bias value is randomly produced and added to all the sentences. The results of these operation are fed into an activation function, which produces the nodes output (Fig. 4).

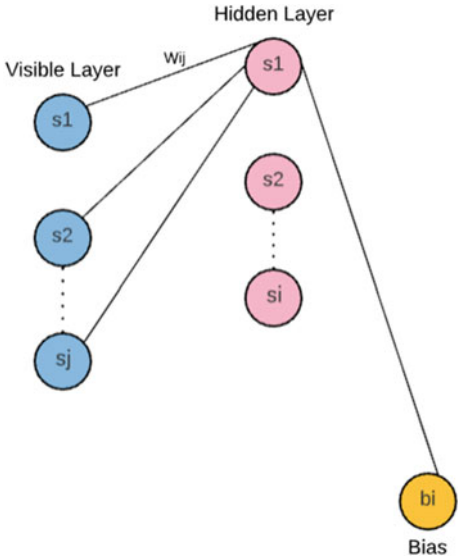


Fig. 4 Visible layer to Hidden layer

$$P(s_i) = \sigma\left(\sum_{j=1}^{n} s_j \times W_{ij} + b_i\right)$$

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

where, $\sigma(x)$ is the sigmoid function, w_{ij} stands for randomly generated weights, whereas bi stands for bias.

After this, the restricted Boltzmann machine learns to reconstruct data by itself in an unsupervised manner. This is done by reversing the above process, i.e., the hidden layer will become the input layer with activations as the new input. Then these activations are again multiplied with previous weights which are associated with the visible layer nodes and these products are added to the visible layer bias at each visible node. Therefore, obtained results are known as the reconstructions which are then compared to the original input (Fig. 5).

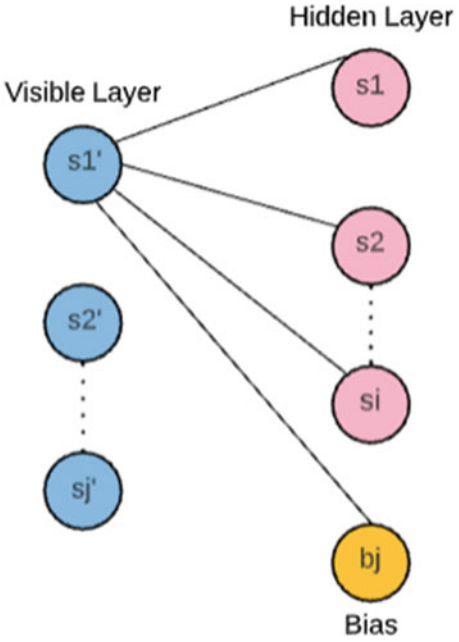


Fig. 5 Hidden layer to visible layer

The likelihood of activation of a visible unit s_j is stated as

$$P(s_j) = \sigma\left(\sum_{j=1}^{n} s_j \times W_{ij} + b_i\right)$$

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

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

Thus, the hidden unit values are predicted during training and after determining the hidden unit values, the new input values are predicted. This procedure is known as Gibbs sampling. The training loss is obtained by calculating the difference between the old s'_j and new input values s_j. This procedure is performed for a number of epochs, and the weights are determined using (CD) contrastive divergence given by:

$$w_{ij\,(\text{new})} = w_{ij\,(\text{old})} + (\text{Learning\_Rate} \times w^{'})$$
$$w^{'} = (s_j \otimes P(s_i) - s^{'}_j)$$

where, $w^{'}$ is the difference between the outer products of probabilities with the original input values s_j and the new input values as a change in weights values $s^{'}_j$. An enhanced feature matrix is obtained after the epochs which have enhanced feature values for each sentence.

3.4 Summary Generation

A list is created that contains the sum of all enhanced feature values for each sentence in the document. As a result, a value is generated for each sentence, which is the sentence's score. Sentences are sorted according to their scores in decreasing order. The first sentence is always included in the summary because it is the most essential sentence. The top 50% of the remaining sentences are included in the first summary and sorted in descending order according to their original location in the text.

