JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 39, 561-574 (2023)
DOI: 10.6688/JISE.202305_39(3).0007

High Impact of Rough Set and K-Means Clustering Methods in Extractive Summarization of Journal Articles

SHEENA KURIAN K+ AND SHEENA MATHEW
School of Engineering
Cochin University of Science and Technology
Kerala, 682022 India
E-mail: sheenakuriank@gmail.com; sheenamathew@cusat.ac.in

Text Summarization is the technique of shortening a long text but having all the relevant and significant topics conveyed in the text. As the number of journal articles published every year is growing steadily, the relevance of research on journal article summarization also increases. In this work, six extractive summarization methods are implemented and compared with the results of four standard methods applied to the dataset of journal articles. Precision, Recall and F-measure of Rouge-1, Rouge-2, Rouge-*L* and Rouge-Lsum measures are analyzed. Eight features are used in the implementation of the sum of features method and the BernoulliRBM method. It is observed from the experiments conducted that the Rough set method and the *K*-Means clustering and summarization method have high rouge scores in 10 out of the 12 measures analyzed here. The recall of the generated summary by the Roughset method is further improved when the first part of the article is used as a heuristic yard in calculating the similarity score with selected sentences.

Keywords: summarization, rouge score, KMeans clustering, rough set, graph method

1. INTRODUCTION

The research on text summarization started way back in the 1950's [1]. The relevance of this area of study keeps on increasing every day as the volume of e-text grows exponentially. Text summarization helps to get a gist of the content described in the article, which saves time for the readers. Text summarization has applications in many fields like generating news titles or summaries, media monitoring, question and answering, social media marketing, telemedicine, legal contract analysis and journal article summarization. The motivation behind this work is that journal article summarization will be of great help to researchers who are new to a particular field and those working in interdisciplinary areas of study. The summarized text will help readers choose the relevant journal articles for elaborate reading. This work focuses on the extractive summarization of journal articles.

Text summarization process is classified into extractive and abstractive summarization. An extractive summary is formed by ranking the sentences and selecting the highly ranked ones from the text. There are different ways of ranking and selecting sentences. Statistical features like word frequency, uppercase words, sentence length, keywords, position in the text, and phrase structure are used commonly for sentence ranking [1–4]. An

Received February 7, 2022; revised May 2 & 27 2022; revised July 29, 2022.
Communicated by Jimson Mathew.
+ Corresponding author.

abstractive summary is formed by understanding the original text and generating a summary from the concepts, like the human summary of a document. Many linguistic techniques may have to be used to generate an abstractive summary. The research was being carried out more extensively on extractive summarization than abstractive summarization until recently. Currently, many works are going on in generating abstractive summaries with the developments in deep learning techniques and computational power [5].

The large number of journal articles published each year are now easily accessible to the readers. It would be of great help to the researchers, new to a particular field if they could get a good summary of the journal article. They can easily understand the concepts discussed in the article by reading the summary. The articles that require elaborate reading can then be easily selected. This work aims to identify a summarization technique that generates a comprehensive summary of the journal articles. Rouge (Recall-oriented understudy for gisting evaluation) evaluation measure is used to analyze the quality of the generated summary. The rouge scores evaluate the overlaps of words in the generated summary over the gold summary [6].

Section 2 explains important related works in the area of text summarization based on which the experiments explained are designed. Section 3 describes the extractive summarization approach. Section 3.1 defines the features used in the Sum of features method and the BernoulliRBM method in this work. Section 3.2 explains the extractive summarization methods implemented in this work, and Section 3.3 explains the standard methods implemented for the dataset under consideration, for comparing the results. Section 3.4 explains the method of calculating a heuristic text similarity score. Section 4 gives the results and discussion. Section 5 presents the concluding remarks of the paper.

2. RELATED WORKS

There are various methods in literature to solve the problem of text summarization. This section discusses the earlier works based on which the experiments in this paper are designed. The earliest works on scientific document summarization use statistical measures like word frequency and relative position to find the significant words and rank sentences [1]. A writer usually repeats words that are important to the topic of discussion. The words that are closely associated intellectually tend to be close physically also, and they are clustered. The stop words (of, an, is, a, at, the) appearing very frequently in the document are removed before processing as they do not carry any significant meaning. The important sentences are identified by analyzing the frequency of words and are extracted to form the summary [1]. The position of a sentence is also important in calculating the significance of a sentence [2]. Features like cue words and document structure were found to be significant in text summarization [3]. A large variety of features were identified over the years for the summarization task. Features include the mean term frequency-inverse sentence frequency, sentence length, sentence position, similarity to the title, similarity to keywords, sentence to sentence cohesion, sentence to centroid cohesion, depth in the tree, the occurrence of proper names, the occurrence of non-essential information *etc*. [4]. Few of these features found important by the authors of this paper were used in the sum of features and BernoulliRBM method implementation.

