Chapter 5
Survey of Sentence Scoring Techniques for Extractive Text Summarization

Anushka A. Deshpande and Vinayak G. Kottawar

1 Introduction

Text Summarization is formally defined as technique of collecting important information from an elaborated original text and present it in the form of a summary [1]. Recently, the need for summarization has grown in many domains, and its applications include but are not limited to news articles summarization, email summarization, research papers summarization, medical history summarization and in website summarization to gain information regarding the relevance of that web page to the search made. The primary aim of automatic text summarization is to provide a short summary of a relatively large source text.

Text summarization techniques are broadly classified into two categories: Abstractive and Extractive Summarization [1]. Abstractive Summarization is that in which the source text is read and understood by using linguistic methods. These methods require a deeper understanding of the text but also have the ability to generate new sentences which improve the focus and reduce the redundancy of the text. On the other hand, Extractive Summarization employs various methods to extract sentences and phrases from the source text and groups them together to form a comprehensive summary [1]. This is done without altering any part of the extracted text.

2 Literature Survey

This section describes the related research work in the field of Text Summarization. Text Summarization is broadly classified into two categories, Abstractive and Extractive Summarization. Abstractive Summarization consists of the process of linguistic

A. A. Deshpande (✉) · V. G. Kottawar
D. Y. Patil College of Engineering, Pune, India

S. Bhalla et al. (eds.), *Proceeding of International Conference on Computational Science and Applications*, Algorithms for Intelligent Systems,
https://doi.org/10.1007/978-981-19-0863-7_5

understanding of the text followed by the generation of a summary. This does not use sentences part of the input text, but, creates new, meaningful sentences for the summary. In Extractive Summarization, sentences from the input text are used as is to form the summary of the text. To perform this operation, each sentence is scored based on certain parameters.

In Madhuri and Ganesh Kumar [2], the author proposes a method for scoring sentences based on the frequency of words. The input file is tokenized into words and stopwords are removed from the corpus. Words not removed are known as keywords and play an important role in summarization. A dictionary of keywords and their frequencies is formed by iterating through the corpus. Weighted frequency is calculated for each sentence by diving the frequency in the sentence by the total frequency in the text. Score of each sentence is the sum of weighted frequencies of the words in that sentence.

In Ramos [3], the concept of Term Frequency—Inverse Document Frequency is proposed for finding word relevance for document queries. TF-IDF method is used for finding the topics of a document. Using these, document retrieval is performed for different queries.

In Shiva Kumar et al. [4], the authors have used the Gensim algorithm to summarize the text. The Gensim algorithm uses multiple techniques such as Word2Vec, FastText, Latent Semantic Indexing, etc. to discover the semantic structure and relationship of the sentences in the document. Based on the results, the sentences are scored and a summary is generated.

3 Extractive Text Summarization Techniques

There are various approaches to perform extractive text summarization. In this subsection, a brief summary of each of these methods is described.

3.1 Statistical-Based Methods

These methods select a sentence based on some assigned weights that determine the relevance for a sentence. The “most important” sentence is the one which has the highest weight assigned [5]. In most of these methods, a value is assigned to each sentence based on the words in that sentence. Multiple algorithms such as word frequency are utilized to assign a value to each sentence.

3.2 Concept-Based Methods

In these methods, concepts are extracted from external knowledge bases. Using these extractions, a vector or graph based model is created to demonstrate a relationship between the concept and sentence [6]. Then, a ranking algorithm is used to extract the most relevant sentences.

3.3 Topic-Based Methods

These methods are dependent on identifying the topic of a document [7]. Methods such as TF-IDF are used to identify the topic of a document. After identifying the topic, each sentence is scored based on its relevance and importance to the topic. Highest scored sentences are then used to form the summary.

3.4 Clustering-Based Methods

These methods are useful in multi-document summarization. All the sentences of each document are placed in a vector space, and clustering algorithms are used to determine similar sentences [8]. The centroid of each cluster, which is a sentence, is used to form the summary of these documents. These methods consider both redundancy and relevance to form the summary.

3.5 Semantic-Based Methods

These methods intend to consider the meaning of the words or sentences to form a summary. LSA is an example of an unsupervised semantic-based method which observes the co-occurrence of words to determine text semantics [9].

3.6 Machine-Learning Based Methods

These transform the problem of text summarization into a supervised learning problem. Models are created to classify each sentence are “summary” or “non-summary” with the help of a training dataset. Neural networks are used to train the model which generates an output score for each sentence [6].

