www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 04, April 2025, pp : 1394-1397

e-ISSN : 2583-1062
Impact Factor : 7.001

DOCUMENT, BOOKS SUMMARIZATION THROUGH DEEP LEARNING

Naitik Pareek[1], Joel Varghese[2], Rishabh Gupta[3], Ritik Sharma[4], Punit Kumawat[5]

[1,2,3,4,5]Poornima Institute Of Engineering And Technology, India.

DOI: https://www.doi.org/10.58257/IJPREMS39978

ABSTRACT

This project employs a deep learning-based approach in summarizing large textual material, such as books and documents. The system utilizes an encoder-decoder model with Long Short-Term Memory (LSTM) networks, along with an attention mechanism to generate abstractive summaries that resemble human-written content. The ultimate goal is to generate coherent and meaningful summaries that summarize the original work. The development was underpinned by the Agile Scrum methodology, enabling effective collaboration and tracking progress through four well-defined sprints. Preprocessing of data, model training, and testing were conducted using a range of tools, including TensorFlow, Keras, and NLTK. ROUGE and BLEU scores were used to assess the quality of generated summaries, producing promising results. The final model efficiently summarizes long inputs to concise summaries, indicating its potential application in the education, research, and publishing sectors. Future improvements include the utilization of transformer-based models and the support of multiple languages.

Keyword: Deep Learning, Abstractive Summarization, LSTM, Attention Mechanism, Agile Scrum, ROUGE, BLEU, Text Summarization, Natural Language Processing, Document Summary.

1. INTRODUCTION

In this age of excessive information, the amount of text available in formats such as documents, books, articles, and reports can often be overwhelming for individuals. The challenge of efficiently extracting pertinent information from this plethora of data continues to grow for students, professionals, and researchers at all levels. Manual summarizing tends to take a long time, and errors and subjective human reasoning are common. To help with this issue, we proposed a project entitled "Document, Books Summarisation through Deep Learning," which elaborates on a smart and scalable approach to gaining benefits from artificial intelligence to create concise, coherent summaries from extensive text format assets or documents of literature. The main goal of this project is to design and create an intelligent system to summarize documents and books content using deep learning approaches using advanced methods in Natural Language Processing (NLP). For instance, we plan to implement an Abstractive approach as opposed to the traditional extractive summarization, which include selecting sentences from content prescribed. In saying this, we will have a model that processes and learns how to create new phrases and sentences to form a summary to present the ideas of the input in a humanistic fashion. The intelligent system will accept input in formats of plain text, documents, PDFs, and/or documents (which can be retrieved through Optical Character Recognition from a scanned image) and extract valuable and meaningful summary that has been established, filtered, and summarized.

2. METHODOLOGY

2.1 Agile Scrum Process:

In order to foster a sustainable and efficient development process, we deployed Agile Scrum as a software development methodology for the project. The iterative process allowed the team to break down the tasks into smaller segments and improve the system continuously using feedback and collaboration. The entire development process was split into four distinct sprints, and each sprint was assigned specific milestones that were crucial to the success of the project.

- **Sprint 1:** It was about preliminary groundwork, such as finalizing the project title, a thorough literature review, and creating a Gantt chart to map the overall timeline and task dependencies. Sprint 1 assisted in defining clear objectives and scheduling team responsibilities.
- **Sprint 2:** It involved data-related tasks such as discovering and collecting appropriate datasets appropriate for training a summarization model. Sprint 2 also involved requisite text preprocessing steps such as cleaning data, tokenization, and padding to get the raw input text model-training-ready.
- **Sprint 3:** It was the core development phase, where the team built and trained the summarization model. An LSTM encoder-decoder model with attention was used, implemented in Python and deep learning libraries like TensorFlow and Keras. The phase involved multiple iterations to optimize hyperparameters and improve model performance.
- **Sprint 4:** It was focused on model evaluation. The trained model was evaluated with common NLP evaluation metrics like ROUGE and BLEU. The final outcome, conclusions, and limitations were also noted in detail.

www.ijprems.com
editor@ijprems.com
INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 04, April 2025, pp : 1394-1397
e-ISSN : 2583-1062
Impact Factor : 7.001

There was a designated role for each team member:

To ensure proper coordination and task management, team roles were assigned according to the Agile Scrum model. Every team member was assigned a specific role to enhance responsibility and workflow efficiency.

