IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Text Summarization: Techniques with NLTK and Deep Learning Models Using Keras

Ruman Ashraf Bhat[1], Gurinder Kaur Sodhi [2]

[1]M. Tech Scholar, Department of Electronics and Communications Engineering, Desh Bhagat University, Punjab, India
[2] Professor, Department of Electronics and Communications, Desh Bhagat University, Punjab, India

***Abstract*:** . Text summarization plays a crucial role in natural language processing, with applications spanning from document summarization to chatbots. This thesis examines various techniques for text summarization, focusing on both extractive and abstractive methods. It starts with a thorough literature review that traces the historical development of text summarization, discussing various methodologies and evaluation metrics. Following this, the study details data collection and preprocessing methods, then moves on to the implementation of both extractive and abstractive summarization models. Extractive summarization uses techniques like word frequency scoring and sentence selection, while abstractive summarization relies on deep learning models trained on preprocessed text. A novel approach is presented, utilizing a sequence-to-sequence (seq2seq) model with an attention mechanism. This method applies deep learning, specifically Long Short-Term Memory (LSTM) networks, to produce concise summaries from input texts. The model features an encoder-decoder structure, where the encoder processes the text and the decoder creates the summary. An attention mechanism is used to enable the decoder to concentrate on relevant sections of the input. The proposed method's effectiveness is tested on a diverse dataset of texts, including reviews and articles, and evaluated using ROUGE, BLEU, and METEOR metrics. The results show that the model performs competitively in generating abstractive summaries compared to extractive methods, effectively capturing key information while producing coherent and grammatically accurate summaries

Keywords: Written text, text generation, Machine learning, NLTK, Text summarisation

I. INTRODUCTION

Text summarization is a fundamental aspect of natural language processing (NLP), crucial for condensing large amounts of content into brief, essential summaries without losing key information. It plays a significant role in managing data overload, from aggregating news articles to simplifying lengthy documents for easier understanding and efficient information retrieval. Two main methods stand out: extractive and abstractive summarization. Extractive summarization involves selecting and rearranging the most important sentences or passages from the source text using algorithms that evaluate factors like relevance and frequency. Abstractive summarization, on the other hand, creates new sentences that capture the essence of the original text, often through paraphrasing and rephrasing for greater coherence and brevity.

Early text summarization methods relied on heuristic techniques and superficial linguistic analysis, which often led to summaries that were basic and lacked coherence. However, the field evolved with the introduction of machine learning and statistical approaches, leading to more sophisticated, data-driven methods. Extractive summarization, by focusing on key sentences from the source, is valued for its simplicity and accuracy in preserving the original text. Abstractive summarization offers more flexibility and creativity but involves more complex computations.

With the advent of deep learning and transformer models like BERT, GPT, and T5, text summarization has seen significant advancements in performance and scalability. These models have pushed the boundaries of summarization quality, allowing for more nuanced and contextually aware summaries. Despite these advancements, challenges remain, such as handling multi-document summarization, incorporating domain-specific knowledge, and ensuring fairness and inclusivity in automated summaries.

The choice between extractive and abstractive summarization presents a trade-off: extractive methods preserve text fidelity but may lack coherence, while abstractive methods provide more flexibility but can struggle with factual accuracy and consistency. This ongoing challenge drives the NLP community's quest to balance fidelity and fluency in summarization systems.

With the increasing volume of digital content, automated summarization methods are becoming more essential to efficiently process and understand large datasets. Traditional manual summarization is time-consuming and may not scale effectively. Existing automatic techniques often fall short in capturing the complete and accurate essence of the original text.

This research aims to develop and assess a sequence-to-sequence model with attention mechanisms for text summarization. Utilizing deep learning techniques, the proposed model seeks to extract key information and generate coherent summaries across various text types,

including news articles, scientific papers, social media posts, and product reviews. The study will explore the impact of different hyperparameters and architectural choices on summarization performance, with the goal of identifying optimal configurations for effective summarization. The ultimate aim is to create a robust, adaptable summarization framework that helps users efficiently comprehend and digest extensive textual information.