Graph-based approaches use the mutual relationship of sentences to rank and select

sentences to generate a summary. Words, sentences or paragraphs can be the nodes and the relationship between nodes be the edges of the graph. The edges are assigned weights based on sentence similarity measures like cosine similarity, term frequency-inverse document frequency (tfidf) *etc*. Sentences are selected to be included in the summary if the similarity index among nodes is above a defined threshold. TextRank uses the keywords or sentences of an article to form the nodes and edges if they have common words between the nodes. TextRank algorithm uses a voting-based method for scoring sentences based on the incoming and outgoing edges. An edge exists between keywords if they have common nouns, adjectives, or verbs [7, 8]. A paragraph provides more context and can be a unit of extraction with more readability and coherence. A highly bushy node is related to several other nodes and is a good candidate to be included in the summary [9].

Extractive summarization can be done using the rough set concept of reduct and core [10]. A rough set is a formal approximation of a conventional crisp set in terms of a pair of sets giving the lower and upper approximation of the original set [11]. Data is represented as an information table with rows representing the different words (entities) and columns representing the different sentences (attributes). Rough sets can be used to find dependency between attributes and reduce superfluous attributes, or find the most significant attributes from the attribute list. Equivalence relation induces partitioning of the universe. The words are partitioned into equivalence classes based on the co-occurrence of words in sentences. The objects belonging to the same equivalence class are indiscernible. Two objects are indiscernible if they co-occur in all sentences in which they appear. Reduct is the minimal attribute subset representing the same information as the original attribute set. The intersection of reducts makes a core set representing the key attributes of the system containing most of the knowledge in the data. Reducts of sentences are found by finding those sets which can produce the original equivalence classes by omitting one sentence at a time. A core is formed by taking the intersection of sentences in all reducts [10]. The core set defines the summary of the article.

Restricted Boltzmann Machine (RBM) is a generative stochastic neural network that learns a probability distribution over its input sentence-matrix containing the set of feature vectors. The term restricted means that the neurons in the Boltzmann machine must form a bipartite graph. This helps in the efficient training of the network using a contrastive divergence algorithm. It finds a positive gradient as the outer product of training sample v and hidden activation vector h. Gibbs Sampling is used to sample a reconstruction v' of visible units from h, and then resample the hidden activations h' from this. The negative gradient is found as the outer product of v' and h'. Then the weight matrix is updated as positive gradient minus negative gradient, times the learning rate [12, 13].

Various machine learning and deep learning techniques are also employed to improve the quality of the generated summary [14,15]. Transfer learning is a powerful technique in natural language processing (NLP) where a model is pre-trained on a data-rich unlabeled text data with a self-supervised task, such as language modelling or filling in missing words. The model is then fine-tuned on smaller labelled datasets for tasks like question answering, document summarization, sentiment classification *etc*. [16]. This results in better performance than training on the labelled data alone [17]. Summary using BERT pre-trained model is generated for analysis in this work. Bidirectional Encoder Representation from Transformers (BERT) is a bidirectional language representation model built using a 12-stacked transformer encoder and trained with a masked language modelling

and next sentence prediction task on a corpus of 3,300M words [18]. The pre-trained BERT model can be fine-tuned with just one additional output layer for the different NLP tasks. BERT combines word and sentence representations on a very large single Transformer. A sequence of tokens will be transformed into token embeddings, segment embeddings and position embeddings. Token embeddings refer to contextualized word embeddings. Segment embeddings include two embeddings(0 or 1) to represent the first sentence and second sentence. Position embeddings store the token position relative to the sequence. BERT uses 12 (bert base) or 24 layers (bert large) of attention, and also incorporates 12 or 16 attention heads in every layer. Since model weights are not shared between layers, a BERT model has 24x16 different attention mechanisms [18].

A document-level encoder based on BERT expresses the semantics of a document and obtains representation for sentences. The extractive model BERTSUM is built on this encoder by stacking several inter-sentence transformer layers to capture document-level features for extracting sentences. For abstractive summarization, a fine-tuning schedule adopts different optimizers for encoder and decoder. Encoder is a pre-trained BERTSUM, and decoder is trained from scratch. Long-range dependencies are not well-captured by BERT pre-trained on sentence pairs. Discourse aware summarization model extracts subsentential discourse units instead of sentences. Structural discourse graphs are constructed based on RST trees and coreference mentions to capture long-range dependencies among discourse units [19, 20]. The hierarchical structure of the document is made use of to generate an extended summary of the article through a multitask learning approach- the BERTSUMEXT model deciding on whether to include a sentence in the summary or not, and a section prediction task predicting the relevant section for each sentence in the document [21]. Sentences appearing in sections like introduction, overview, or motivation are used as pointers to extract salient information from the article [22]. A good summary is semantically similar to the source document. A Siamese-BERT architecture can compute the similarity between the source document and candidate summary [23].

