PPT GENERATION FROM REPORT

Sebin Thomas
Dept. of Computer Science & Eng.
Amal Jyothi College of Engineering
Kanjirapally
sebinthomas2023@cs.ajce.in

John V G
Dept. of Computer Science & Eng.
Amal Jyothi College of Engineering
Kanjirapally
johnvg2023@cs.ajce.in

Josin Chacko
Dept. of Computer Science & Eng.
Amal Jyothi College of Engineering
Kanjirapally
josinchacko2023@cs.ajce.in

Mariyam Shajahan
Dept. of Computer Science & Eng.
Amal Jyothi College of Engineering
Kanjirapally
mariyamshajahan2023@cs.ajce.in

Sharon Sunny
Department of computer science
Amal Jyothi College of Engineering
Kanjirapally
sharonsunny@amaljyothi.ac.in

***Abstract*— Presentations have become an important part of work life whether you are a student or an employee or a businessman. But creating a presentation is a time consuming process. Our method aims to reduce the amount of work put into creating a presentation by automatically generating an outline or a framework of the presentation based on text input. The system is a website that will be able to read text input and create segments of text by analyzing the text and grouping sentences that belong to the same topic. Then, each of these segments will be summarised and key points will be generated. These points will be distributed over several slides. Appropriate titles will be generated for each slide based on the contents. The presentation can be exported as a .pptx file which means the user can edit the file easily and make any changes if necessary. The system uses Google App Script to generate a file with .pptx and GPT-2 NLP model to summarise text and generate key points and to generate titles.**

***Index Terms*— Segmentation, text summarization, key point generation**

I. INTRODUCTION

The ability to quickly and easily generate high-quality presentations can be a valuable asset in today's fast-pace business world. However, creating a professional-looking presentation can be time-consuming and require a lot of effort. This work aims to address this problem by developing a system that can automatically generate a presentation from a given text document. A preprocessed input file is fed to the system for further processing. Text preprocessing is a critical step in Natural Language Processing (NLP) that involves cleaning, normalizing, and transforming raw text data into a more structured format that can be easily analyzed by machine learning algorithms. Common text preprocessing techniques include lowercasing, removing stop words, stemming/lemmatization, tokenization, and part-of-speech tagging. These techniques help to standardize and reduce the complexity of the text data, making it easier for machine learning models to extract meaningful insights from it. The system converts the text into segments based on topic. Topic-wise text segmentation is a technique used in NLP to divide a document into segments based on its content. The aim is to identify the main topics or themes covered in the text and group related sentences together. Topic segmentation can be achieved using various methods, including supervised and unsupervised approaches. One common method is to use clustering algorithms that group similar sentences together based on their semantic content. Topic segmentation is a critical preprocessing step in many NLP applications, such as document summarization, information retrieval, and text classification. It helps to reduce the complexity of the text data and enables more accurate analysis and interpretation of the information contained in the document. These segments are summarised and titles are generated. Text summarization is a NLP technique that involves automatically generating a shortened version of a long text while preserving its most important information. The goal of text summarization is to create a condensed version of a document that is easier to read and comprehend, while still retaining the key points and important details of the original text. Summarization can be done using various techniques, including extractive and abstractive methods. Extractive summarization involves selecting the most relevant sentences from the original text, while abstractive summarization involves creating new sentences that capture the meaning of the original text. Text summarization has many practical applications, such as in news article summarization, document summarization, and search result snippets. Title generation is a NLP technique that involves automatically generating a headline or title for a given text or document. The goal of title generation is to create a brief, catchy, and informative headline that accurately reflects the content of the document. Title generation can be achieved using various methods, including rule-based systems and machine learning algorithms. Machine learning-based approaches involve training a model to generate titles based on a large corpus of labeled examples. Title generation has many practical applications, such as in news article headline generation, social media post captioning, and content marketing. It can help to increase engagement and improve the readability of the text by providing a concise and informative summary of the document. Key points are generated from the summary and are placed on different slides in the correct order. Key point generation in NLP involves automatically identifying and summarizing the most important information from a given text. This task is usually