4 Sentence Scoring Algorithms

Extractive text summarization techniques are heavily dependent on sentence scoring algorithms to determine whether a sentence is part of the summary. These algorithms are discussed in this section.

4.1 Word Frequency Algorithm

The word frequency algorithm is the simplest of all algorithms for sentence scoring [2]. A word frequency table is created for the text. Using this table, each sentence is scored. For example,

Consider a sentence S_i with words $W[n]$, then the score of the sentence is (1):

$$\text{Score}[S_i] = \sum \frac{(F[W[i]])}{n} \tag{1}$$

where,

$W[i]$ ith word in the sentence
$F[x]$ frequency of word 'x'
n number of words in the sentence.

This method gives sufficiently accurate results. But, since this method is heavily dependent on the vocabulary of the document, there is a high chance of irrelevant sentences becoming part of the summary. Another drawback is that semantically similar words have independent effect on the sentence. This implies that words such as "run" and "sprint" are counted separately.

4.2 TF-IDF Algorithm

The TF-IDF approach is best used when there are multiple documents to be summarized into a single summary. TF-IDF considers two very important values as follows:

Term Frequency (TF) of a word is defined as [10],

$$\text{TF} = \frac{\text{number of times } t \text{ occurs in a document}}{\text{Total number of terms in the archive}} \tag{2}$$

The Inverse Document Frequency (IDF) is defined as [10],

$$\text{IDF} = \ln\left(\frac{t}{n}\right) \tag{3}$$

Essentially, TF identifies how incessant a word is, while IDF identifies how a unique a word is. TF-IDF is the score obtained on multiplying these two values [10].

$$\text{TFIDF} = \text{TF} * \text{IDF} \tag{4}$$

where,

t term or word in the document
n number of documents in the archive with the term 't' in them.

TF-IDF is an effective method to generate quick extractive summaries. One of the major downsides of this method is that it can be extremely time consuming in case of large archives that need to be summarized. Also, similar to the word frequency algorithm, this does not take into consideration, the semantic meaning of the word. This can result in imprecise summaries.

4.3 TextRank Algorithm

The TextRank Algorithm is based on the PageRank algorithm used to calculate the weight for web pages [3]. While using the algorithm for TextRank, each web page which was a vertex, now represents each sentence. In this manner, we can find the similarity of two sentences. The formula for calculating the similarity is [11],

$$S(V_i) = (1-d) + d * \sum_{j \varepsilon \text{In}(v_i)} \frac{1}{|\text{Out}(V_j)|} S(V_j) \tag{5}$$

where,

$S(V_i)$	the weight of the sentence 'i'
d	damping factor, in case of no outgoing links. The default value for this is 0.85
$\text{In}(V_i)$	set of Inbound links to sentence 'i'
$\text{Out}(V_j)$	set of Outbound links to sentence 'j'
$\|\text{Out}(V_j)\|$	number of Outbound links to sentence 'j'.

The TextRank algorithm can be closely compared with the TF-IDF approach since both algorithms use scoring methods to determine relevance of a word or sentence [3]. The main difference between the two is that TextRank uses the context information of the words to assign the weights. Thus, it assigns comparatively lower scores to words that co-occur only with stop words. On the other hand, TF-IDF simply uses the single word frequency to assign the weight to each word. Thus, it can be said that TextRank weighs the words or sentences with more "strictness" as compared to TF-IDF.

4.4 KL Sum Algorithm

The KL Sum (Kullback-Lieber Sum) algorithm is a sentence scoring algorithm in which the length of the summary is fixed to L words. This method greedily adds sentences to a summary as long as it decreases the KL Divergence [11].

The KL Divergence measures how a probability distribution is different from another. If the divergence is less, it implies that the documents are similar to each other in terms of understanding and meaning conveyed. The KL Divergence is always non-negative and calculated as follows [11]:

$$D_{\mathrm{KL}}(P||Q) \geq 0 \tag{6}$$

$$D_{\mathrm{KL}}(P||Q) = \int_{x_a}^{x_b} P(x) \log \frac{P(x)}{Q(x)} \mathrm{d}x \tag{7}$$

where,

P document set unigram
Q summary distribution
x sentence.

Since this sentence compares the entire document to each sentence to find the KL Divergence, semantic meaning is also taken into account. Through the divergence, relevance of each sentence to the summary is calculated and maintained. Thus, this can give the most accurate summary of all the mentioned algorithms.

5 Text Pre-processing for Each Algorithm

5.1 Word Frequency Algorithm

The word frequency algorithm depends completely on the words in the corpus. Hence, it is necessary to meticulously pre-process the text data to obtain accurate results. First, all punctuation marks are removed from the text and, in case of multi-document summarization, the text is concatenated to form one whole text.