3. EXTRACTIVE SUMMARIZATION METHODS

The entire XML file of the journal article in the dataset is used as input in the summarization process. The steps followed in the summarization process are listed below.

1. Pre-processing: Text is extracted from the XML file of the article. The acknowledgement section is not considered. Sentences with less than three words, special symbols (non-alpha-numeric characters, citations) or short terms (Figure, Table, Section, formula) are removed as they are not good candidates for the summary.

2. Different extractive summarization methods explained in Sections 3.2 and 3.3 are applied to the cleaned data to rank the sentences. Top scored sentences are selected to form the summary until they form 30% length of the original article. These sentences are arranged in the order in which they appear in the original article.

3. Precision, Recall and F-Measures of Rouge-1, Rouge-2, Rouge-L and Rouge-Lsum are used to find word overlapping scores between the summary generated by different methods and the gold summary available in the dataset.

3.1 Features Used in Journal Article Summarization

The following statistical features are used in the implementation of the Sum of features method and the BernoulliRBM method of journal summarization in this work [4, 13, 24]. These features are assigned values using the equations given below.

1. Length of the sentence is calculated as $\frac{count\ of\ words\ in\ the\ sentence}{length\ of\ the\ longest\ sentence\ in\ the\ article}$.

2. The keyword list is formed from unique words in the article title other than the stop words, keywords listed in the article, and uppercase words in the abstract. The keyword match of a sentence is calculated as $\frac{count\ of\ words\ matching\ keyword\ list}{length\ of\ the\ longest\ sentence\ in\ the\ article}$.

3. The sentence position score is calculated based on position, i of the sentence in the section as $\frac{1}{i}$.

4. Term frequency-Inverse sentence frequency (tf x isf): To find the (tf x isf) score of a sentence, the normalized term frequency of a word and inverse sentence frequency of a word is calculated for each word in a sentence and then added up using the following equations.

   $$Term\ frequency\ of\ a\ word(tf_{word}) = \frac{Number\ of\ occurrences\ of\ a\ word\ in\ the\ article}{Total\ number\ of\ sentences(N)}$$

   $$Inverse\ sentence\ freq.\ of\ a\ word(isf_{word}) = \frac{Number\ of\ sentences\ that\ include\ the\ word}{Total\ number\ of\ sentences(N)}$$

   $$Term\ freq.\ Inverse\ sentence\ freq.\ of\ a\ word(tfisf_{word}) = tf_{word} \times isf_{word}$$

   $$Term\ freq.\ Inverse\ sentence\ freq.\ of\ a\ sentence(tfisf_{sent}) = \sum_i tfisf_{word\,i}$$

   The normalized value of tf x isf score of a sentence is calculated using the equation: $tfisf_{sent\,normalized} = \frac{tfisf_{sent}}{max.\ value\ of\ tfisf_{sent}\ in\ article}$.

5. Sentence to sentence cohesion is calculated using the Jaccard similarity score of each sentence in the article.
   $JS\ score\ of\ s_i = \sum_i^j i \neq j \frac{No.\ of\ unique\ words\ common\ to\ s_i\ and\ s_j}{No.\ of\ total\ unique\ words\ in\ s_i\ and\ s_j}$

6. Sentence to centroid cohesion: Centroid is calculated using the K-means algorithm (K = 1), and pairwise cosine similarity of each sentence with the centroid sentence is calculated.

7. The positive score of a sentence is calculated as the count of positive cue words. Words like summary, result, precisely, important, conclusion, purpose, argue, and develop are considered positive cue words. Sentences with positive cue words are good candidates to be included in the summary than others.

8. Negative score of sentence is the count of negative cue words in the sentence. Words like however, for example, because, additionally, previous, furthermore, and pronouns is considered negative cue words. Sentences with negative cue words are not good candidates for summary.

3.2 Methods Implemented for Extractive Summarization of Journal Articles

3.2.1 Sum of features method

The eight features defined above are used to calculate the score of a sentence. It is defined as the sum of the products of weights and feature values of a sentence. All feature weights and threshold are assumed to be one. $Total\ Score = \sum_i x_i$. The sentences are selected to be included in summary only if total score of a sentence is above the fixed threshold. If there are many sentences with total score above the threshold, highly ranked sentences are selected and added to the summary till the required summary length. Here, the summary length is set as 30% length of the original article. A variant called weighted sum of features method was also implemented with different weights for different features (for example, weight for keyword match = 4, sentence position = 3, sentence to sentence cohesion = 2, others = 1 and threshold = 5). The rouge scores of summary generated by these two methods is compared with the gold summary, and it is seen that both methods have almost the same performance. The sum of features method is chosen arbitrarily for further analysis here.

