Harnessing Deep Learning for Effective Extractive Text Summarization: A Comparative Study

Deema mohammed alsekait
Department of Computer Science and Information Technology, Applied College , Princess Nourah Bint Abdulrahman University, Saudi Arabia.
Email: dmalsekait@pnu.edu.sa

Waref Almanaseer
Faculty of Information Technology , Applied Science Private University, Amman 11931, Jordan
Email: w_manaseer@asu.edu.jo

Ibrahim A Gomaa , Magdy Abd-Elghany Zeid
Computer Science Department, Alobour high institute for computer and informatics, Cairo, Egypt
Emails : ibraheemg , magdy_zeid83 @oi.edu.eg

Salma Baligh , Hana Alaa , Jasmine Hegazy, Julia Magdy
Faculty of Computer Science, Misr International University, Cairo, Egypt
Email: salma2005893,hana2000171 , jasmine2004084, Julia2000457 @miuegypt.edu.eg

Alaa Tariq Eldmrat
Faculty of Human Sciences Midocean University
Email : Alaadmrah@gmail.com

Diaa Salama AbdElminaam
Jadara Research Center, Jadara University, Irbid, 21110, Jordan
Faculty of Computers , Misr International University, Egypt.
MEU Research Unit, Middle East University, Amman 11831, Jordan
Email :diaa.salama @miuegypt.edu.eg

2024 Intelligent Methods, Systems, and Applications (IMSA) | 979-8-3503-6263-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/IMSA61967.2024.10652647

***Abstract*—Text Summarization might seem like a trivial problem, but in reality, is essential for data acquisition and understanding by compressing large amounts of text into specific a nd t o-the-point s ummaries. T his research is conducted to help compare the proposed methods' applications and make a summary system accessible to applications such as information retrieval, legal documents summarization and much more. from this, we intend to start by acquiring the dataset, preprocessing and training, and testing the models. In this paper will explore the methods that can be applied on summarization. These methods are Machine learning models like the random forest, naive Bayes , SVM, logistic regression, Deep learning like a neural networks, Transformers-based models like T5, and performance metrics like ROUGE. By trying all these various methods and models on a news article dataset we are going to compare the results from each of them and get the best performance. Finally, the results would be a comparison between all outputs and found that Natural Language Processor .In conclusion, Text Summarization is best done using NLP , and challenges in this project would be due to the diversity of text and context.**

Keywords: Text Summarization, Machine Learning ,Deep Learning, T5, ROUGE, Extractive.

I. INTRODUCTION

Text summarization is a challenging natural language processing (NLP) problem that involves summarizing large amounts of information from a particular document while preserving essential meaning. The need for text summarization arises from the vast amount of text data available in various fields, such as news articles, research papers, and legal documents. In a world full of information, the ability to efficiently extract important insights from large texts is paramount. Generally, there are two types of abstraction: extraction and abstraction. Extractive summarization involves selecting existing sentences and phrases from the original text and rearranging them to create a concise summary. Abstract summarization, on the other hand, generates new sentences that capture the central meaning of the text. Both approaches have their own challenges: extractive summarization requires identifying important information and ensuring consistency, whereas abstract summarization provides a deep understanding of the content and summarization that is human and contextual must be created. Researchers and practitioners in this field are using advanced NLP techniques such as machine learning algorithms and deep neural networks to improve the performance of their summarization models. Evaluation of summary quality includes metrics such as his ROUGE (Recall-Oriented Understudy for Gisting Evaluation) and his BLEU (Bilingual Evaluation Understudy), which evaluate the overlap and fluidity of the generated summary with reference summaries. It will be. As the digital environment continues to expand, the need to quickly extract relevant information from an ever-growing sea of textual data is expected to increase the demand for effective text summarization methods. Along this paper, extractive summary is to be used implementing several deep and machine learning methods. This paper is divided into 5 sections, related works, proposed methodology, results, and conclusion.

II. Related Work

The exploration of text summarization is a well-established field, with numerous data scientists contributing to the development of highly accurate outcomes. In our research, we will provide a concise overview of select papers that we have examined and assessed, with proper citations included in the references section[1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11].