Next, stopwords are removed from the text. Stopwords are those words that do not contribute to the meaning of the passage, for example, “the”, “a”, “an”, “he”, “she”, “it” and so on.

Most times, stemming of the words is also performed. Stemming is performed using the Porter’s Stemmer method [2]. This method essential converts each variation of the word into a single root word by removing any prefixes and suffixes. For example, the words “flying” and “fly” are both reduced to “fly”.

After performing stemming, the corpus is ready for the word frequency algorithm to be applied.

5.2 TF-IDF Algorithm

This algorithm can calculate the frequency as well as the importance of a word in the document. Hence, it is not essential to remove stopwords from the corpus. On the contrary, since each word will be allocated a certain importance based on frequency, it is essential to group the similar words together.

Stemming and lemmatization is performed to the text during pre-processing. Stemming refers to reducing the word to its stem or root. Lemmatization considers the place of the word in the sentence, that is, it performs a morphological analysis of the words [10]. To understand their difference, consider the following example.

Stemming for the words "copying" and "copies" are follows:

Form	Suffix	Stem
Copies	-es	Copi
Copying	-ing	Copy

On the other hand, lemmatization for the same two words is:

Form	Morphological information	Lemma
Copies	Third person, singular, present tense of "copy"	Copy
Copying	Continuous tense of verb "copying"	Copy

From above, it can be seen that lemmatization is a more meaningful conversion of a word to its root. After applying stemming and lemmatization, the TF-IDF algorithm can be applied to the corpus.

5.3 TextRank Algorithm

The TextRank algorithm is applied to a graph in which the vertices are sentences of the corpus [3]. To generate this graph, text is first pre-processed. Steps followed in pre-processing the data are:

1. Sentence tokenization
2. Removal of punctuation marks
3. Removal of stopwords
4. Removal of duplicates
5. Changing the text to lowercase.

After the pre-processing of the text, word embeddings are used to represent the individual words are real-valued vectors. The most common algorithm used is GloVe, the Global Vectors for Word representation. It is an extension to the word2vec method of word embedding.

GloVe is an unsupervised learning algorithm used for obtaining vector representation of words [4]. It is based on two most common algorithms used: Latent Semantic Analysis (LSA) and Word2Vec.

Latent Semantic Analysis uses global statistics to derive semantic relationships between words in a corpus. The fundamental idea behind Word2Vec is that a dataset of tuples is formed consisting of (some word X, a word in the context of X). Then a neural network is applied to predict the context of X, given the word X [4]. The major disadvantage of this is that Word2Vec relies only on the local information of that language. Thus, the semantics that are learnt for a given word are only affected by the surrounding words. Word2Vec, which captures local statistics, works great with analogy tasks. On the other hand, LSA does not perform well in analogy tasks.

GloVe algorithm combines the best of both algorithms [12]. According to the author, "GloVe embeddings are a type of word embeddings that encode the co-occurrence probability ratio between two words as vector representations" [4]. GloVe uses the weighted least square objective "J". It minimizes the difference between the dot product of the vectors of the two words and the logarithm of their number of co-occurrences. The mathematical formula for "J" is as follows:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \overline{w}_j + b_i + \overline{b}_j - \log X_{ij}\right)^2 \tag{8}$$

where,

w_i Word Vector of word 'i'
b_i Bias of word 'i'
$\overline{w}_j$ Context Word Vector of word 'j'
b_j Bias of word 'j'
X_{ij} number of times 'i' occurs in the context of 'j'
f is the weighted function that assigns lower weights to rare and frequent co-occurrences.

One important highlight is that the nearest neighbour algorithms can be applied to find similar words. The Euclidean distance in this space determines the similarity of two words.

Next, vectors are created for each sentence which is then plotted into a similarity matrix. A similarity matrix is a numerical representation of the similarity between any two sentences. In the paper, a cosine similarity is used to compute the similarity of two sentences [3]. This matrix is then converted into a graph where each node represents a sentence and each edge between two nodes represents their cosine similarity value. On this graph, the TextRank algorithm is applied to generate the "n" most relevant sentences for the summary.

5.4 KL Sum Algorithm

The KL Sum algorithm finds the KL divergence for a sentence to determine whether it will be included in the summary or not [11]. To calculate this divergence, again, pre-processing of text is required.