3.2.2 Graph with bushy path method

A graph is built with sentences as vertices and the number of common words between two sentences as the weight of the edges connecting the vertices. The score of a sentence is the number of its edges with a weight greater than 0. A sentence with a high score is connected to many other sentences and is a good candidate for the summary. Top scored sentences are selected to form the summary depending on the required length. The stop words are removed and a sentence-to-sentence matrix is created with the count of common words between sentences. If the edge weight is higher than the set threshold, both vertices connecting the edges are selected. The selected list is converted into a set to avoid the repetition of vertices. The edge weights are sorted and the vertices with high scores are selected to form the summary of the required length. The vertices are sorted so that the summary has sentences in the order of appearance in the article.

3.2.3 Graph with sum of edge weights method

The original article is cleaned, and the sentence to sentence matrix is build after removing the stop words, with each entry as the count of common words in both sentences. Sentences form the vertices of the graph, and the word counts the edge weights of the graph. Row sum of the sentence to sentence matrix (sum of edge weights of all edges of a sentence) is calculated, and vertices (sentences) with the highest scores are added to the summary list. The vertex numbers in the summary list is sorted. The summary is formed by arranging the sentences in the order of appearance in the original text.

3.2.4 Rough set method

Rough set theory and fuzzy set theory are used to model incomplete and vague information, respectively. Unique words in the article without stop words are identified. The words are stemmed, and a word-sentence matrix (information table) is created. Words represent the rows (objects), and the sentences represent the columns (attributes). The

matrix entry is one if the word is present in the sentence and zero otherwise. The dimension of the information table is big if the article considered is long. The dimension of the information table is reduced by removing words occurring in only one sentence [10]. The words are partitioned into equivalence classes based on the co-occurrence of words in sentences. Reducts of sentences are found by finding subsets that produce the original equivalence classes by omitting one sentence at a time. The core is formed by taking the intersection sentences of all reducts. The sentences in the core set are selected to form the article summary. Consider a passage of 4 sentences in Table 1. The stop words and words of frequency one are removed. The information table is shown in Table 2. The equivalence class is given by {[w1], [w2,w3]}, and reducts by {(S1, S4), (S2, S3, S4)}. The core sentence is S4, and it can summarize the passage.

Table 1. Sample dataset.

S1	People have been coming to the wise man, complaining about the same problem every time.
S2	One day he told them a joke and everyone roared in laughter.
S3	After a couple of minutes, he told them the same joke and only a few of them smiled.
S4	When he told the same joke for the third time, no one laughed any more.

Table 2. Information table.

Words	S1	S2	S3	S4
w1: same	1	0	0	1
w2: told	0	1	1	1
w3: joke	0	1	1	1

Three variants of Rough set method are implemented in this work.

1. Rough Set Core Method: The words are partitioned into equivalence classes based on the co-occurrence of words in sentences. Two sentences are equivalent if they have similarities in the occurrence of words. Some of these equivalence classes of words are easily classifiable as a summary sentence (belongs to positive region) or non-summary sentence (belongs to negative region). Some others may not be easily classifiable as either summary or non-summary sentences. Reduct is the minimal set of attributes (sentences) preserving the positive region. There may be many reducts in the information table. Minimal reduct is found by finding sets that can produce the original equivalence classes of words even after n number of sentences is removed from the table. The core is the set of attributes found in all reducts. It is found by taking the intersection of sentences across all reducts. The core contains sentences that can be certainly included in the summary.
2. Rough Set Fuzzy Core Method: Rough set fuzzy core method assigns a fuzzy score value to all sentences in the article. The sentences with high fuzzy score values are selected to form the summary of the article based on the required length.

   $$Fuzzy\ Score\ of\ a\ sentence = \frac{No\ of\ reducts\ containing\ the\ sentence}{Total\ no\ of\ reducts}$$

3. Rough Set tfidf Method: From the word-sentence matrix, word-frequency counts are found by adding the row entries and are then normalized. Sentences are scored

by adding the normalized frequency count of all words in the sentence. The summary of the article is formed by adding top scored sentences for the required length.

$$Normalized\ freq\ count\ of\ words = \frac{No.\ of\ sentences\ with\ the\ word}{Total\ no.\ of\ sentences}$$

$$Sentence\ score = \sum(Normalized\ freq\ count\_all\ words\ in\ sentence)$$

The rouge scores of the summary generated by the above three methods were calculated. The summary generated by the Roughset tfidf method had the highest rouge score. Hence Roughset tfidf method is used in the further analysis in this work.