- **Scrum Master**: The Scrum Master facilitated the team, facilitated the sessions, and kept the project on track in accordance with the timelines and goals for the sprint.
- **Developers:** The developers were responsible for productionizing the deep learning model. They would implement, train, and optimize the performance of a LSTM (Long Short-term Memory) based encoder-decoder model with attention.
- **Tester:** Validate was done by the Tester who ensured that the model outputs were accurate and aligned with expectations. They also used evaluation metrics, such as ROUGE and BLEU to assess the model performance.
- **Documenter:** The Documenter kept up all of the required documents, such as sprint reports, minutes from meetings, technical reports, and the project summary.

  Progress tracking was facilitated through the use of Excel-based sprint backlogs and burndown charts. These provided a clear-to-understand visual presentation of work done versus work remaining, facilitating team transparency and the capacity to make iterative refinement throughout the project.

  This systematic distribution of tasks and monitoring of progress ensured timely completion and quality delivery of each stage of every project.

2.2 Technical Stack:

- Programming Language: Python
- Selected for its simplicity and strong support for machine learning and NLP.
- •Libraries and Frameworks:
- TensorFlow and Keras – For designing and training the deep learning model.
- NLTK (Natural Language Toolkit) -Utilized for text preprocess operations such as tokenization, stop-word deletion.
- **Architectural Framework:**
- The Seq2Seq model uses Long Short-Term Memory (LSTM) networks.
- Attention mechanism in the decoder to pay attention to contextually appropriate words while generating the summary.
- **Dataset:**
- Large documents and public domain books.
- Preprocessing included lowercasing, punctuation removal, tokenization, and sequence padding.

2.3 Preprocessing Steps:

Text data preprocessing is an important process of constructing a correct and effective deep learning model for text summarization. Various preprocessing methods were used to pre-process the dataset for model training:

- **Text Cleaning:** The unprocessed textual data was replete with various extraneous components, including special characters, surplus whitespace, punctuation marks, and stop words, all of which could adversely affect the model's learning efficacy. Consequently, these elements were methodically eliminated to preserve solely the significant content for subsequent analysis and summarization.
- **tokenization:** The text was subsequently decomposed into single word sequences, which is referred to as tokenization. This procedure enables the input text to be split into comprehensible pieces (tokens), thereby allowing the model to comprehend and learn patterns and word relations.
- **padding:** As neural networks need to receive inputs of fixed length, all of the tokenized sequences were normalized with padding in order to ensure that each input had the same number of tokens by either truncating the longer sequences or padding the shorter sequences with zero tokens.
- **Vocabulary Limitation:** To achieve maximum efficiency in terms of memory use and model training, the vocabulary was limited to the most frequent 50,000 vocabulary words from the corpus. This vocabulary limitation was intended to prevent the model from being over-populated with single words that were used either infrequently or redundantly, while also preventing any potential reduction in critical grammatical features.

These pre-processing steps were essential to producing clean, consistent, quality input data for the summarization model training process.

www.ijprems.com
editor@ijprems.com
INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 04, April 2025, pp : 1394-1397
e-ISSN : 2583-1062
Impact Factor : 7.001

2.4 Model Design:

The summarization structure used here is a sequence-to-sequence (Seq2Seq) architecture that uses Long Short-Term Memory (LSTM) networks. The two key components of the model are an encoder and a decoder.

The encoder LSTM is the one that reads and is aware of the input text. It reads the input sequence word by word and maps it to a fixed-dimensional context vector, which contains semantic information about the entire sequence. The encoded representation is used as input to the decoder.

The decoder LSTM then utilizes this context vector to produce the output sequence, i.e., abstractive summary. However, using just a single context vector tends to lose valuable information, particularly for long input sequences.

To mitigate this constraint, an attention mechanism is incorporated between the encoder and decoder. The attention layer selectively emphasizes various segments of the input sequence during the generation of each word in the summary. This enables the decoder to "attend" to particular words or phrases in the input that hold the greatest relevance at each stage of the generation process.

This improvement greatly boosts the contextual coherence, relevance, and general quality of the produced summaries, making them closer to human writing and semantically accurate.

3. RESULTS

The performance of the constructed deep learning model was assessed employing three standard metrics in the natural language processing community: ROUGE-1, ROUGE-2, and BLEU. The metrics provide an overall view of the model's capacity to generate summaries accurately and fluently compared to human-crafted references.

Metric	Score
ROUGE-1	0.51
ROUGE-2	0.39
BLEU	0.43

- **ROUGE-1 (0.51)** estimates the degree of overlap in unigrams (single words) between the machine summary and reference summary and, therefore, the model's ability to maintain significant keywords of the original text.
- **ROUGE-2 (0.39)** is a measure of recall of bigrams (two-word units), which suggests that the model has the capacity to recognize and replicate important phrase-level information from the source.
- **BLEU (0.43)** measures fluency, grammar, and overall language structure, showing that the decoder is successful in acquiring language rules and producing coherent sentences.