Since the divergence depends on the words in the sentence, removing stopwords in the sentence becomes essential [13]. After removing stopwords, the rest are normalized before calculating the word frequencies. In normalization, the word is converted into its base form. Normalization is usually performed in 3 steps which include: stemming, lemmatization and everything else (conversion to lowercase, removing numbers, removing punctuation and removing extra white spaces).

After performing normalization, a word frequency table is generated and term frequency of each is calculated. Then, the KL Divergence is calculated for each sentence. The sentences that have minimum divergence values are taken as part of the summary.

6 Experimental Analysis

Each of these algorithms was applied to the BBC News Dataset [14]. The dataset consists of 2225 documents from the BBC news reports corresponding to five topics from 2004 to 05. The five classes are: business, entertainment, politics, sports and technology.

The dataset also contains files of ideal extractive summaries. The similarity of the summary generated by each of these algorithms was calculated against the ideal summaries. For each of the topic areas, an average of the similarities was computed which is displayed in Table 1 (where "Ent" stands for Entertainment):

From the above table, it can be concluded that the TextRank algorithm has a higher probability of accurately ranking sentences and generating a comprehensible summary for most topics. To understand the dispersion of similarities for these algorithms, a graph was plotted as shown below.

It can be successfully concluded from Fig. 1 that although the algorithms give an average similarity index in the range of 40–60%, the spread of the similarity scores

Table 1 Average similarity index for algorithms (values are in percentages)

Topic	Business	Ent	Politics	Sport	Tech
Word Freq	49.33	48.90	51.93	**47.60**	**54.41**
TF–IDF	53.93	51.52	46.46	42.82	47.16
KL Sum	53.69	54.43	45.68	43.68	47.73
TextRank	**58.08**	**56.49**	**53.78**	44.89	51.44

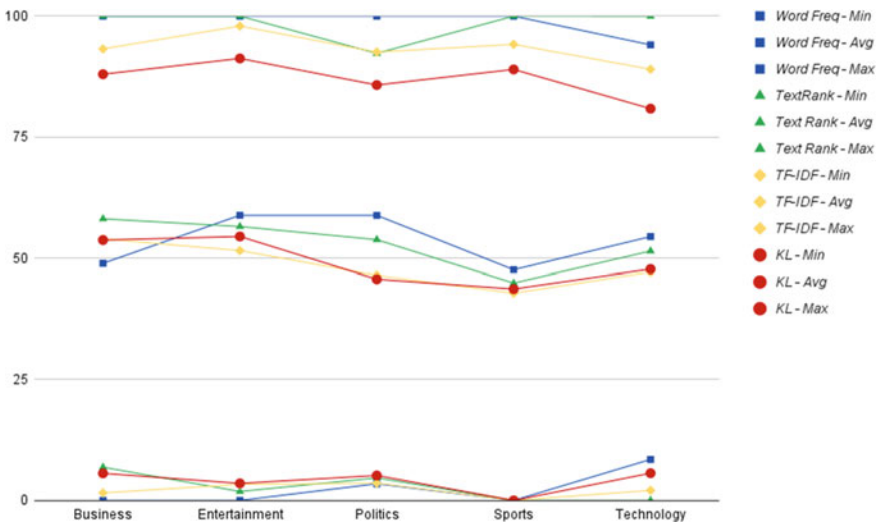

Fig. 1 Line chart showing spread of similarity values for each topic

Table 2 Standard deviation for similarity index for algorithms

Algorithm	Business	Ent	Politics	Sport	Tech
Word Freq	0.247	0.257	0.213	0.236	0.188
TF–IDF	0.186	0.184	**0.184**	0.205	0.191
KL Sum	**0.170**	**0.181**	0.190	**0.202**	**0.176**
TextRank	0.183	0.207	0.192	0.223	0.192

varies between 3 and 97%. Hence, mean of the similarity index is not an effective way to determine effectiveness of these algorithms.

Therefore, standard deviations for each of these observations was also calculated. These are tabulated in Table 2.

From Table 2, it can be inferred that in most topical areas, and the KL Sum algorithm shows minimum deviation. The algorithms that show high average similarity also exhibit higher standard deviation. This implies that although the algorithms are more accurate; there is a higher chance of inaccuracy.

Besides the mathematical differences, some other advantages and disadvantages are enlisted in Table 3.

7 Conclusion and Future Scope

Manual text summarization is an extremely tedious and time consuming task. Due to this, Automatic Text Summarizers have emerged and taken up this role.