3.2.5 K-means clustering and summarization method

K-Means clustering algorithm is an unsupervised algorithm used to cluster sentences into K clusters based on the similarity measure among sentences. K-Means clustering is implemented here using methods in the sklearn.cluster package [25]. TfidfVectorizer is used for feature extraction. All the sentences in the article is partitioned into K clusters, and the sentences closest to the cluster centres are selected uniformly from each cluster to form the required summary length (30% of the article length here). Results are analyzed for the number of clusters as K=3, 5 and 7. The rouge score values were almost the same for the three cases. So K=5 is chosen arbitrary for further analysis in this work.

3.2.6 TFSummarizer

The nltk package [26] is used to tokenize the cleaned text. Stop words are removed, and the frequency of words is calculated. The score of a sentence is the sum of the frequency counts of words in the sentence. These scores are sorted in reverse order, and high-scoring sentences are selected to form the summary of 30% length of the article.

3.3 Comparison of our Extractive Summarization Methods

The following standard methods are used for comparing the results of summary generated by our methods:

1. BernoulliRBM method: Scikitlearn library has an implementation of Bernoulli RBM for summarization task [25], and is used in this work. RBMs are unsupervised nonlinear feature learners based on a probabilistic model. The features described in Section 3.1 is used in this model. RBM takes data values between 0 and 1 as the weights of edges. The different parameters are estimated using Stochastic Maximum Likelihood (SML), also known as Persistent Contrastive Divergence (PCD) [25]. This indicates the probability that the node is activated or not. It allows for quicker learning of feature sets. Each sentence is assigned a score using this method, and the top scored sentences from each section are selected to form the summary of 30% length of the article.

2. Gensim Summarization: Gensim summarizer is a variant of the graph-based ranking algorithm for NLP, called the TextRank algorithm and is used in this work [27]. The algorithm pre-processes the text by stemming and removing stop words and punctuations. A graph is constructed with sentences as vertices and edges representing the similarity between the vertices. The sentences are ranked by the Text

Rank algorithm and the highest-scoring sentences are selected to be included in the summary based on the ratio or word count specified [28]. The ratio specifies how much the article size should be reduced in the summary. The ratio of 0.3 is used in this work to generate a summary of 30% of the length of the original article.

3. LexRank algorithm is an unsupervised approach that uses a cosine transform-based weighting algorithm. The importance of a sentence or keyword is found by eigenvector centrality in the graph representation of the sentences [8]. LexRankSummarizer of the standard Sumy package is used to generate summary of the different articles in this work for comparison. PlaintextParser and Tokenizer from the Sumy package is also used in the summary generation process.

4. BERT Summarization: The HuggingFace Pytorch transformers library is used to generate an extractive summary using BERT in this work [29]. Sentences are embedded and K-means clustering algorithm is used to find the sentences that are closest to the cluster's centroids [30]. This library also uses co-reference techniques, utilizing the neuralcoref library to resolve words in summaries that need more context. The greediness value of the neuralcoref was given as 0.4 to the CoreferenceHandler class in this work. The default value of 0.2 was taken as the ratio of sentences to summarize the original text. The bert-extractive-summarizer model was used (https://pypi.org/project/bert-extractive-summarizer/).

3.4 Roughset with Heuristic Text Similarity Score

The first few sentences from the article are chosen as a yardstick when generating the summary by the Rough set method. Embedding vectors are generated for these yardstick sentences and sentences selected by the Roughset method. The pre-trained SentenceTransformer model from the Hugging Face library is used to generate sentence embeddings. The cosine similarity of these vectors is calculated to get the text-similarity score. The sentences picked by the rough set method that is most similar to the yardstick sentences are selected to form the summary of the required 30% length of the article.

4. RESULTS AND DISCUSSION

The DUC (Document Understanding Conference) datasets are the de facto standard datasets that the NLP community uses for evaluating extractive summarization systems. It consists of news articles paired with human-written summaries. SCIRLDR dataset is a combination of TLDRs written by human experts and author written TLDRs of computer science articles from OpenReview [31]. Wikipedia-summary dataset consists of text extracted from Wikipedia [32]. SumPubMed dataset has scientific articles from PubMed archive with human analysis of summary coverage, redundancy, readability, coherence and informativeness [33]. The scientific articles for summary generation task here is taken from the CL-SciSumm Github repository [34]. CL-SciSumm consists of articles in computational linguistics and NLP along with a human summary written by a trained annotator. ScisummNet dataset, the first large scale human-annotated dataset of 1009 articles and its human summary is used in this work. The dataset consists of pdf, XML, annotated citation information and a manual summary of the articles.

