

Extractive Summarization using Extended TextRank Algorithm

Ansh Vora, Rinit Jain, Aastha Shah and Sheetal Sonawane

Department of Computer Engineering, SCTR's Pune Institute of Computer Technology

Abstract

With so much information available online, it's more important than ever to have reliable tools for summarizing text quickly and accurately. In this paper, we introduce a new way to improve the popular TextRank algorithm for extractive summarization. By adding a dynamic damping factor and using Latent Dirichlet Allocation (LDA) to enhance how text is represented, our method creates more meaningful summaries. We tested it with metrics like Pyramid, METEOR, and ROUGE, and compared it to the original TextRank. The results were promising, showing that our approach produces better summaries and could be useful for real-world applications like text mining and information retrieval.

1 Introduction

There is a gigantic amount of data being generated on the internet in recent times. The amount of textual content generated in the form of legal documents, scripts, news articles, etc., is very vast. Generating concise summaries of such vast textual sources manually is a very time-consuming and tedious task to do. Hence, this gives rise to various approaches that help in generating automatic summaries through ATS (Automatic Text Summarization) systems. These ATS systems help to extract important information from documents. The main aim of text summarization is to shorten the length of the source text while preserving all important sentences and the overall meaning of the source text. Text summarization is divided into two types, mainly single-document summarization and multi-document summarization as mentioned in (Jalil et al., 2021) and (Tas and Kiyani, 2017). There are three types of approaches that can be followed for generating ATS systems as mentioned in (El-Kassas et al., 2020). These approaches Extractive Summarization, Abstractive Summarization and Hybrid Summarization. The extractive method

aims to extract the most important sentences from the input document. Abstractive method on the other hand aims at generating sentences that differ from the original sentences in the input text and generate summary that conveys the entire meaning of the document. The above mentioned paper also explains the different methodologies that can be followed to model the ATS systems. Extractive ATS systems can be modelled by various methods like Statistical based, Concept based, Graph based, Semantic based, etc methods. Abstractive ATS systems are modelled by using Graph based, Tree based, Ontology based, etc methods including BERT summarization which outperforms TextRank summarization as seen in (Harinatha et al., 2021)

Extractive summarizers are less complex and they preserve the fidelity of the original document. They are not prone to errors related to text generation and misinterpretation as they directly extract the highest-ranked sentences. (Sonawane et al., 2019) gives a survey of extractive graph-based summarization and states how it outperforms other approaches. (Mihalcea and Tarau, 2004) is a graph-based method for extractive text summarization that aligns with PageRank's ranking algorithm (Brin and Page, 1998). The advantage of the TextRank algorithm is that it is an unsupervised method and does not require any training to generate the summary. Along with being unsupervised, TextRank is also a language-independent method that generates a summary based on the occurrences of words. Aligning with this graph-based method of text summarization, our proposed model extends the TextRank algorithm by integrating LDA topic modeling ((Issam et al., 2021) and (Jelodar et al., 2018)) and a dynamic damping factor as parameters to enhance the vector embeddings of the nodes of the graph and shows the results obtained by our model using various evaluation metrics like ROUGE-1, ROUGE-2, and PYRAMID that evaluate the scores based on measures like recall, preci-

sion, and F1 score.

2 Literature Survey

Given the large corpus of data being generated daily, it has become increasingly important to develop automatic text summarization models that can produce precise and concise summaries. (Mihalcea and Tarau, 2004) initially proposed a graph-based ranking algorithm, TextRank, which uses an unsupervised method for sentence extraction. This technique is an extension of Google’s renowned PageRank algorithm (Brin and Page, 1998). One key advantage of this state-of-the-art method is its language independence, making it easily adaptable to different domains, genres, and languages. Building on this, (Gulati et al., 2023) made further advancements by analyzing 75 articles extracted from the Medium site. In their study, they combined a similarity matrix generated through BM25+ with the original TextRank, normalizing the matrix by dividing it by the maximum similarity score in a given matrix. Their primary focus was to enhance the mean F-score for all ROUGE metrics compared to both the original TextRank algorithm and BM25+. Their results showed an improvement of 1.654% and 0.413%, respectively.

(Jelodar et al., 2018) suggested that LDA is an unsupervised generative probabilistic model for corpus modeling. They introduced various LDA parameters such as Gibbs sampling, Expectation-Maximization (EM), and Variational Bayes inference (VB). (Onah et al., 2023) evaluated the extractive summarization method and the efficacy of LDA using Latent Semantic Analysis (LSA) and ROUGE metrics to assess its accuracy and reliability. (Gialitsis et al., 2019) focuses on enhancing extractive text summarization by using topic-based sentence representation. (Erkan and Radev, 2004) used lexical centrality to determine sentence salience within a document.