II. LITERATURE REVIEW

McKeown and Raedev (NLP group) at Columbia University made the most significant contribution in 1995, when SUMMONS was created [1]. Extractive methods and similarity measurements were employed. McKeown et al. [2] in 1999 and Radev et al. [3] in 2000 utilized clustering to identify common themes, while Barzilay et al. 4] in 1999 created composite sentences from clusters, and Carbonell and Goldstein [5] in 1998 employed maximal marginal relevance (MMR). Evans made a significant contribution in 2005 [6], when he combined multi-document summarizing with a bilingual environment. Other pioneers have worked in this subject, with various methodologies..

Carenini et al. (2014) [7] demonstrated a robust abstractive meeting summarization method in meetings. It builds theoretical layouts using a new multi-sentence combination approach. It also used the link between source meeting transcripts and summaries to determine the ideal layouts for creating abstractive meeting summaries. The word graph algorithm is used here, and the ROUGE metric is used to evaluate it.

III. OBJECTIVES

- Develop a robust sequence-to-sequence model for text summarization.
- Implement an attention mechanism to enhance the model's ability to capture important information during summarization.
- Evaluate the effectiveness of the model in generating accurate and concise summaries compared to existing techniques.
- Investigate the impact of different hyperparameters, such as embedding dimensions and LSTM units, on the performance of the summarization model.
- Explore the applicability of the proposed model to various domains and datasets to assess its generalizability and versatility

IV. METHODOLOGY

The process of text summarization is a meticulous endeavor that involves numerous intricate steps, each contributing to the goal of distilling large volumes of text into concise and meaningful summaries.

First, data collection is essential, where a wide array of sources such as documents, articles, and web pages are systematically gathered to create a comprehensive dataset for summarization. This may involve web scraping, database queries, or manual collection from targeted sources. After gathering the raw text data, it undergoes thorough data cleaning to remove any irrelevant elements that could hinder the summarization process. This step includes eliminating HTML tags, special characters, punctuation, and other formatting issues that might distort the original content. Once cleaned, the text is segmented into individual sentences through sentence tokenization. This segmentation simplifies further analysis and processing by breaking the text into smaller, manageable units for more detailed examination.

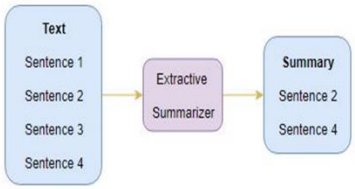

Figure 1 Summariser

Each sentence in the text undergoes word tokenization, where it is broken down into individual words. This step is crucial for detailed analysis, enabling tasks such as stemming, lemmatization, and the removal of stopwords.

Stopwords, which are common words like "and," "the," and "is," are systematically removed from the tokenized words to reduce noise and improve the relevance of the summary. This helps ensure that the summary focuses on the most important content. In extractive summarization, the next step involves calculating the word frequency of each term in the text to determine its significance. Sentences are then scored based on the cumulative frequencies of their words, with the highest-scoring sentences selected to form the summary.

In contrast, abstractive summarization, often using deep learning techniques, transforms the preprocessed text into a format suitable for model input, typically involving sequences of word embeddings that capture semantic meaning and word relationships. After training, the model generates abstractive summaries by predicting new sentences that convey the core ideas of the original text. This approach allows the creation of summaries that incorporate creativity and abstraction, going beyond simply extracting existing sentences.

Finally, the effectiveness of both extractive and abstractive summarization methods is evaluated using metrics such as ROUGE scores, which compare the generated summary to a reference summary. This evaluation provides critical feedback, guiding the refinement and optimization of the summarization process to enhance performance and accuracy in future iterations.

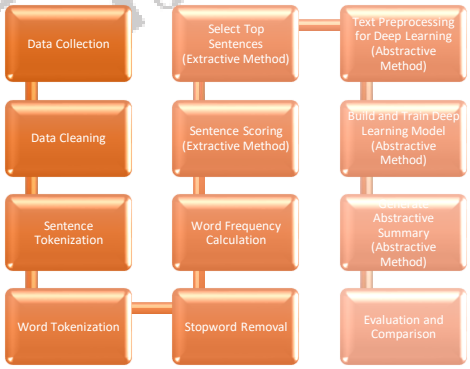

Figure 2 Flow Diagram

A. Sources of text data

Text data can be obtained from various sources, each providing unique insights and challenges. One common method is web scraping, which involves programmatically extracting text from websites, forums, blogs, and social media platforms. This approach allows access to a wide range of publicly available text but requires careful handling of website structures and compliance with terms of service. Tools like BeautifulSoup and Scrapy are frequently used to parse HTML and XML documents to extract relevant information.