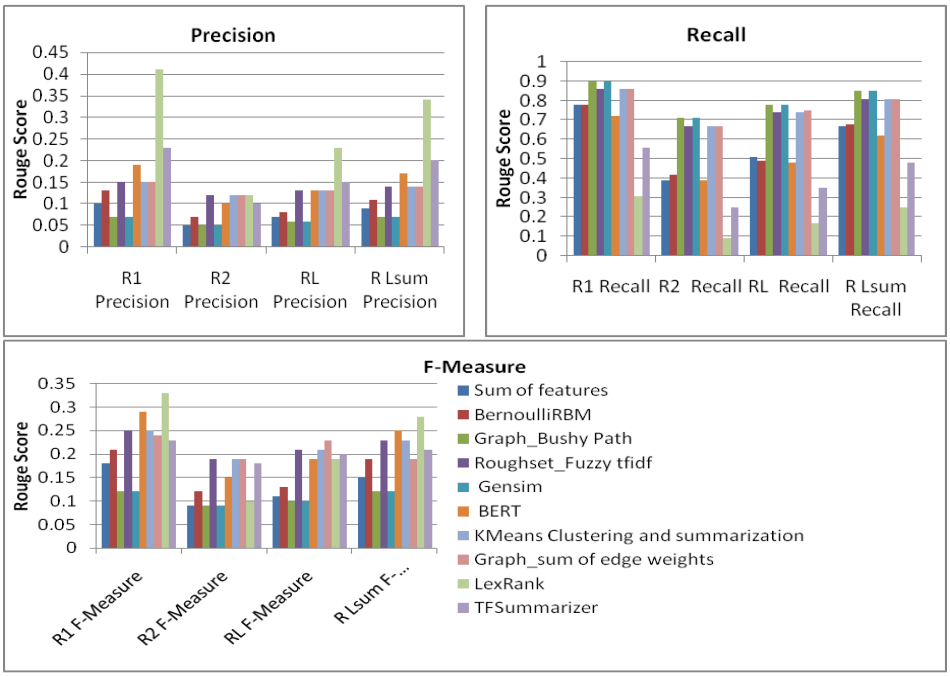

Fig. 1. Precision, Recall and F-Measure of different rouge scores of extractive summarization techniques on CLSciSumm dataset.

The different summarization techniques implemented in this work are evaluated using the precision, recall and f-measure of Rouge (Recall-Oriented Understudy for Gisting Evaluation) [6]. Rouge has a set of metrics to evaluate automatic text summarization. Rouge-1 (R1), Rouge-2 (R2), Rouge-*L* (RL) and Rouge-Lsum (RLsum) are calculated here. Rouge-*n* measures the *n*-gram overlap of terms and Rouge-*L* measures the longest common subsequence in the generated summary and the gold summary. Rouge-*L* doesn't require consecutive matches but in-sequence matches in a sentence. Rouge-Lsum is identical to Rouge-*L* except that it considers newline also. Recall indicates the number of words in the gold summary that are present in the generated summary and is given by $\frac{No.\ of\ overlapping\ words}{Total\ words\ in\ reference\ summary}$. Recall=1 indicates that all the words in the reference summary is present in the generated summary. But other less important words may also be present in the generated summary. Precision indicates how many of the words in generated summary are present in the gold summary. It indicates how much of the generated summary is relevant, and is calculated as $\frac{No.\ of\ overlapping\ words}{Total\ no.\ of\ words\ in\ generated\ summary}$. F-measure is a composite method combining precision and recall given by $\frac{2 * Precision * Recall}{Precision + Recall}$.

The 12 rouge measures of the summary generated by the different methods for 1009 articles is calculated. Precision, Recall and F-measure scores of R1, R2, RL and RLsum measures are analyzed [6]. The results are shown in Fig. 1. Rouge measures are calculated using the rouge-score package in python. TFSummarizer method has precision scores of all rouge measures close to LexRank and BERT methods. Sum of features, Graph-Bushy path, Rough set, Gensim, *K*-Means clustering and summarization, and Graph-sum of edge weights methods have high recall scores for all rouge measures. Rough set and *K*-Means

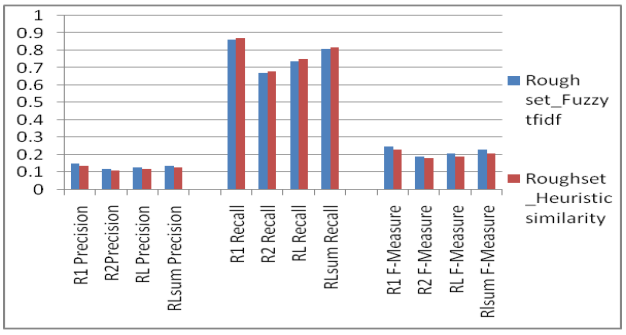

Fig. 2. Precision, Recall and F-Measure of text similarity approach on Rough set tfidf.

clustering and summarization methods have high F-measure values close to the BERT method. Also, the Rough set, *K*-Means clustering and summarization, and Graph-sum of edge weights methods have high precision, recall and F-measure values for R2 and RL. Table 3 lists the top scoring extraction methods for the 12 rouge measures.