(Shi et al., 2019) proposed DeepChannel, an attention-based neural network that achieved robust results using only 1 % of the CNN/DailyMail dataset. (Niepert et al., 2016) and (Jin et al., 2020) introduced CNN based approaches to achieve high accuracy against benchmark datasets. (Basyal and Sanghvi, 2023) focuses on text summarization methods using various LLM models. (Syed et al., 2020) used the Webis-EditorialSum-2020 dataset, containing 1,330 summaries for 266 articles. Their approach involved two models for generating ex-

tractive summaries. The first was TextRank, which employed two similarity functions: lexical overlap and common named entities. The second model used BERT-based embeddings and clustered segments using K-Means, with two variants: ExtSum-XLNet and ExtSum-DistilBERT (Yang et al., 2020). (Liu et al., 2024) contributed to the field by employing the SumSurvey dataset, a long-document abstractive summarization dataset composed of scientific survey papers. Well-known evaluation metrics such as ROUGE and BERT Score were used as reference-based measures in extractive summarization. Additionally, their abstractive summarization made use of UniEval, with large language models (ChatGPT, Vicuna) serving as reference-based metrics. They also applied reference-free metrics, such as Novel N-Gram, to measure the abstractiveness of the summaries. In this research, we also use Pyramid scores to evaluate the generated text summaries, as noted in (Gao et al., 2019). This paper introduces PyrEval, which outperforms previously developed automated pyramid methods and sets a new benchmark for precision and accuracy in text summarization evaluation.

3 Methodology

3.1 Data Preprocessing

Data preprocessing is the initial stage in this summarization model and is used to make the documents relevant and targetable using the machine learning algorithms. We import the dataset of article-reference summary pairs and carry out the pre-processing phase on the text corpus of the article. The following steps are taken in the Data Preprocessing Phase as referred in Figure 1:

1. **Sentence Tokenization** – The entire text corpus is composed of several sentences that form the article. LDA and TextRank work at a sentence level by creating summaries using the top ranked sentences. Thus, the first step of processing involves sentence tokenization of the text corpus.
2. **Word Tokenization** – Each of these individual sentences is then broken down into words thereby increasing the amount of analysis that can be carried out, such as using TextRank to calculate word overlaps and using LDA to create a document-term matrix where each topic is represented by a number of frequently co-occurring words. This function breaks

down each sentence into tokens and removes the punctuations while expanding the contractions.

3. **Removal of Stopwords** – Stopwords are simple English words used in the article for coherence and fluency but do not contribute significantly to the importance and meaning of a sentence. Stopwords include words like 'is', 'in', 'this', and so on.

3.2 LDA for Topic Modelling

LDA is a generative probabilistic statistical model that aims to discover latent topics within a document. It assigns a set of topics to each sentence based on the words it contains, where each topic is identified by a group of words that occur frequently together. LDA is used to find the topical distribution of the document and identifies the topics or themes encapsulated by the text by looking for groups of words that frequently occur together across sentences and modelling them into a topic.

Before LDA modelling, preprocessing of the raw text is performed to convert it into a numerical representation known as a document-term matrix that captures the frequency of words in every sentence. The columns of the document-term matrix correspond to unique words, and the rows correspond to sentences.

LDA Modelling calculates the word-topic distributions based on how likely a word is to belong to a topic given the other words in the sentence and determines how much each topic contributes to the overall content of a sentence as well as the importance of the sentence in the entire corpus of text.

The LDA model creates a Topic Matrix that maps sentences to the topics. Each row corresponds to a sentence, and each column corresponds to a topic. Thus, each sentence is mapped to various topics it encapsulates. The values in the matrix represent the probability that the sentence belongs to a particular topic.

Sentences that have higher topic scores are more likely to be about the key topics of the document and thus are more likely to be included in the final summary. More weight is to be given to sentences that are central to the main ideas or topics, improving the quality and coherence of the generated summary.

3.3 TextRank

TextRank is a graph-based algorithm used for extractive summarization of text. It is a sentence-based approach that computes the similarity between sentences based on word overlap. In the TextRank algorithm, each sentence represents a node in a graph, and the edges between the nodes represent the similarity score between the sentences. A similarity matrix stores the similarity scores between all sentences and is used for summarization.

The similarity score between two sentences, A and B, is calculated as:

$$\text{Similarity}(A, B) = \frac{|A \cap B|}{\log(\text{len}(A) + 1) + \log(\text{len}(B) + 1)}$$

After computing the similarity matrix, the PageRank algorithm is applied to rank sentences according to their centrality, based on the assumption that the more overlaps a sentence has, the greater its importance. The algorithm updates the rank (importance) of each sentence based on its connections to other sentences.

The PageRank algorithm is expressed as:

$$PR(i) = \frac{1 - \alpha}{N} + \alpha \sum_j W_{ji} \cdot PR(j)$$

Where α is the damping factor, ideally set to 0.85.

After ranking the sentences, we selected the length of the summary to be generated in terms of the number of sentences (n), where the top-ranked n sentences are joined to form the required summary.