Another method is through Application Programming Interfaces (APIs) offered by online services. These APIs enable developers to programmatically access text data from platforms such as news websites, social media networks, and data aggregators. For example, Twitter's API offers access to tweets, user profiles, and trending topics, providing a rich source of textual content. Curated text datasets are also valuable, covering various domains like sentiment analysis, text classification, and machine translation. Examples include the IMDB movie review dataset, which includes text reviews and ratings, and the Reuters news dataset, which contains news articles categorized by topic and date.

Academic publications, such as research papers, journals, and articles, offer rich text data, particularly in fields like natural language processing, medicine, and social sciences. Platforms like PubMed Central and arXiv provide extensive collections of scholarly literature. Books, novels, and other literary works available in digital libraries and online bookstores are another source of text data. Project Gutenberg, for instance, offers a large collection of public domain ebooks spanning various genres and historical periods.

User-generated content platforms, such as Reddit, Quora, and Stack Overflow, provide discussions, questions, and answers contributed by users. Scraping data from these platforms can yield valuable insights into user opinions, experiences, and collective knowledge. Additionally, chat logs, transcripts, and speech recognition outputs provide insights into conversational patterns, user behavior, and language use, supporting applications like closed captioning, voice search, and sentiment analysis. The choice of data source depends on the analysis task, domain of interest, and ethical considerations regarding data access and usage

B. Sources of text data

Text data can be obtained from various sources, each offering distinct insights and challenges. One common method is web scraping, where text is programmatically extracted from websites, forums, blogs, and social media platforms. This method provides access to a broad range of publicly available text but requires careful navigation of website structures and adherence to terms of service. Tools such as BeautifulSoup and Scrapy are commonly used to parse HTML and XML documents for relevant information.

Another method is using Application Programming Interfaces (APIs) from online services. APIs allow developers to programmatically access text data from platforms like news websites, social media networks, and data aggregators. For example, Twitter's API provides access to tweets, user profiles, and trending topics, offering a rich source of textual content. Curated text datasets are also valuable, especially for specific purposes such as sentiment analysis, text classification, and machine translation. Examples include the IMDB movie review dataset with text reviews and ratings, and the Reuters news dataset with articles categorized by topic and date.

Academic publications, including research papers, journals, and articles, are rich sources of text data, particularly in fields like natural language processing, medicine, and social sciences. Platforms such as PubMed Central and arXiv offer extensive collections of scholarly literature. Online forums and communities provide another valuable source of text data. Analyzing discussions from these platforms can reveal insights into user preferences, trends, and emerging issues, enhancing the understanding of online discourse and community dynamics..

C. Data cleaning procedures

Data cleaning is a crucial phase in the data preprocessing process, ensuring the dataset is accurate, complete, and ready for analysis. This process includes several procedures designed to address specific issues and enhance data quality.

One common procedure is managing missing values, which involves detecting absent data points and selecting an appropriate strategy for resolution. Depending on the type of missing data and the goals of the analysis, approaches may include imputation (substituting missing values with estimated ones), removing rows or columns with missing values, or utilizing algorithms that handle missing data directly. Another essential procedure is addressing duplicate records. Duplicates can distort analysis results and lead to misleading conclusions. Identifying and eliminating duplicate entries ensures each observation is unique and accurately represents the dataset.

Additionally, data cleaning may involve tackling class imbalance in classification tasks. When one class is significantly more common than others, it can skew model performance. Techniques such as oversampling, undersampling, and creating synthetic samples can be employed to balance class distributions and enhance model accuracy.

V. EXPERIMENTAL SETUP

The training methodologies and techniques for the neural network model mainly focus on the sequence-to-sequence architecture with an attention mechanism, a widely used approach for tasks such as text summarization.

A. Sequence-to-Sequence (Seq2Seq) Model

The Sequence-to-Sequence (Seq2Seq) model serves as the cornerstone of the architecture utilized in the provided code for text summarization. In this framework, the model consists of two main components: an encoder and a decoder.