4 Result and Discussion

We have used the BCC news dataset which includes several articles from different domains such as technology, politics, business, entertainment, and sport along with human-generated summaries for those articles. We have used ten documents from the BCC new dataset for evaluation purpose. Proposed methodology is compared with other state of art extractive summarization algorithm such as Text Rank [6], Lex Rank [8], LSA [9], Luhn [5]. For generating summary from Text Rank, Lex Rank, LSA, Luhn we have used Sumy Tool. By using Sumy tool, we can directly import this entire algorithm and for creating summary by using these algorithm users have just pass the text document. Each document is summarized with four different summarization techniques. We have extracted the top-ranked sentence from the document to form a summary at 10%, 20%, 30%, 40%, and 50% of document length. The purpose of experimenting with different percentage levels is to investigate the performance of various approaches.

We evaluated each system summary in conjunction with its corresponding reference summary. The ROUGE tool was used for the evaluation. ROUGE includes precision, recall and f-measure. The result fshows that proposed approach give better result than all other techniques that stated above (Tables 1, 2, and 3).

Table 1 Precision

	Summary length (%)				
Technique	10%	20%	30%	40%	50%
Text Rank	0.54	0.54	0.48	0.47	0.49
Lex rank	0.60	0.50	0.58	0.48	0.41
LSA	0.54	0.63	0.58	0.47	0.42
Luhn	0.75	0.62	0.55	0.51	0.44
RBM	0.87	0.74	0.70	0.52	0.54

Table 2 Recall

	Summary length (%)				
Technique	10%	20%	30%	40%	50%
Text Rank	0.27	0.44	0.46	0.60	0.75
Lex Rank	0.19	0.38	0.52	0.62	0.73
LSA	0.25	0.43	0.50	0.61	0.63
Luhn	0.25	0.35	0.41	0.52	0.56
RBM	0.30	0.45	0.56	0.66	0.82

Table 3 F1 Score

	Summary length (%)				
Techniques	10%	20%	30%	40%	50%
Text rank	0.36	0.48	0.47	0.51	0.66
Lex rank	0.33	0.52	0.50	0.40	0.59
LSA	0.34	0.54	0.58	0.53	0.68
Luhn	0.38	0.45	0.47	0.52	0.49
RBM	0.40	0.56	0.62	0.58	0.75

5 Conclusion

We have developed an unsupervised methodology that makes use of RBM to summarize single document. The algorithm works separately for each input document, as each document is uniquely different in itself. This is an advantage that the proposed methodology gives. Our method uses RBM and generates an effective and efficient summary. We have evaluated our model using the ROUGE1 score compared with other techniques, and the result shows that our model gives better results than other compared techniques.

The proposed methodology could be extended to multiple-document summarization. Different languages can be used to summarize documents.

References

1. Madhuri JN, Ganesh Kumar R (2019) Extractive text summarization using sentence ranking. In: IEEE international conference on data science and communication, Mar 2019
2. Naik SS, Gaonkar MN (2017) Extractive text summarization by feature-based sentence extraction using rule-based concept. In: IEEE international conference on recent trends in electronics information and communication technology (RTEICT), May 2017
3. Krishnaveni P, Balasundaram SR (2016) Automatic text summarization by local scoring and ranking for improving coherence. In: IEEE international international conference on automation and computing, Sept 2016

4. Jafari M, Shahabi AS, Wang J, Qin Y (2017) Automatic text summarization using fuzzy inference. In: Proceedings 22nd international conference on automation and computing, May 2017
5. Luhn HP (1958) The automatic creation of literature abstracts. IBM J Res Dev 2
6. Mihalcea R, Tarau P (2004) Text rank: bringing order into texts. In: Association for Computational Linguistics, July 2004
7. Sharma B, Tomer M, Kriti K (2020) Extractive text summarization using F-RBM. J Stat Manage Syst 23(6)
8. Erkan G, Radev DR (2004) Lex rank: graph-based lexical centrality as salience in text summarization. J Artif Intell Res
9. Xiangen H, Zhiqiang C, Max L, Andrew O, Phanni P, Art G (2003) A revised algorithm for latent semantic analysis. In: Proceedings of the 18th international joint conference on artificial intelligence