3.4 Damping Factor

In the PageRank formula, α is the damping factor used to control how the algorithm navigates through the connected nodes. With probability α , the rank of a sentence is determined by its similarity to other sentences (i.e., the structure of the graph), and with probability $1 - \alpha$, the algorithm allows random jumps to any other sentence, preventing rank accumulation from being too dependent on just a few influential nodes. With a damping factor of 0.85, the algorithm jumps to a connected node with 85% probability but, in 15% of cases, makes random jumps to avoid a thread of sentences monopolizing the summary.

A higher damping factor gives more weight to sentence connectivity, ensuring sentence similarity is emphasized. A lower damping factor places

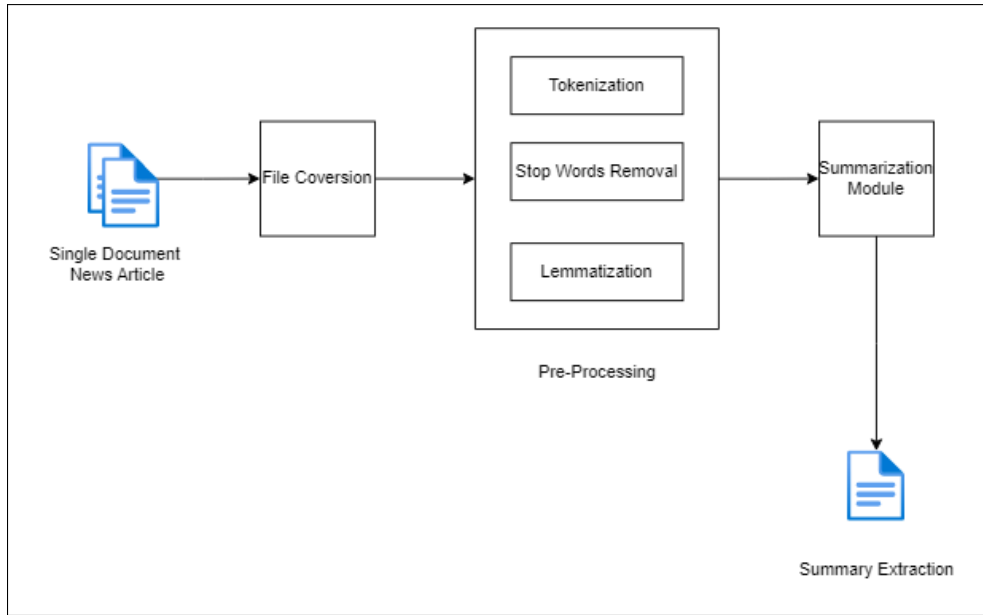


Figure 1: System Architecture Diagram

more emphasis on random jumps, making the system less reliant on sentence similarity. Hardcoding a value of 0.85 may lead to inefficient summarization in articles with lesser overlaps due to the diversified nature of content. In this model, we dynamically calculate the value for the damping factor using the following approach:

- **Node Connectivity** – A higher node connectivity factor means the sentences are very similar, positively contributing to the damping factor. Higher node connectivity indicates more coherence in the text corpus.

$$\text{Connectivity Factor} = \frac{\text{node connectivity factor}}{5}$$

- **Entropy Factor** – Entropy measures the distribution of topics throughout the document. High entropy indicates diverse topical spread and reduced similarity scores, suggesting a need to lower the damping factor.

$$\text{Entropy Adjust} = \frac{\text{entropy factor}}{5}$$

- **Variance Factor** – The variance of the integrated matrix represents values from the Topic Matrix formed by LDA modeling and the similarity score. A higher value denotes significant differences in sentence scores, indicating that some sentences are much more important than others, thus increasing the damping factor.

$$\text{Variance Factor} = \frac{\text{lda variance} + \text{textrank variance}}{2}$$

- **Length Factor** – This is based on the length of a document. If a document is too long, then more randomness in sentence selection is efficient. This factor has a lower weight and is inversely proportional to the damping factor.

$$\text{Variance Factor} = \frac{\text{lda variance} + \text{textrank variance}}{2}$$

- **Number of Topics** – This measures the diversity in the article. A higher number of topics typically correlates with reduced similarity scores due to topical spread. Thus, an increase in the number of topics should reduce the damping factor.

$$\text{Topic Factor} = \frac{50}{\text{Num topics}}$$

Thus, we calculate the dynamic damping factor as:

$$\text{Damping Factor} = 0.5 + 0.25 \times Cf - 0.35 \times EA - 0.2 \times Lf - 0.3 \times Tf + 0.15 \times Vf$$

Where:

- Cf is Connectivity factor
- EA is Entropy Adjust

- Lf is Length factor
- Tf is Topic factor
- Vf is Variance factor

Note: Weights for each factor can be calculated by iterating over from 0 to 1 with a 0.05 increment and choosing the best-performing weights. They may vary from dataset to dataset.