Duy Duc An Bui PhD, Guilherme Del Fiol MD, PhD, John F. Hurdle MD, PhD and Siddhartha Jonnalagadda PhD [12] developed a text summarization system to improve data extraction from full-text scientific publications.The system uses machine learning and natural language processing to automatically generate summaries of scientific publications and compare them with human-written summaries. The results showed that computer-generated sentence-level summaries covered more information and had a higher density of related sentences than human-written summaries. An ensemble approach combining rule-based, concept-based, and dictionary-based methods performed better at the fragment level. This system suggests that computer-generated summaries are a potential alternative source for data extraction when creating systematic reviews. Systematic reviews (SRs) are an important source of information for health care providers, researchers, and policy makers. Its goal is to identify, evaluate, and synthesize the best available evidence to provide reliable answers to research questions. The Cochrane Collaboration, an internationally recognized non-profit organization, develops SR for health-related topics. However, SR development has been criticized for being resource-intensive and time-consuming. Data extraction is an important step in SR development. However, manual data extraction is often prone to errors due to human factors such as limited time and resources, inconsistencies, and errors caused by boredom. Computational techniques have been proposed as potential solutions to improve productivity and reduce errors in SR data extraction. Machine learning approaches have been used to classify sentences containing PICO (Population, Intervention, Control, Outcome) elements, but these methods are insufficient to extract SR information. ExaCT, a successful full-text clinical item extraction system, uses a machine learning classifier to select the five most relevant sentences for each item, then uses handcrafted weak extraction rules to extract the value of each item. Collect. This study investigates an automated extraction text summarization system that collects relevant data from full-text publications to support the development of systematic reviews.This system aims to reduce text while preserving the most important information.The research design he consisted of three main parts. Developing a gold standard for data extraction from Cochrane reviews, developing a computer system that can automatically generate sentence- and fragment-level summaries, and evaluating system performance in summarizing and comparing clinical trial data elements.

III. Proposed Methodology

In this study, we utilized the BBC News Summary dataset, which includes news articles and their corresponding human-generated summaries, covering various topics such as politics, business, entertainment, sports, and technology. The methodology comprised several key phases. Initially, we performed text preprocessing, which involved converting text to lowercase, removing special characters and numbers, tokenizing the text, and removing stopwords to ensure uniformity and relevance. Following preprocessing, we applied Term Frequency-Inverse Document Frequency (TF-IDF) vectorization to transform the textual data into numerical format suitable for machine learning algorithms. We then trained and evaluated multiple machine learning models, including Random Forest, Naive Bayes, Support Vector Machine (SVM), and Logistic Regression, using metrics such as accuracy, F1-score, precision, and recall. Additionally, we employed deep learning techniques by fine-tuning the T5 Transformer model for abstractive summarization. The performance of the models was assessed using evaluation metrics like ROUGE and BLEU scores, providing a comprehensive analysis of their effectiveness in text summarization tasks.

A. Dataset Description

Our dataset is the BBC News Summary dataset. It is a collection of news articles and corresponding human-generated summaries, designed for text summarization tasks. The dataset encompasses a diverse range of news topics, including but not limited to politics, business, entertainment, sports, and technology. It serves as a valuable resource for researchers and practitioners interested in developing and evaluating text summarization algorithms. The dataset comprises news articles originally published by the BBC, a reputable news organization. The articles cover a broad spectrum of events and developments, providing a rich and varied source for summarization tasks. The dataset contains a substantial number of news articles, each covering a specific event or topic. These articles vary in length and complexity, reflecting the diversity of news content encountered in real-world scenarios. For each news article, there is a corresponding human-generated summary. These summaries are concise representations of the main points and key information present in the respective articles. They serve as reference summaries for evaluating the performance of automatic summarization models. The BBC News Summary dataset is suitable for various natural language processing (NLP) tasks, with a specific focus on text summarization. Researchers and practitioners can leverage this dataset to train, fine-tune, and evaluate automatic summarization models, including extractive and abstractive summarization approaches.