These findings prove that the attention-based LSTM-based encoder-decoder effectively learns and extracts contextual information in the text. The attention

mechanism dynamically attends to the most important parts of the input sequence at decoding time, producing more accurate and informative abstracts.

3.1 Qualitative Example:

- **Input Excerpt:** “The history of medicine began ages ago, before writing was introduced. The early societies developed their own way of treating disease, and all those were utilized by subsequent population.”
- **Generated Summary:** “ The history of medicine has developed from ancient times, with present-day practices drawing upon previous knowledge.”
- This qualitative result indicates that the model is abstractive. It does not simply copy input sentences but generates a short, meaningful abstraction based on context, which indicates that it can abstract and generalize the source information well.

Overall, the evaluation suggests that deep learning, specifically attention-based models, is particularly well-suited for summarization tasks.

4. CHALLENGES FACED

During the development of our deep learning text summarization model, we encountered a sequence of significant challenges that needed to be addressed with care:

- **Extended Input Management:** One of the primary issues with standard LSTM models is that they don't handle and retain data from extremely lengthy sequences well. This is a significant problem when dealing with long texts or large documents, which are common for summarization. To address this, we employed input truncation methods to reduce lengthy sequences without sacrificing key details. We also implemented token-level padding to ensure all input sizes are equal among training samples, which is crucial for batch processing and model stability.

- **Overfitting**: Since our training data was small, the model initially exhibited overfitting. That is, it performed well on the training data but not on novel data. To avoid this, we introduced dropout layers to the LSTM units to randomly switch off some neurons during training, thus enabling the model to learn more. In addition, we employed early stopping to stop training when the validation loss stopped decreasing. This prevented unnecessary training and minimized overfitting.
- **Computational Constraints**: Training deep learning models, especially sequence-to-sequence models with attention, is computationally intensive. To overcome the constraint of access to hardware, we made use of Google Collab, which provides access to free GPUs. We also used batch sizes and adjusted them to operate within available memory constraints to encourage training efficiency.

5. FUTURE WORK

- **Integration of Transformer Models:**
- Use more sophisticated transformer-based models like BERT, GPT, and T5.
- These architectures have improved language understanding and generate smoother and more accurate summaries.
- **Hybrid Summarization Approach:**
- Combine both extractive and abstractive summarization methods.
- Improves information retention without compromising on natural language generation.
- **Web-Based Deployment:**
- Deploy the model using web development frameworks such as Streamlit or Flask.
- Offers users an engaging interface through which they can submit.
- **Multilingual Support:**
- Train the model on multilingual corpora.
- Allows the system to summarize documents in various languages for broader dissemination.
- **Training on Larger and Diverse Datasets:**
- Allow for more types of documents and more data.
- Enhances model stability, performance, and generalization across different content spaces.

6. CONCLUSION

The project successfully showcased how deep learning models can tackle the real-life problem of automatic text summarization with an emphasis on books and long-form documents In utilizing an encoder-decoder deep learning architecture based on Long Short-Term Memory coupled with an attention mechanism, we showed how this model could produce concise, readable, and coherent summaries that could be classified as a new abstractive summary without loss of important content or context for the original text. The development process adopted an orderly approach using the Agile Scrum management methodology, which promoted timely and collaborative completion of project stages through iterative synchronized work. By organizing the project into four well-defined sprints with specific and defined roles of a Scrum Master, Developer, Tester, and Documenter, the team was able to maintain collaborative efforts while delivering quality outputs at every stage. Results from the evaluation phase with ROUGE and BLEU scores showed that the model was able to maintain semantic congruence of natural language output that closely approximated summaries created by humans. Along with the utilization of preprocessing methods, training methods, and optimization for performance, the project was able to avoid many pitfalls such as overfitting, input length, and computational limits. Thus, our project provided a framework that illustrates that deep learning can be operationalized with the appropriate design choices and project management.

7. REFERENCES

[1] A. See, P. J. Liu, and C. D. Manning, "Get to the Point: Summarization with Pointer-Generator Networks," Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL), 2017, pp. 1073–1083.

[2] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," arXiv preprint arXiv:1409.0473, 2014.

[3] K. Rush, S. Chopra, and J. Weston, "A Neural Attention Model for Abstractive Sentence Summarization," arXiv preprint arXiv:1509.00685, 2015.

[4] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv preprint arXiv:1810.04805, 2018.

[5] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," Text Summarization Branches Out: Proceedings of the ACL-04 Workshop, 2004.