3.5 Integration Model

We integrate the Topic Matrix and Similarity Matrix in our model to generate more contextually relevant summaries. The weights for the LDA Matrix and Similarity Matrix are adjusted based on the document’s structure and diversity. For instance, the CNN-Daily Mail dataset performs best with an LDA weight of 0.1 and a Similarity Matrix weight of 0.9, while BBC Extractive summarization benefits from an LDA weight of 0.8. These weights are determined through variance analysis of the matrices or brute-force and trial-and-error techniques. The Integrated Matrix for ranking sentences and summarization is computed using:

$$\text{Combined Score}(i) = \text{LDA Weight} \times \text{LDA Score}(i) + \text{TextRank Weight} \times \text{Similarity Score}(i).$$

The final Combined Score leverages both topical modeling (via LDA) and word overlap (via TextRank), enabling a more balanced and efficient summary.

3.6 Computational Cost

The integrated summarization system combines TextRank and LDA-TextRank for topic-aware summaries. TextRank computes sentence importance using a similarity matrix and PageRank, with a computational complexity of $O(N^2)$, where N is the number of sentences. This involves constructing a similarity graph and iteratively calculating sentence scores.

LDA-TextRank, on the other hand, introduces Latent Dirichlet Allocation (LDA) for topic modeling, which adds significant computational overhead with a complexity of $O(I \times T \times V)$, where I is the number of iterations, T is the number of topics, and V is the vocabulary size.

Additionally, the dynamic damping factor incorporates LDA-derived metrics such as topic entropy and variance, improving summary coherence at the expense of increased computational cost.

4 Evaluation

The summary generated is evaluated against the reference summary from the dataset. Various summary evaluation methods are available, such as the ROUGE Metric (ROUGE-N and ROUGE-L Scores), Pyramid Evaluation Metric, and METEOR (Metric for Evaluation of Translation with Explicit Ordering). The evaluation is performed for multiple articles.

4.1 Dataset 1: CNN/Daily Mail

The CNN/Daily Mail dataset is an English-language dataset containing just over 300k unique news articles written by journalists at CNN and the Daily Mail. Each instance includes a string for the article, a string for the highlights, and a string for the ID. Token Count for CNN/Daily Mail Dataset can be seen in Table 1. This dataset is primarily used for abstractive summarization; therefore, the articles generated through our model may exhibit lower ROUGE scores. However, the proposed model still outperforms the general TextRank implementation across ROUGE, Pyramid, and METEOR scores.

Feature	Mean Token Count
Article	781
Highlights	56 (2 Sentences)

Table 1: Token Count for CNN/Daily Mail Dataset

4.2 Dataset 2: Webis Corpus

The Webis Editorial Sum Corpus consists of 1330 manually curated extractive summaries for 266 news editorials spanning three diverse portals: Al-Jazeera, Guardian, and Fox News. Each article has five summaries, each labeled for overall quality and fine-grained properties such as thesis relevance, persuasiveness, reasonableness, and self-containedness. The summary statistics for Webis Corpus is shown in Table 2.

Feature	Count
Number of Words in Article	913
Number of Words in Summaries	200
Average Length of Summary	20% to 22%

Table 2: Summary Statistics for Webis Corpus

4.3 Dataset 3: BBC Summary Extractive

This dataset for extractive text summarization includes 417 political news articles from BBC, covering the years 2004 to 2005. Each article is accompanied by five summaries, stored in the Summaries folder. The first clause of the text of each article serves as the respective title. The summary statistics for the same is shown in Table 3.

Feature	Count
Number of Words in Article	380
Number of Words in Summaries	160

Table 3: Summary Statistics for BBC Summary Extractive Dataset

4.4 Metrics Used for Evaluation

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a commonly used metric for comparing a reference summary (Expert summary) with the generated summary. It focuses on n-gram overlaps and other similarity measures. The variants of ROUGE are:

- **ROUGE N:** Measures the overlap of N-grams (e.g., Unigrams, Bigrams, Trigrams).
- **ROUGE L:** Measures the longest common subsequence (LCS) between the reference and generated summaries, capturing sentence-level structure similarity.
- **ROUGE S:** Measures skip-bigram overlap, allowing gaps between words in the sequence. The Algorithm for computing ROUGE-S is given in Algorithm 1

Each ROUGE metric is computed in three aspects:

- **Recall:** The proportion of overlap in the reference summary that is also in the generated summary.
- **Precision:** The proportion of overlap in the generated summary that is also in the reference summary.
- **F1-score:** The harmonic mean of Recall and Precision, calculated as:

$$F1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

Pyramids: The Pyramids Evaluation method assesses the content coverage and relevance of a summary, determining how well the essential content

is represented. Each reference summary is broken down into atomic information units (SCUs), which represent important pieces of information. The system-generated summary is evaluated by identifying which SCUs it contains, with each SCU assigned a weight based on how many reference summaries include it.