B. Dataset Visualization

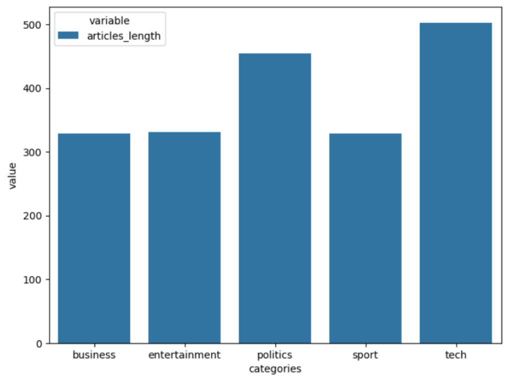

Fig. 1. Visualization of articles

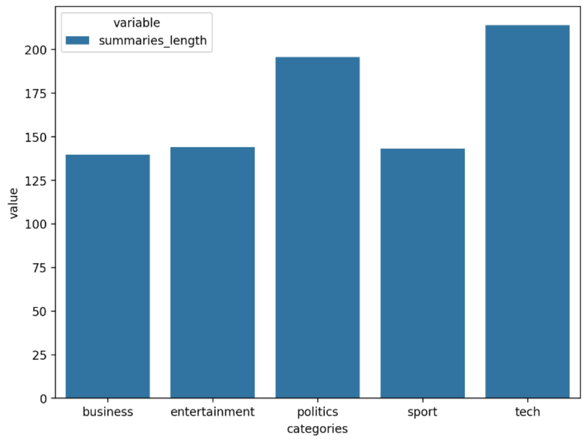

Fig. 2. Visualization of summaries

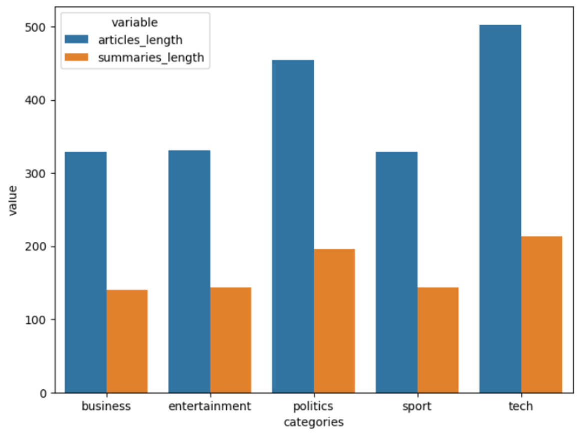

Fig. 3. Comparison between articles and summaries

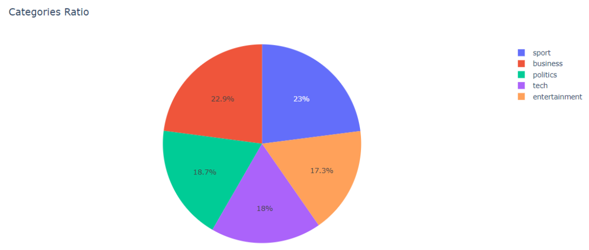

Fig. 4. Visualization of the ratio between the files

C. *Pre-Processing Phase*

Algorithm 1 Preprocess Text

```
1: function PREPROCESS_TEXT(text)
2:     text ← $lowercase$(text)
3:     text ← $remove\_special\_chars\_and\_numbers$(text)
4:     tokens ← $word\_tokenize$(text)
5:     tokens ← $remove\_stopwords$(tokens)
6:     processed_text ← $join$(tokens)
7:     return processed_text
8: end function
```

In the preprocessing phase of natural language processing (NLP), the following code implements a series of text transformations designed to enhance the quality and relevance of textual data. This code, written in Python, is encapsulated within a function named preprocess text. The primary objective of this function is to prepare raw text data for subsequent analysis and modeling. The steps involved in the preprocessing pipeline are elucidated below:

- Lowercasing: The text is converted to lowercase to ensure uniformity and to prevent the model from distinguishing words based on case, thereby improving the consistency of subsequent analyses.
- Special Character and Number Removal: Regular expression operations are employed to remove special characters and numerical digits from the text. This step aids in eliminating noise and non-essential information, focusing the analysis on the linguistic content of the text.
- Tokenization: The word tokenize function is applied to segment the text into individual words or tokens. This step is crucial for further analysis, as it breaks down the text into its fundamental components.
- Stopword Removal: Common English stopwords, such as articles, prepositions, and conjunctions, are excluded from the tokenized list. Removing stopwords helps reduce noise in the data and focuses the analysis on more meaningful words.
- Token Joining: The preprocessed tokens are then rejoined into a coherent sentence. This step facilitates the creation of a cleaned and refined representation of the original text, ready for subsequent stages of analysis or modeling.

It's important to note that the code assumes the availability of the Natural Language Toolkit (NLTK) library, as evidenced by the usage of functions like word tokenize and stopwords.words(english). Additionally, the code does not handle stemming or lemmatization, techniques commonly employed to reduce words to their root forms, which might be considered in a more comprehensive preprocessing strategy.

This preprocessing code can be employed in various NLP applications, including sentiment analysis, text classification, and information retrieval, to enhance the efficiency and accuracy of downstream tasks by ensuring that the input data is appropriately formatted and devoid of unnecessary noise.