Table 3. Performance for different systems.

	Precision	Recall	F-Measure
R1	BERT, LexRank, TF-Summarizer	Sum of features, BernoulliRBM, Graph-Bushy path, Rough set, Gensim, *K*-Means clustering and Summarization, Graph-sum of edge weights	Rough set, BERT, *K*-Means clustering and summarization, LexRank
R2	Rough set, BERT, *K*-Means clustering and summarization, Graph-sum of edge weights, LexRank, TFSummarizer	Sum of features, BernoulliRBM, Graph-Bushy path, Rough set, Gensim, *K*-Means clustering and summarization, Graph-sum of edge weights	Rough set, BERT, *K*-Means clustering and summarization, Graph- Sum of edge weights, TFSummarizer
RL	Rough set, BERT, *K*-Means clustering and summarization, Graph-sum of edge weights, LexRank, TFSummarizer	Sum of features, Graph-Bushy path, Rough set, Gensim, *K*-Means clustering and summarization, Graph-sum of edge weights	Rough set, BERT, *K*-Means clustering and summarization, Graph-sum of edge weights, TFSummarizer
R Lsum	BERT, LexRank, TF-Summarizer	Sum of features, BernoulliRBM, Graph-Bushy path, Rough set, Gensim, *K*-Means clustering and summarization, Graph-sum of edge weights	BERT, Rough set, *K*-Means clustering and summarization, LexRank

The first few sentences of an article are selected as a yardstick for measuring the similarity of the model-generated summary with the human-generated summary. Here, the first 15 sentences are chosen, considering the average length of the abstract of the article. A rough set with a heuristic text similarity approach improves the recall measure of R1, R2, RL and RLsum by 1% as seen in Fig. 2. It indicates that there is more overlap of words in the model-generated summary and the human-generated summary.

5. CONCLUDING REMARKS

The summary of a set of journal articles is generated using six different summarization methods: Sum of features, Graph-bushy path, Rough set fuzzy tfidf, *K*-Means clustering and summarization, Graph-sum of edge weights and TF Summarizer methods. The rouge scores of the generated summary using these methods are compared with the rouge scores of a generated summary using standard methods like BernoulliRBM, Gensim, BERT and LexRank methods. Rouge score is calculated by comparing the word overlaps between the generated summary and the human-written summary available with the articles in the dataset. The Roughset and *K*-Means clustering and summarization method has good rouge scores for 10 out of the 12 measures analyzed in this work, and the Graph-sum of edge weights method have good rouge scores for 8 out of the 12 measures. Rough set methods generate a summary from sentences with almost similar occurrences of words. The *K*-Means clustering and summarization method select sentences for the summary from the clusters generated. The graph-sum of edge weights method finds summary by selecting sentence related to most other sentence. The obtained results in this work are especially right as humans include sentences in summary if they have concepts connected to many other sentence. So it is concluded that out of the ten different methods used here, Rough set and K-means clustering and summarization methods have a high impact in generating an extractive summary of journal articles. An improvement in recall scores has been observed by using the first few sentences of an article as a heuristic for generating the summary. The method was tested on the Roughset tfidf approach and can be extended to others.

REFERENCES

1. H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of Research Development*, Vol. 2, 1958, pp. 159-165.
2. P. B. Baxendale, "Machine-made index for technical literature – an experiment," *IBM Journal of Research and Development*, Vol. 2, 1958, pp. 354-361.
3. H. P. Edmundson, "New methods in automatic extracting," *Journal of the ACM*, Vol. 16, 1969, pp. 264-285.
4. F. Kiyoumarsi, "Evaluation of automatic text summarizations based on human summaries," *Procedia – Social and Behavioral Sciences*, Vol. 192, 2015, pp. 83-91.
5. S. K. Kurian and S. Mathew, "Survey of scientific document summarization methods," *Computer Science*, Vol. 21, 2020, pp. 141-177.
6. C. Y. Lin, "Rouge: A package for automatic evaluation of summaries," *Text Summarization Branches Out*, 2004, pp. 74-81.
7. R. Mihalcea and P. Tarau, "TextRank: Bringing order into text," in *Proceedings of Conference on EMNLP*, 2004, pp. 404-411.
8. G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," *Journal of Artificial Intelligence Research*, Vol. 22, 2004, pp. 457-479.
9. M. Mitra, A. Singhal, and C. Buckley, "Automatic text summarization by paragraph extraction," in *Proceedings of the ACL Workshop on Intelligent and Scalable Text Summarization*, 1997, pp. 39-46.