METEOR Scores: This metric improves on precision-recall metrics like BLEU by incorporating semantic features such as synonymy and stemming. It compares exact matches between words in the reference and generated summaries, applying stemming and matching synonyms, as well as phrase and paraphrase matching. A penalty is applied for how well-ordered and contiguous the matched words are, encouraging proper word order in summaries. The METEOR score reflects recall, precision, F1, and penalty scores.

4.4.1 Algorithm for ROUGE-S Calculation

Algorithm 1: ROUGE-S Calculation

Data: A list of words from the reference and generated summaries, and a maximum skip value

Result: Precision, Recall, and F1 score

Function

```
generate_skip_bigrams(words,
max_skip)
begin
  Initialize skip_bigrams
  for each word i in words do
    for each word j within max_skip of
      i do
        Add (words[i], words[j]) to
          skip_bigrams
  return skip_bigrams
```

Function

```
compute_skip_bigram_score(ref_summary,
gen_summary, max_skip)
```

begin

```
  Tokenize ref_summary, gen_summary
  Set ref_bigrams as skip-bigrams from
    ref_summary
  Set gen_bigrams as skip-bigrams from
    gen_summary
  Set overlap as intersection of
    ref_bigrams, gen_bigrams
  if no bigrams then
    return (0, 0, 0)
  Compute precision, recall, f1
  return (precision, recall, f1)
```

4.5 Result Analysis

The evaluation of the models was performed using various ROUGE metrics across different dataset splits. Table 4 demonstrates that the Integrated Model surpasses the TextRank model in terms of ROUGE Recall and F1 scores, while exhibiting lower precision. The reduction in precision is attributed to the incorporation of LDA, which enhances topic coverage but leads to the inclusion of less informative sentences, thereby sacrificing precision for improved recall. The given can be easily visualized using Figure 2 and Figure 3.

Table 5 and Table 7 contains the Pyramid and METEOR metrics for the TextRank and Integrated Models of The Webis Corpus and BBC Summarization dataset which are Extractive datasets. Figure 4 visualizes the Pyramid and METEOR Scores for CCN Daily Mail Datasets.

Table 6 contains the various evaluation metrics for the TextRank and Integrated Models of The BBC Summarization dataset which is an Extractive dataset. Figure ?? visualizes various ROUGE metrics for the BBC Dataset.

Table 8 shows comparison between the proposed model and various LLM models. The proposed model demonstrates a significant increase in recall across all other listed models, accompanied by improved F1 scores, indicating enhanced overall performance.

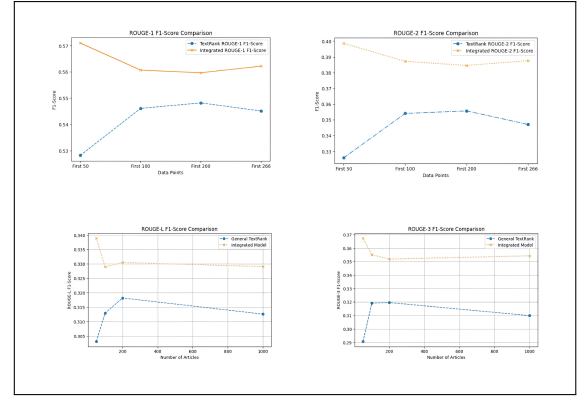


Figure 2: ROUGE Scores for Webis Corpus Dataset

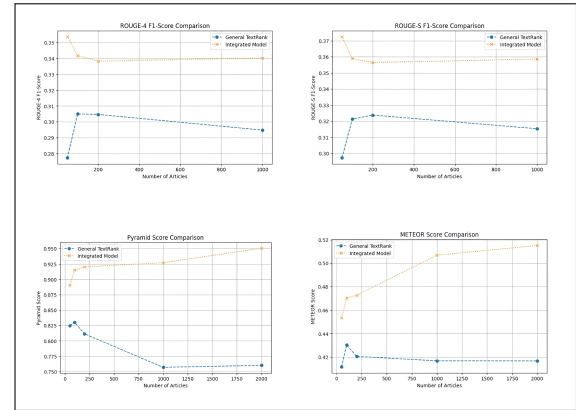


Figure 3: ROUGE-4, ROUGE-S, Pyramid and METEOR Scores for Webis Corpus Dataset

Articles	TextRank Model		Integrated Model	
	Pyramid Score	METEOR Score	Pyramid Score	METEOR Score
50	0.8126	0.4018	0.9186	0.4750
100	0.8367	0.4194	0.9130	0.4718
200	0.8364	0.4212	0.9175	0.4708
266	0.8321	0.4158	0.9204	0.4748