D. *Feature Extraction Phase*

In the feature extraction phase of natural language processing (NLP), the provided code exemplifies the utilization of the Term Frequency-Inverse Document Frequency (TF-IDF) vectorization technique to transform preprocessed textual data into a numerical representation suitable for machine learning models. This code, written in Python and utilizing the scikit-learn library, is encapsulated within a snippet employing the TfidfVectorizer class. The primary goal of this phase is to convert the preprocessed textual information into a structured format that can be fed into machine learning algorithms. The steps involved in this feature extraction process are outlined below:

- TF-IDF Vectorization: The TfidfVectorizer class is instantiated, configuring the vectorizer with specific parameters. TF-IDF is a numerical statistic that reflects the importance of a term in a document relative to a collection of documents (corpus). This statistic is computed based on the term frequency (TF) and inverse document frequency (IDF). TF represents the frequency of a term within a document, while IDF reflects the rarity of the term across the entire corpus.
- Stopword Removal: The vectorizer is configured to exclude common English stopwords during the vectorization process. Stopwords, being frequently occurring words with little discriminatory power, are typically removed to enhance the relevance of the extracted features.
- Feature Dimensionality Control: The max features parameter is set to limit the number of features retained after vectorization. In this specific code, only the top 1000 features with the highest TF-IDF scores are retained. This step helps manage computational resources and may improve model performance by focusing on the most informative features.
- Training and Testing Set Transformation: The vectorizer is applied to both the training (X train preprocessed) and testing (X test preprocessed) sets separately. This ensures consistency between the feature spaces used during training and evaluation.
- Resultant Feature Matrices: The transformed feature matrices (X train vec and X test vec) now represent the original textual data in a numerical format, with each row corresponding to a document and each column corresponding to a unique feature. The values in these matrices represent the TF-IDF scores for the respective terms in each document.

This feature extraction code prepares the textual data for machine learning tasks by converting it into a structured and numerical format, allowing for the application of various models such as classifiers or clustering algorithms. The resulting TF-IDF matrices can serve as input features for training and evaluating these models, enabling the extraction of meaningful patterns and insights from the textual data.

E. Used Algorithms

Algorithm 2 Evaluate Model

```
1: function EVALUATE_MODEL(model, X_train, y_train, X_test, y_test)
2:     model.fit(X_train, y_train)
3:     predictions ← model.predict(X_test)
4:     accuracy ← accuracy_score(y_test, predictions)
5:     f1 ← f1_score(y_test, predictions, average='weighted')
6:     precision ← precision_score(y_test, predictions, average='weighted')
7:     recall ← recall_score(y_test, predictions, average='weighted')
8:     return accuracy, f1, precision, recall
9: end function
```

Four distinct machine learning algorithms, Random forest, Naive Bayes, SVM, Logistic Regression and 2 deep learning algorithm and architectures , Neural network utilizing MLP and Transformers were applied to the aforementioned dataset. And to evaluate the results from these models we used performance metrics such as accuracy, BLEU, ROUGE. Results from this will be shown in the result section.

1) **Random Forest:**

Random Forests is made up of multiple decision trees and a mix of tree predictors meaning that each tree is dependent on random values of vectors sampled each on their own and equally distributed on all the the trees in the forest.

2) Naive Bayes:

Naive Bayes is a probabilistic machine learning

Algorithm 3 Train and Evaluate Models

```
1: Define models ← {Random Forest, Naive Bayes, SVM, Logistic Regression, MLP Classifier}
2: for each model in models do
3:     accuracy, f1, precision, recall ← evaluate_model(model, X_train_vec, y_train, X_test_vec, y_test)
4:     print(f"model - Accuracy: accuracy, F1-score: f1, Precision: precision, Recall: recall")
5: end for
```

algorithm rooted in Bayes' theorem. Its simplicity and computational efficiency make it particularly advantageous for high-dimensional datasets, and it performs well even with limited training data.

Bayes' Theorem:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

3) Supprt Vector Machine(SVM) :
Support Vector Machines (SVM) is a robust and versatile supervised machine learning algorithm widely utilized for classification and regression tasks. Its strength lies in finding the optimal hyperplane that effectively separates data into distinct classes or predicts continuous outcomes. The algorithm introduces the concept of support vectors, essential data points crucial for determining the optimal decision boundary.

The SVM equation for a binary classification problem with a linear kernel can be expressed as:

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + b\right)$$

Here:

$f(x)$ is the decision function,

α_i are the Lagrange multipliers,

y_i is the class label of the training data,

$K(x, x_i)$ is the kernel function,

b is the bias term,

N is the number of support vectors.