Table 3 Advantages and disadvantages of the algorithms

Method	Advantages	Drawbacks
Word frequency	It requires less processing power. There is no need of linguistic knowledge and hence, the method is language independent	Important sentences may fail to be included in the summary due to their less score. Similarly, sentences having same meaning may be included due to high scores
TF-IDF	It is suitable for "single domain multi-document" summarization since it calculates significance as well as frequency of words	Sentences having low scores may not be included in the summary, even if they are relevant
KL Sum	It performs exceptionally well at identifying the relevance of a sentence to the topic	The summary generated is not comprehensible since the order of the sentences is vague
TextRank algorithm	It can detect coherence as well redundancy in the sentences. This algorithm is also domain-independent	TextRank algorithm disregards any word which has a lower chance of occurrence, despite maybe being meaningful in the context. This can be enhanced through the use of GloVe or Word2Vec approaches to word embeddings

There is continuous research taking place in this field, yet, it is far from being comparable to human summarization. A majority of survey papers are based on extractive summarization techniques since it is an easier approach towards the problem. Abstractive techniques require the understanding of language, like a human brain, which is a difficult task.

The main contributions of this paper include:

- Explaining the various Extractive Text Summarization methods
- Explaining the most commonly used sentence scoring algorithms for text summarization
- Providing a listing of the future research scope in this field.

The future scope of research would be to overcome the following challenges.

7.1 Challenges for Multi-document Summarization

One of the biggest challenges for multi-document summarization is redundancy [1]. There is a high chance of similar sentences being used in various documents. This may result in similar sentence scores and can affect the sentences being included in the summary. Secondly, some sentences may refer to previously mentioned content in a document. While using that sentence in the summary, context may be lost for the reader.

7.2 Challenges for Input

Presently, most summarizers work well for summarizing short length documents. When put to use on longer texts, for example, chapters of a book, the accuracy reduces significantly [1].

7.3 Challenges Related to Length of Summary

When humans summarize a text, the length of the summary varies based on the content. Deciding the stop condition for automatic summarization is challenging [1]. Setting a retention rate is the most common way, but this is not the same for all summaries.

7.4 Challenges for Evaluating the Summary

It is difficult to define what a good summary constitutes of and varies based on the type of text being summarized [1]. For example, the criteria for a good summary for sports articles and academic research articles will differ. Also, whether a summary is understood by humans or not is very subjective. Every person may select different sentences for summarization, depending on what they find useful and important. There is a need to propose methods for standardizing this.

References

1. El-Kassas WS, Salama CR, Rafea AA, Mohamed HK (2021) Automatic text summarization: a comprehensive survey. Elsevier
2. Madhuri JN, Ganesh Kumar R (2019) Extractive text summarization using sentence ranking. In: International conference on data science and communication (IconDSC)
3. Ramos J (2003) Using TF-IDF to determine word relevance in document queries. In: Proceedings of the first instructional conference on machine learning, vol 242, no 1
4. Shiva Kumar K, Priyanka M, Rishitha M, Divya Teja D, Madhuri N (2021) Text summarization with sentimental analysis. Int J Innovative Res Comput Sci Technol (IJIRCST) 9(4)
5. Gupta V, Lehal GS (2010) A survey of text summarization extractive techniques. J Emerg Technol Web Intell 2(3):258–268
6. Moratanch N, Chitrakala S (2017) A survey on extractive text summarization. Paper presented at the 2017 international conference on computer, communication and signal processing (ICCCSP), Chennai
7. Nenkova A, McKeown K (2012) A survey of text summarization techniques. In: Aggarwal CC, Zhai C (eds) Mining text data. Springer US, Boston, pp 43–76
8. Nazari N, Mahdavi MA (2019) A survey on automatic text summarization. J AI Data Min 7(1):121–135

9. Al-Sabahi K, Zhang Z, Long J, Alwesabi K (2018) An enhanced latent semantic analysis approach for Arabic document summarization. Arab J Sci Eng
10. Kiran Kumar G, Malathi Rani D (2021) Paragraph summarization based on word frequency using NLP techniques. In: AIP conference proceedings, vol 2317, no 1
11. Tanvi, Ghosh S, Kumar V, Jain Y, Avinash B (2019) Automatic text summarization using TextRank. Int Res J Eng Technol (IRJET)
12. Pennington J, Socher R, Manning CD (2014) GloVe: global vectors for word representation. Stanford University
13. Baoyi W, Zhang S (2005) A novel text classification algorithm based on naïve bayes and KL-divergence. In: Sixth international conference on parallel and distributed computing applications and technologies (PDCAT'05). IEEE, 2005
14. Greene D, Cunningham P (2006) Practical solutions to the problem of diagonal dominance in Kernel document clustering. In: Proceedings of the ICML