Table 5: Pyramid and METEOR Scores on Webis Corpus Dataset

Articles	Metric	Recall	Precision	F1	Integrated Model Scores					
					Articles	Metric	Recall	Precision	F1	
50	TextRank Scores					50	ROUGE-1	0.6519	0.5183	0.5710
	ROUGE-2	0.3368	0.3247	0.3259	ROUGE-2		0.4511	0.3639	0.3986	
	ROUGE-L	0.3143	0.3015	0.3031	ROUGE-L		0.3877	0.3070	0.3389	
	ROUGE-3	0.2860	0.2766	0.2773	ROUGE-3		0.4150	0.3355	0.3673	
	ROUGE-4	0.2999	0.2899	0.2907	ROUGE-4		0.3997	0.3231	0.3537	
100	ROUGE-S	0.3148	0.2902	0.2972	ROUGE-S	0.4336	0.3333	0.3725		
	ROUGE-1	0.5723	0.5373	0.5461	100	ROUGE-1	0.6548	0.5006	0.5607	
	ROUGE-2	0.3703	0.3489	0.3541	ROUGE-2	0.4473	0.3482	0.3872		
	ROUGE-L	0.3288	0.3071	0.3129	ROUGE-L	0.3848	0.2931	0.3290		
	ROUGE-3	0.3183	0.3009	0.3050	ROUGE-3	0.4093	0.3198	0.3551		
200	ROUGE-4	0.3334	0.3149	0.3192	ROUGE-4	0.3940	0.3079	0.3418		
	ROUGE-S	0.3444	0.3101	0.3213	ROUGE-S	0.4261	0.3163	0.3589		
	ROUGE-1	0.5724	0.5397	0.5482	200	ROUGE-1	0.6512	0.5014	0.5597	
	ROUGE-2	0.3709	0.3507	0.3557	ROUGE-2	0.4423	0.3473	0.3846		
	ROUGE-L	0.3325	0.3132	0.3182	ROUGE-L	0.3851	0.2956	0.3305		
266	ROUGE-3	0.3174	0.3008	0.3047	ROUGE-3	0.4035	0.3183	0.3519		
	ROUGE-4	0.3330	0.3153	0.3196	ROUGE-4	0.3879	0.3061	0.3383		
	ROUGE-S	0.3457	0.3129	0.3238	ROUGE-S	0.4208	0.3159	0.3565		
	ROUGE-1	0.5681	0.5372	0.5451	266	ROUGE-1	0.6544	0.5032	0.5622	
	ROUGE-2	0.3612	0.3426	0.3471	ROUGE-2	0.4458	0.3497	0.3876		
	ROUGE-L	0.3263	0.3077	0.3126	ROUGE-L	0.3834	0.2942	0.3291		
	ROUGE-3	0.3064	0.2912	0.2948	ROUGE-3	0.4065	0.3204	0.3543		
	ROUGE-4	0.3222	0.3060	0.3099	ROUGE-4	0.3903	0.3077	0.3403		
	ROUGE-S	0.3358	0.3052	0.3153	ROUGE-S	0.4232	0.3177	0.3586		

Table 4: ROUGE Scores Comparison Between TextRank and Integrated Model on Webis Corpus Dataset.