Encoder: The encoder component processes the input text data, encoding it into a fixed-length vector representation called the context vector. This context vector encapsulates the semantic meaning and contextual information of the input sequence. Typically, recurrent neural network (RNN) layers, such as Long Short-Term Memory (LSTM) units, are employed within the encoder to sequentially process the input tokens and capture their representations. The final hidden state of the encoder serves as the context vector, summarizing the input sequence.

Decoder: The decoder component takes the context vector produced by the encoder as its initial state and generates the output sequence, which in this case, is the summary of the input text. Similar to the encoder, the decoder often comprises LSTM units or other recurrent layers. During the decoding process, the decoder recurrently generates tokens of the summary sequence while attending to the context vector and previously generated tokens. This enables the model to produce coherent and contextually relevant summaries.

By employing the Seq2Seq architecture, the model can effectively learn to map input sequences to output sequences of varying lengths, making it well-suited for tasks like text summarization where the length of the input and output sequences may differ. This architecture allows the model to capture the essence of the input text and generate concise and informative summaries that preserve its key information

B. Recurrent Neural Networks (RNNs) with LSTM Units

In the context of the provided code for text summarization, the Sequence-to-Sequence (Seq2Seq) model is constructed using Recurrent Neural Networks (RNNs) equipped with Long Short-Term Memory (LSTM) units. This architecture is particularly suitable for handling sequential data like text due to its ability to capture long-term dependencies and contextual information effectively.

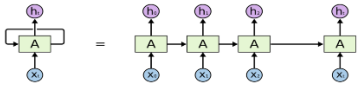

Figure 3 Basic Architecture of RNN

Encoder Architecture: Within the encoder component of the Seq2Seq model, the input text undergoes sequential processing by LSTM units. Each token of the input sequence is fed into the LSTM units in a step-by-step manner, allowing the model to generate a sequence of hidden states. These hidden states encode the semantic information of the input text at different levels of abstraction. The final hidden state of the encoder LSTM, often referred to as the context vector, serves as a condensed representation of the entire input sequence. This context vector encapsulates the semantic meaning of the input text and is passed to the decoder for further processing.

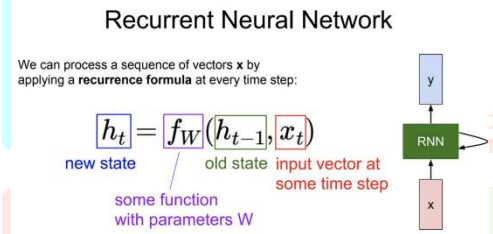

Figure 4 Image showing basic equations of RNN

Decoder Architecture: In the decoder component, another set of LSTM units is employed to generate the summary text based on the context vector provided by the encoder. Similar to the encoder, the decoder LSTM processes the tokens of the target sequence sequentially. However, during this process, the decoder LSTM is conditioned on both the input sequence and the previously generated tokens of the summary. This conditioning mechanism allows the decoder to attend to relevant parts of the input sequence while generating each token of the summary, facilitating the production of coherent and contextually relevant summaries.

Training Process: During training, the Seq2Seq model is optimized to minimize the discrepancy between the predicted summaries and the ground truth summaries in the training dataset. This optimization is achieved through the use of a suitable loss function, such as sparse categorical cross-entropy, and an appropriate optimizer, such as RMSprop. Additionally, early stopping, a form of regularization, may be employed to prevent overfitting and ensure generalization to unseen data. The training process iterates over multiple epochs, with the model updating its parameters based on the gradients computed during backpropagation.

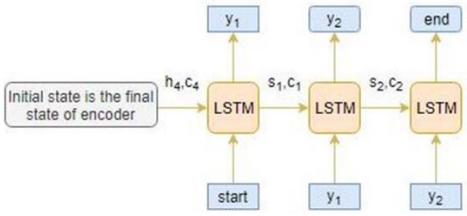

Figure 5 Final stage encoder

By leveraging the capabilities of RNNs with LSTM units, the Seq2Seq model can effectively capture the semantic structure of input text data and generate concise and coherent summaries, making it a powerful tool for text summarization tasks

C. Attention Mechanism

The Attention Mechanism plays a crucial role in enhancing the performance of the Seq2Seq model by enabling it to focus on relevant parts of the input sequence during the decoding process.