10. R. V. Murali Krishna and C. Satyananda Reddy, "Extractive summarization technique based on fuzzy membership calculation using rough sets," *International Journal of Computational Intelligence and Applications*, Vol. 14, 2015, No. 1550012.
11. P. Zdzislaw, "Rough sets," *International Journal of Computer and Information Sciences*, Vol. 11, 1982, pp. 341-356.
12. N. S. Shirwandkar and S. Kulkarni, "Extractive text summarization using deep learning," in *Proceedings of the 4th International Conference on Computing, Communication Control and Automation*, 2018, pp. 1-5.
13. S. Verma and V. Nidhi, "Extractive summarization using deep learning," *Research in Computing Science*, Vol. 147, 2018, pp. 107-117.
14. W. H. Alquliti and N. B. A. Ghani, "Convolutional neural network based for automatic text summarization," *International Journal of Advanced Computer Science and Applications*, Vol. 10, 2019, pp. 200-211.
15. T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, "Neural abstractive text summarization with sequence-to-sequence models," *ACM Transactions on Data Science*, Vol. 2, 2021, pp. 1-37.
16. C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv Preprint*, 2020, arXiv:1910.10683.
17. R. Sarkhel, M. Keymanesh, A. Nandi, and S. Parthasarathy, "Transfer learning for abstractive summarization at controllable budgets," *arXiv Preprint*, 2020, arXiv:2002.07845.
18. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv Preprint*, 2018, arXiv:1810.04805.
19. Y. Liu and M. Lapata, "Text summarization with pretrained encoders," *arXiv Preprint*, 2019, arXiv:1908.08345.
20. J. Xu, Z. Gan, Y. Cheng, and J. Liu, "Discourse-aware neural extractive text summarization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 5021-5031.
21. S. Sotudeh, A. Cohan, and N. Goharian, "On generating extended summaries of long documents," *arXiv Preprint*, 2021, arXiv:2012.14136.
22. S. Sotudeh and N. Goharian, "TSTR: Too short to represent, summarize with details! intro-guided extended summary generation," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022, pp. 325-335.
23. M. Zhong, P. Liu, Y. Chen, D. Wang, X. Qiu, and X. Huang, "Extractive summarization as text matching," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 6197-6208.
24. W. T. Visser and M. Wieling, "Sentence-based summarization of scientific documents, the design and implementation of an online available automatic summarizer," in *Citeseer*, 2005.
25. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion *et al.*, "Scikit-learn: Machine learning in python," Technical Report, Vol. 12, 2011 pp. 2825-2830.
26. S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*, O'Reilly Media, Inc., 2009.

27. R. Rehurek, "Gensim: Topic modelling for humans," *URL: https://radimrehurek.com/gensim*, 2019.
28. R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proceedings of LREC Workshop on New Challenges for NLP Frameworks*, 2010, pp. 45-50.
29. T. Wolf, L. Debut, V. Sanh *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv Preprint*, 2020, arXiv:1910.03771.
30. D. Miller, "Leveraging bert for extractive text summarization on lectures," *arXiv Preprint*, 2019, arXiv:1906.04165.
31. I. Cachola, K. Lo, A. Cohan, and D. Weld, "TLDR: Extreme summarization of scientific documents," in *Findings of the ACL: EMNLP*, 2020, pp. 4766-4777.
32. T. Scheepers, "Improving the compositionality of word embeddings," Master's thesis, Universiteit van Amsterdam, 2017.
33. V. Gupta, P. Bharti, P. Nokhiz, and H. Karnick, "SumPubMed: Summarization dataset of PubMed scientific articles," in *Proceedings of the 59th Annual Meeting of the ACL and the 11th International Joint Conference on NLP: Student Research Workshop*, 2021, pp. 292-303.
34. M. K. Chandrasekaran, M. Yasunaga, D. Radev, D. Freitag, and M.-Y. Kan, "Overview and results: Cl-scisumm shared task 2019," in *arXiv Preprint*, 2019, arXiv: 1907.09854.

Sheena Kurian K Research Scholar, Division of Computer Science and Engineering, School of Engineering, Cochin University of Science and Technology, Kerala, India.

Sheena Mathew Professor, Division of Computer Science and Engineering, School of Engineering, Cochin University of Science and Technology, Kerala, India.