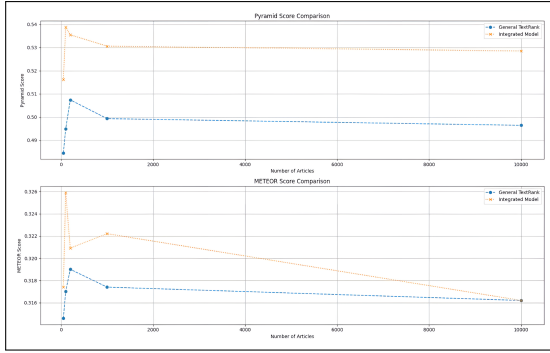


Figure 4: Pyramid and METEOR Scores for CNN Daily Mail dataset

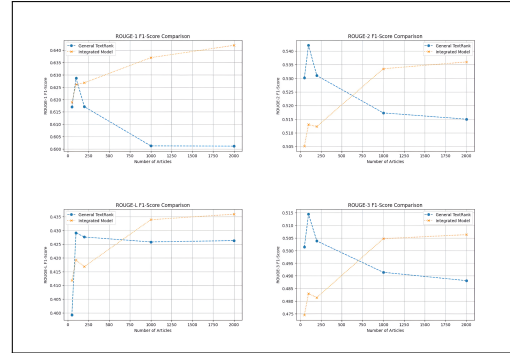


Figure 5: Various ROUGE Metrics for BBC dataset

Articles	Metric	Recall	Precision	F1-Score	Articles	Metric	Recall	Precision	F1-Score
ROUGE Scores for TextRank on BBC Dataset					Integrated Scores for BBC Dataset				
50	ROUGE-1	0.4968	0.8477	0.6170	50	ROUGE-1	0.5337	0.7676	0.6189
	ROUGE-2	0.4253	0.7354	0.5302		ROUGE-2	0.4359	0.6258	0.5051
	ROUGE-L	0.3198	0.5585	0.3992		ROUGE-L	0.3550	0.5106	0.4118
	ROUGE-4	0.3840	0.6719	0.4801		ROUGE-4	0.3927	0.5656	0.4555
	ROUGE-3	0.4015	0.6987	0.5014		ROUGE-3	0.4094	0.5884	0.4745
ROUGE-S	0.4008	0.6428	0.4853	ROUGE-S	0.4151	0.5477	0.4643		
100	ROUGE-1	0.5102	0.8508	0.6287	100	ROUGE-1	0.5616	0.7572	0.6261
	ROUGE-2	0.4385	0.7400	0.5421		ROUGE-2	0.4624	0.6185	0.5130
	ROUGE-L	0.3466	0.5900	0.4291		ROUGE-L	0.3782	0.5045	0.4192
	ROUGE-4	0.3983	0.6801	0.4938		ROUGE-4	0.4183	0.5594	0.4637
	ROUGE-3	0.4155	0.7054	0.5144		ROUGE-3	0.4358	0.5820	0.4830
ROUGE-S	0.4150	0.6472	0.4978	ROUGE-S	0.4365	0.5402	0.4683		
200	ROUGE-1	0.4982	0.8431	0.6171	200	ROUGE-1	0.5659	0.7469	0.6268
	ROUGE-2	0.4268	0.7326	0.5310		ROUGE-2	0.4640	0.6098	0.5123
	ROUGE-L	0.3431	0.5939	0.4276		ROUGE-L	0.3779	0.4952	0.4168
	ROUGE-4	0.3878	0.6744	0.4842		ROUGE-4	0.4188	0.5506	0.4621
	ROUGE-3	0.4041	0.6985	0.5038		ROUGE-3	0.4363	0.5729	0.4814
ROUGE-S	0.4059	0.6458	0.4908	ROUGE-S	0.4417	0.5382	0.4720		
1000	ROUGE-1	0.4875	0.8197	0.6012	1000	ROUGE-1	0.6001	0.7284	0.6370
	ROUGE-2	0.4178	0.7125	0.5173		ROUGE-2	0.5047	0.6082	0.5335
	ROUGE-L	0.3426	0.5913	0.4258		ROUGE-L	0.4121	0.4922	0.4339
	ROUGE-4	0.3799	0.6584	0.4725		ROUGE-4	0.4596	0.5534	0.4852
	ROUGE-3	0.3960	0.6810	0.4914		ROUGE-3	0.4779	0.5752	0.5047
ROUGE-S	0.3987	0.6347	0.4808	ROUGE-S	0.4807	0.5430	0.4941		
2000	ROUGE-1	0.4878	0.8194	0.6011	2000	ROUGE-1	0.6135	0.7216	0.6420
	ROUGE-2	0.4163	0.7088	0.5150		ROUGE-2	0.5140	0.6004	0.5360
	ROUGE-L	0.3433	0.5918	0.4263		ROUGE-L	0.4192	0.4869	0.4359
	ROUGE-4	0.3778	0.6532	0.4691		ROUGE-4	0.4673	0.5453	0.4867
	ROUGE-3	0.3938	0.6759	0.4881		ROUGE-3	0.4860	0.5669	0.5063
ROUGE-S	0.3969	0.6275	0.4772	ROUGE-S	0.4890	0.5334	0.4944		

Table 6: ROUGE Scores Comparison Between TextRank and Integrated Model on BBC Dataset

Articles	TextRank Model		Integrated Model	
	Pyramid Score	METEOR Score	Pyramid Score	METEOR Score
50	0.8248	0.4117	0.8906	0.4533
100	0.8301	0.4303	0.9153	0.4703
200	0.8114	0.4206	0.9204	0.4727
1000	0.7570	0.4169	0.9271	0.5068
2000	0.7602	0.4168	0.9506	0.5151

Table 7: Pyramid and METEOR Scores Comparison Between TextRank and Integrated Model on BBC Dataset

Model	Metric	Recall	Precision	F1-Score
Integrated Model	ROUGE-1	0.4698	0.1797	0.2575
	ROUGE-2	0.1825	0.0675	0.0973
	ROUGE-L	0.2989	0.1122	0.1615
Falcon-7b-instruct	ROUGE-1	0.3002	0.1945	0.2270
	ROUGE-2	0.0786	0.0442	0.0538
	ROUGE-L	0.2635	0.1688	0.1980
MPT-7b-instruct	ROUGE-1	0.3213	0.1991	0.2370
	ROUGE-2	0.0883	0.0490	0.0604
	ROUGE-L	0.2906	0.1787	0.2132
OpenAI ChatGPT	ROUGE-1	0.4246	0.1928	0.2616
	ROUGE-2	0.1661	0.0618	0.0884
	ROUGE-L	0.3885	0.1761	0.2390

Table 8: ROUGE Scores Comparison Across Models

5 Conclusion

We explored a new approach to text summarization by integrating TextRank with Latent Dirichlet Allocation (LDA) to perform Topical Modelling and using a dynamically calculated damping factor. This approach demonstrated significant improvements over traditional TextRank method. In this model we are using the semantic information extracted through LDA topic modelling to capture the themes within a document. From the results summarized in the earlier section, it is clear that the proposed model works well with multi-sentence summaries (around 20% of Original length). When we tried the model to generate smaller summaries (2 or 3 sentences long), the results were inferior to general textrank due topic relevance shifts are influenced by the damping factor. Longer documents tend to have better results due to better captured topical coherence through LDA.