This equation represents the decision function where the sign determines the predicted class.

4) Linear Regression:
Linear regression is a fundamental and widely used statistical technique for modeling the relationship between a dependent variable and one or more independent variables. The model seeks to find the best-fitting line represented by the linear equation. For more complex relationships involving multiple independent variables, multiple linear regression extends the equation accommodating the contributions of each variable.

For simple linear regression with one independent variable x, the equation is:

$$y = mx + b$$

For multiple linear regression with n independent variables $x_1, x_2, ..., x_n$, the equation is:

$$y = b_0 + b_1x_1 + b_2x_2 + ... + b_nx_n$$

5) Multi-Layer Perceptron (MLP):
The MLP Classifier, or Multi-Layer Perceptron Classifier, represents a powerful approach in machine learning, particularly within the realm of artificial neural networks. The architecture of this classifier is inspired by the human brain, consisting of layers of interconnected nodes that process and transform input data to make predictions.

$$a_i = \sigma\left(\sum_{j=1}^{3} w_{ij}^{(1)} x_j + b_i^{(1)}\right) \quad (1)$$

$$\hat{y}_k = \sigma\left(\sum_{i=1}^{4} w_{ik}^{(2)} a_i + b_k^{(2)}\right) \quad (2)$$

Here, a_i represents the activation of the i-th neuron in the hidden layer, x_j is the j-th input, σ is the activation function, $w_{ij}^{(1)}$ are the weights connecting inputs to hidden layer neurons, and $b_i^{(1)}$ are the biases for the hidden layer neurons.

Similarly, $\hat{y}_k$ represents the predicted output for the k-th class, $w_{ik}^{(2)}$ are the weights connecting hidden layer neurons to output neurons, and $b_k^{(2)}$ are the biases for the output layer neurons.

6) T5 Transformer:

T5, which stands for "Text-To-Text Transfer Transformer," is a transformer-based language model developed by Google. It was introduced in the paper titled "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," presented at NeurIPS 2019.

IV. Results and Analysis

We employed a diverse set of machine learning (ML) and deep learning models for both extractive and abstractive text summarization. In the realm of machine learning, we leveraged Random Forest, Naive Bayes, Support Vector Machine (SVM), and Logistic Regression. For neural networks, we utilized Multi-Layer Perceptron Classifier (MLP) and Text-To-Text Transfer Transformer (T5). The evaluations were conducted using metrics such as accuracy, Rouge, BLEU, and sentence similarity.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{3}$$

$$\text{ROUGE} = \frac{2 \times precision \times recall}{precision + recall} \tag{4}$$

$$\text{BLEU} = BP \cdot exp\left(\sum_{n=1}^{N} w_n log p_n\right) \tag{5}$$

The results collected from the machine learning models and MLP classifier are shown beneath.

TABLE I
Accuracies of the Predicted summaries using ML models

Model	Accuracy	F1-score	Precision	Recall
Random Forest	0.84	0.44	0.50	0.40
Naive Bayes	0.81	0.8	0.88	0.8
Support Vector Machine	0.93	0.68	0.80	0.60
Logistic Regression	0.79	0.80	0.88	0.80
MLP classifier	0.96	0.67	0.68	0.70

We computed the accuracies for generating abstractive summaries using both machine learning models and Multi-Layer Perceptron Classifier (MLP). The results indicate that the MLP classifier achieved the highest accuracy. MLP, employed for tasks like sentiment analysis, topic detection, and language detection, demonstrated superior performance in the context of abstractive summarization.

TABLE II
Results of Text-To-Text Transfer Transformer

ROUGE-1	ROUGE-2	ROUGE-L	BLEU	Sentence Similarity
0.3629	0.2526	0.2676	0.6800	0.8894

We incorporated the T5 transformers into our implementation, leveraging a pretrained model that randomly generated summaries. To assess its performance, we employed metrics such as ROUGE and BLEU scores, along with sentence similarities. The findings from our results suggest that utilizing sentence similarity provides the most effective approach.