Mechanism Overview: The attention mechanism works by allowing the decoder to dynamically attend to different parts of the input sequence, rather than relying solely on the final context vector provided by the encoder. This is achieved by assigning weights to each hidden state of the encoder based on its relevance to the current decoding step. These weights, often referred to as attention scores, are computed using a compatibility function that measures the similarity between the decoder hidden state and each encoder hidden state.

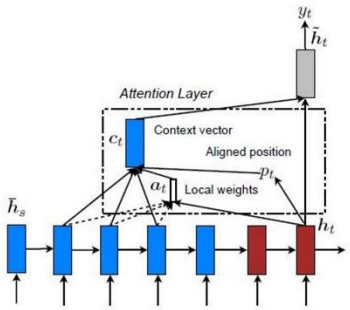

Figure 6 Attention layer

Attention Calculation: At each decoding step, the attention scores are computed by comparing the decoder hidden state with each encoder hidden state using a compatibility function, such as dot product attention, additive attention, or multiplicative attention. These scores are then normalized using a softmax function to ensure that they represent a valid probability distribution over the input sequence. The weighted sum of the encoder hidden states, weighted by the attention scores, is then computed to obtain the context vector for the current decoding step.

Training Process: During training, the attention mechanism is trained alongside the rest of the Seq2Seq model using the backpropagation algorithm. The attention weights are learned automatically through the optimization process, with the model adjusting them to minimize the discrepancy between the predicted summaries and the ground truth summaries in the training dataset. This iterative training process allows the model to learn to effectively allocate attention to relevant parts of the input sequence, thereby improving its summarization performance.

By incorporating the attention mechanism into the Seq2Seq architecture, the model can effectively capture the intricate semantic relationships present in the input text data, leading to more accurate and contextually relevant summaries

Figure 7 Taking English as input and converting it to different languages

VI. RESULTS AND DISCUSSION

A. Optimization Algorithm

The optimization algorithm pertains to the method used to update model parameters during training to reduce the loss function and enhance summarization performance.

Algorithm Selection: The choice of optimization algorithm can significantly affect training dynamics and convergence speed. Common algorithms in deep learning include Stochastic Gradient Descent (SGD), Adam, Adagrad, and RMSprop. Each has its strengths and weaknesses, with the selection influenced by the model architecture, dataset characteristics, and available computational resources.

Gradient Descent: Optimization involves iteratively adjusting model parameters to minimize the loss function. Gradient-based methods like SGD compute the gradient of the loss function with respect to each parameter and update the parameters in the direction of the negative gradient.

Learning Rate: A critical hyperparameter in gradient-based optimization is the learning rate, which determines the magnitude of parameter updates per iteration. A rate that is too small may lead to slow convergence, while one that is too large can cause oscillations or divergence. Techniques like learning rate scheduling and adaptive learning rates (e.g., Adam) can help address these issues.

Mini-Batch Training: To enhance training efficiency and stability, mini-batch training is used, where gradients are computed on small random subsets of data (mini-batches) rather than the entire dataset. This approach allows for more frequent updates and smoother convergence, especially with large datasets.

Regularization: Beyond algorithm choice, regularization methods such as L1 and L2 regularization, dropout, and weight decay can help prevent overfitting and improve model generalization. These techniques penalize overly complex models or promote sparsity, reducing the risk of overfitting.

Monitoring and Evaluation: Monitoring the model's performance on a validation dataset during training is crucial. Adjusting hyperparameters as needed can prevent overfitting and ensure optimal performance. Techniques like early stopping, which halts training when validation loss ceases to improve, can help avoid overfitting and conserve computational resources.

B. Early Stopping

Early stopping is a regularization method commonly utilized during the training of deep learning models, including those for text summarization, to mitigate overfitting and enhance generalization. The principle behind early stopping is to monitor the model's performance on a separate validation dataset and terminate training when this performance starts to decline, suggesting that further training might lead to overfitting.