The dynamic damping factor can be adapted for abstractive summarization by guiding content selection, attention mechanisms, and counter decoder biases. In transformer-based models, it can scale attention scores, prioritize context-based more relevant input, and refine reward function during fine-tuning (if using reinforcement learning). This enhances topic relevance and coherence in abstractive summaries.

The dynamic damping factor adapts to the nature of the input text and ensures a balanced representation of all important concepts. We can also try various embedding methods to further improve the accuracy of summarization.

References

- Lochan Basyal and Mihir Sanghvi. 2023. [Text summarization using large language models: A comparative study of mpt-7b-instruct, falcon-7b-instruct, and openai chat-gpt models.](#)
- Sergey Brin and Lawrence Page. 1998. [The anatomy of a large-scale hypertextual web search engine.](#) *Computer Networks*, 30:107–117.
- Wafaa S. El-Kassas, Cherif Salama, Ahmed Rafea, and Hoda Mohamed. 2020. [Automatic text summarization: A comprehensive survey.](#) *Expert Systems with Applications*, 165:113679.
- G. Erkan and D. R. Radev. 2004. [Lexrank: Graph-based lexical centrality as salience in text summarization.](#) *Journal of Artificial Intelligence Research*, 22:457–479.
- Yanjun Gao, Chen Sun, and Rebecca J. Passonneau. 2019. [Automated pyramid summarization evaluation.](#) In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 404–418, Hong Kong, China. Association for Computational Linguistics.
- Nikolaos Gialitsis, Nikiforos Pittaras, and Panagiotis Stamatopoulos. 2019. [A topic-based sentence representation for extractive text summarization.](#) In *Proceedings of the Workshop MultiLing 2019: Summarization Across Languages, Genres and Sources*, pages 26–34, Varna, Bulgaria. INCOMA Ltd.
- Vaibhav Gulati, Deepika Kumar, Daniela Elena Popescu, and Jude D. Hemanth. 2023. [Extractive article summarization using integrated textrank and bm25+ algorithm.](#) *Electronics*, 12(2).
- Sreeya Reddy Kotrakona Harinatha, Beauty Tatenda Tasara, and Nunung Nurul Qomariyah. 2021. [Evaluating extractive summarization techniques on news articles.](#) In *2021 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pages 88–94.
- Kalliath Abdul Rasheed Issam, Shivam Patel, and Subalalitha C. N. 2021. [Topic modeling based extractive text summarization.](#) *arXiv preprint arXiv:2106.15313*.
- Zakia Jalil, Jamal Abdul Nasir, and Muhammad Nasir. 2021. [Extractive multi-document summarization: A review of progress in the last decade.](#) *IEEE Access*, 9:130928–130946.
- Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. 2018. [Latent dirichlet allocation \(lda\) and topic modeling: models, applications, a survey.](#)
- Hanqi Jin, Tian ming Wang, and Xiaojun Wan. 2020. [Semsum: Semantic dependency guided neural abstractive summarization.](#) In *AAAI Conference on Artificial Intelligence*.
- Ran Liu, Ming Liu, Min Yu, He Zhang, Jianguo Jiang, Gang Li, and Weiqing Huang. 2024. [SumSurvey: An abstractive dataset of scientific survey papers for long document summarization.](#) In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9632–9651, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.
- Rada Mihalcea and Paul Tarau. 2004. [TextRank: Bringing order into text.](#) In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.
- Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. [Learning convolutional neural networks for graphs.](#) *CoRR*, abs/1605.05273.

- Daniel F. O. Onah, Elaine L. L. Pang, and Mahmoud El-Haj. 2023. [A data-driven latent semantic analysis for automatic text summarization using lda topic modelling](#).
- Jiaxin Shi, Chen Liang, Lei Hou, Juanzi Li, Zhiyuan Liu, and Hanwang Zhang. 2019. [Deepchannel: Saliency estimation by contrastive learning for extractive document summarization](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:6999–7006.
- Sheetal Sonawane, Parag Kulkarni, Charusheela Deshpande, and Bhagyashree Athawale. 2019. [Extractive summarization using semigraph \(essg\)](#). *Evolving Systems*, 10.
- Shahbaz Syed, Roxanne El Baff, Johannes Kiesel, Khalid Al Khatib, Benno Stein, and Martin Potthast. 2020. [News editorials: Towards summarizing long argumentative texts](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5384–5396, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Oguzhan Tas and Farzad Kiyani. 2017. [A survey automatic text summarization](#). *PressAcademia Procedia*, 5(1):205–213.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2020. [Xlnet: Generalized autoregressive pretraining for language understanding](#).