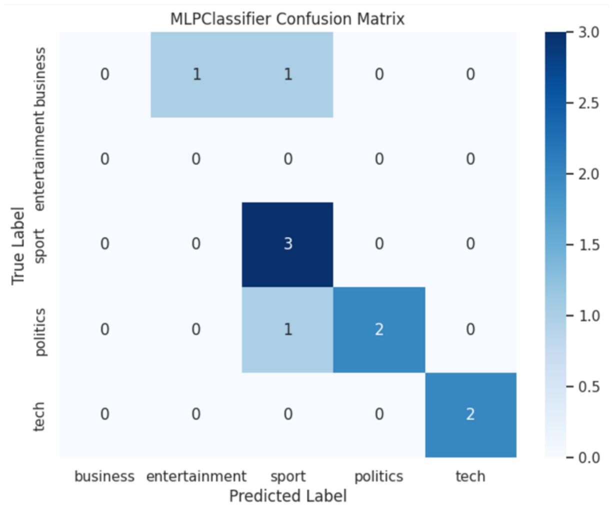

Fig. 5. Visualization of results

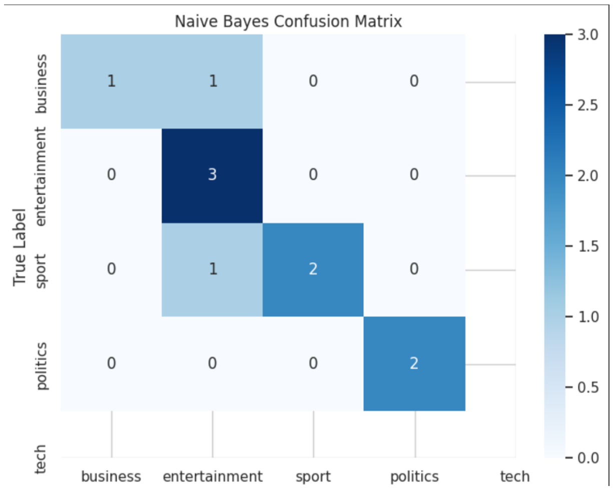

Fig. 6. Visualization of results

V. Conclusion

In conclusion, in this project we explored diverse methods for text summarization, ranging from machine learning models like Support Vector Machine to deep learning approaches such as Bidirectional and Auto-Regressive Transformers (T5). The evaluation on BBC news summaries demonstrated varying degrees of effectiveness, with Support Vector Machine and the MLP classifier excelling in different aspects.

Notably, the study highlighted the efficacy of the extractive summarization method based on sentence similarity scores, emphasizing its ability to evaluate summary quality by focusing on semantic coherence. As the digital landscape continues to grow, the demand for efficient text summarization methods is likely to increase. This research contributes valuable insights into the strengths and limitations of different models, providing a foundation for future advancements in natural language processing and information extraction.

References

[1] R. N. J. Devi Fitrianah, “Extractive text summarization for scientific journal articles using long short-term memory and gated recurrent units,” *ETS*, 2022.

[2] C. X. K. K. W. Hew Zi Jian, Olanrewaju Victor Johnson, “Text summarization for news articles by machine learning techniques,” *Applied Mathematics and Computational Intelligence*, 2022.

[3] D. R. Neeraj Kumar Sirohi, Dr. Mamta Bansal, “Text summarization approaches using machine learning lstm,” *LSTM*, 2021.

[4] C. A. A. K. Joel Larocca Neto, Alex A. Freitas, “Automatic text summarization using a machine learning approach,” *ML*, 2014.

[5] S. Gupta and S. Gupta, “Deep learning in automatic text summarization,” *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 394, 2018.

[6] M. R. S. S. S. A. S. M. V. R. Chaluvadi Abhishek, Mula Anvesh Reddy, “Automatic text summarization using deep learning and nlp model,” *Journal of Emerging Technologies and Innovative Research (JETIR)*, 2014.

[7] H. Z. Jingqiang Chen, “Abstractive text-image summarization using multi-modal attentional hierarchical rnn,” *RNN*, 2018.

[8] W. H. J. Y. Yuanyuan Li, Yuan Huang and Z. Huang, “An abstractive summarization model based on joint-attention mechanism and a priori knowledge,” *https://www.mdpi.com/journal/applsci*, 2023.

[9] T. W. Kishore Papineni, Salim Roukos and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

[10] F. Mehta, “Machine learning techniques for document summarization: A survey,” *ML*, 2016.

[11] S. N.Moratanch, “A survey on extractive text summarization,” *TS*, 2017.

[12] P. J. F. H. M. P. S. J. P. Duy Duc An Bui PhD, Guilherme Del Fiol MD, “Extractive text summarization system to aid data extraction from full text in systematic review development,” *www.elsevier.com/locate/yjbin*, 2016.