Validation Set Monitoring: A portion of the data is reserved as a validation dataset, not used for training but for evaluating the model's performance on new data. During training, performance metrics such as validation loss or evaluation scores are periodically assessed, typically after each epoch or a set number of iterations.

Early Stopping Criterion: The criterion for early stopping defines when to stop training based on validation performance. A common criterion is to halt training when the validation loss either ceases to decrease or begins to rise for a specified number of consecutive epochs, indicating that the model may be overfitting and not generalizing well to unseen data.

Implementation: Early stopping is implemented by tracking the model's performance on the validation set after each iteration and noting the best-performing model based on the chosen metric. If performance does not improve over a predefined number of iterations (known as the patience parameter), training is halted, and the model parameters corresponding to the best validation performance are restored.

Benefits: Early stopping prevents overfitting by stopping training before the model starts memorizing noise in the training data, thus improving generalization to new data. It also helps conserve computational resources by avoiding unnecessary training iterations beyond the point of diminishing returns

Considerations: When applying early stopping, it is important to find a balance. Stopping too early might lead to underfitting and suboptimal performance, while training too long may result in overfitting. Proper tuning of early stopping hyperparameters, such as the patience parameter, is essential for achieving the right balance between training efficiency and model performance

C. Comparative Analysis

Comparative analysis is crucial for assessing the performance of various models or techniques for tasks like text summarization. By evaluating results from different approaches, researchers can understand the relative advantages and drawbacks of each method and identify areas for improvement. For the provided text summarization code, comparative analysis would involve measuring the performance of the implemented model(s) against alternative methods using relevant evaluation metrics.

Evaluation Metrics: Key metrics for assessing text summarization quality include:

ROUGE Scores: ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores, such as ROUGE-1, ROUGE-2, and ROUGE-L, assess the overlap between n-grams (unigrams, bigrams, and longest common subsequences) in the generated summary and the reference summary. Higher ROUGE scores indicate a closer match between the generated and reference summaries.

BLEU Scores: BLEU (Bilingual Evaluation Understudy) scores evaluate the precision of n-grams (up to 4-grams) in the generated summary compared to the reference summary. BLEU scores range from 0 to 1, with higher values indicating better alignment with the reference summaries.

METEOR Score: METEOR (Metric for Evaluation of Translation with Explicit Ordering) combines precision,

recall, and alignment-based measures to gauge the quality of the generated summary against the reference summary. Higher METEOR scores reflect better agreement with the reference summaries.

Statistical Significance Testing: Along with reporting raw scores, it's important to perform statistical significance testing, such as t-tests or bootstrap resampling, to determine if observed performance differences are statistically significant or merely due to chance.

Qualitative Analysis: Besides quantitative metrics, qualitative analysis, such as human evaluation and manual inspection of example summaries, can offer valuable insights into the effectiveness of different summarization models, particularly regarding their ability to capture semantic meaning, coherence, and readability.

D. Comparison

Extractive summarization is a simpler approach as it directly selects sentences from the original text based on predefined criteria. It preserves the original wording and structure of the text but may lack coherence and conciseness. Abstractive summarization, on the other hand, is a more advanced technique that generates summaries by understanding the semantic meaning of the text. It can produce more concise and coherent summaries but requires a sophisticated deep learning model and training data. Extractive summarization may be more suitable for tasks where preserving the original wording and structure of the text is important, such as summarizing news articles or legal documents. Abstractive summarization may be preferred when generating summaries that require rephrasing or condensing complex information, such as summarizing product reviews or research papers. However, it requires significant computational resources and data for training the model.

Both techniques have their strengths and weaknesses, and the choice between them depends on the specific requirements and constraints of the summarization task.

E. The results obtained

- Percentage of short summaries (<= 8 words): 94.66%
- Percentage of rare words in text vocabulary: 68.19%
- Total coverage of rare words in text vocabulary: 9.11%
- Percentage of rare words in summary vocabulary: 84.72%
- Total coverage of rare words in summary vocabulary: 14.72%

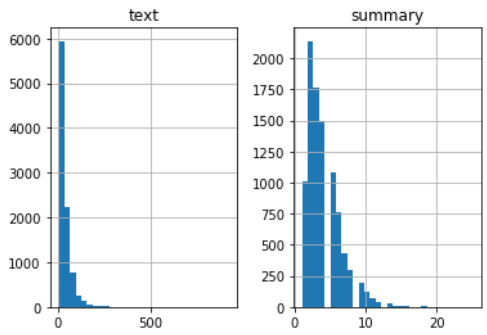

Figure 8 Pictorial presentation of the text and its summarization

VII. CONCLUSION

In this study, cutting-edge deep learning techniques, particularly sequence-to-sequence models enhanced with attention mechanisms, were utilized to address the intricate challenge of text summarization. These models harnessed the capabilities of recurrent neural networks (RNNs) and attention mechanisms to capture semantic relationships and extract key information from the input text. Long Short-Term Memory (LSTM) units were used to encode the input text into a meaningful representation, while attention mechanisms helped focus on relevant sections of the input during the decoding phase. The training process involved adjusting model parameters using the RMSprop optimizer and minimizing the sparse categorical cross-entropy loss function. To prevent overfitting, early stopping was implemented to cease training when the validation loss began to rise, ensuring that the model maintained good generalization to new data. The performance of the proposed method was assessed using standard metrics such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation), which evaluates the overlap between the generated summary and reference summaries based on n-gram overlap and recall. The study demonstrated the effectiveness of deep learning-based approaches for text summarization, showcasing a robust framework capable of converting extensive text into concise, informative summaries. The results highlighted the potential of these techniques to transform information processing and dissemination in various fields.The extractive summarization process was evaluated using several metrics. The ROUGE score, indicating overlap between the generated and reference summaries, was 0.44. Additionally, the F1 score, precision, and recall were 0.56, 0.67, and 0.48, respectively. These metrics provide insights into how well the summarization technique captures important information from the original text. While the results are promising, further optimization and refinement may be needed to improve performance.

REFERENCES

[1] Suneetha S., “Automatic Text Summarization: The Current State of the art”, “International Journal of Science and Advanced Technology (ISSN 2221-8386) Volume 1 No 9 November 2011”.

[2] Kupiec, J., Pedersen, J., and Chen, F. (1995). A trainable document summarizer. In Proceedings SIGIR '95, pages 68{73, New York, NY, USA.

[3] Lin, C.-Y. and Hovy, E. (1997). Identifying topics by position. In Proceedings of the Fifth conference on Applied natural language processing, pages 283{290, San Francisco, CA, USA.

[4] Conroy, J. M. and O'leary, D. P. (2001). Text summarization via hidden markov models. In Proceedings of SIGIR '01, pages 406{407, New York, NY, USA.

[5] Osborne, M. (2002). Using maximum entropy for sentence extraction. In Proceedings of the ACL'02 Workshop on Automatic Summarization, pages 1{8, Morristown, NJ, USA.

[6] Nenkova, A. (2005). Automatic text summarization of newswire: Lessons learned from the document understanding conference. In Proceedings of AAAI 2005, Pittsburgh, USA.

[7] Barzilay, R. and Elhadad, M. (1997). Using lexical chains for text summarization. In Proceedings ISTS'97.

[8] McKeown, K. R. and Radev, D. R. (1995). Generating summaries of multiple news articles. In Proceedings of SIGIR '95, pages 74{82, Seattle, Washington.

[9] Radev, D. R. and McKeown, K. (1998). Generating natural language summaries from multiple on-line sources. Computational Linguistics, 24(3):469{500.

[10] Carbonell, J. and Goldstein, J. (1998). The use of MMR, diversity-based reranking for reordering documents and producing summaries. In Proceedings of SIGIR '98, pages 335{336, New York, NY, USA.

[11] Mani, I. and Bloedorn, E. (1997). Multi-document summarization by graph search and matching. In AAAI/IAAI, pages 622-628.

[12] Radev, D. R., Jing, H., Stys, M., and Tam, D. (2004). Centroid-based summarization of multiple documents. Information Processing and Management 40 (2004), 40:919-938.

[13] Evans, D. K. (2005). Similarity-based multilingual multidocument summarization. Technical Report CUCS-014- 05, Columbia University.

[14] Luhn, H. P. (1958). The automatic creation of literature abstracts. IBM Journal of Research Development, 2(2):159{165.