DOI: 10.5281/zenodo.8012970

performed by a machine learning algorithm that analyzes the text and identifies important phrases, sentences, or even entire paragraphs. The resulting key points can be used to provide a summary of the text, aid in information retrieval, or serve as input for further analysis. Key point generation has applications in various fields, including news article summarization, academic paper summarization, and text analytics. Effective key point generation requires sophisticated NLP techniques, including natural language understanding, topic modeling, and summarization algorithms. Appropriate titles are given to each slide. The output file is generated using Google apps script. Google Apps Script is a scripting language developed by Google that allows users to customize and automate various Google applications, including Google Slides. It is a powerful tool that enables users to create custom functions, automate repetitive tasks, and enhance the functionality of Google Slides. Google Apps Script for Google Slides can be used to automate the process of creating and updating slide decks. By writing scripts, users can add new slides, update existing ones, and even import content from other sources. This can save significant time and effort, especially for users who frequently work with large slide decks or regularly update presentations. Google Apps Script also provides access to various Google Slides APIs, allowing users to perform advanced tasks such as adding animations and interactivity to slides, embedding videos and other media, and creating dynamic charts and graphs. These capabilities can be used effectively to generate the required output file.

II. Literature Survey

As discussed earlier, it aims to create a system that can assist in creating presentations from text documents. Creating presentations automatically is not an active research field. Text generation and summarization are hot research areas because of its usefulness in the new era. This domain is still in its growing phase and hence there is ample scope for improvement. Here, the proposed system takes text report as input, divide them into segments, summarize each segment, generate titles and writes these into different slides of the presentation.

Text segmentation is a method of splitting a document into smaller parts, which is usually called segments. Each segment has its relevant meaning. Those segments are categorized as words, sentences, topics, phrases or any information unit depending on the task of the text analysis. Text segmentation is an important part of the proposed model. The text documents received are preprocessed to remove meaningless elements such as proper names and pronouns. The text is tokenized and nouns, verbs adjectives etc are removed. Now, the words which are left are considered candidate features for topic selection. LDA works on this group of words to form clusters in an unsupervised manner. The resulting clusters are interpreted by human experts and class labels are termed. The labels are later assigned to the documents based on the classmembership probabilities. These documents are used to train the LDA model. To identify topic-specific segments in the document, a sliding window is implemented. The processing is terminated when there are not enough sentences left to do so. The Choi dataset was used to conduct experiments for performance evaluation. [2] describes neural text segmentation and its application in sentiment analysis. A generic end-to-end segmentation model, named SEGBOT, is provided which uses a bidirectional recurrent neural network to encode an input text sequence. SEGBOT consists of three components, a context encoder, a boundary decoder and a boundary pointer. Here, each input is first represented with a distributed representation. Then, the input sequence is encoded issuing an RNN. Here, a bidirectional GRU network is used to memorize part and future information in the input sequence. The decoder takes a start unit in the input sequence as input and transforms it to its distributed representation by looking up the corresponding embedding matrix. It is then passed through the GRU layer. At each step, a distribution over possible positions in the input sequence for a possible segment boundary is computed by the output layer of the decoder. A pointing mechanism is implemented to deal with the problem of changing the number of possible positions in the input sequence. From these two, we are choosing the method proposed by [11] as it is a much simpler concept.

Mishra, et al. discusses an automatic title generator for a given text. Automatic title generation is a fascinating field of research that seeks to automatically generate titles for digital documents, such as webpages and digital articles. The area of automatic title generation has seen a surge of interest in recent years, due to the increasing need for efficient methods of generating titles for digital content. An OpenAI's pretrained transformer language model GPT-2 is used to create the model that can generate a list of suitable titles from which selects the most appropriate title and is refined to get the final title. This model was trained using the arXiv dataset which consists of json objects of about 41k research papers related to different scientific fields. It only considered the abstract and titles for the input for the model. The model was evaluated by comparing it to the baseline algorithms using ROUGE and BLEU metrics which are the current standard evaluation metrics in summarization systems. Also, human evaluators were used to check the accuracy of the model. Their model was able to produce syntactic and semantically correct titles effectively without the need for a larger training dataset.

The purpose of automatic text summarization is to produce concise summaries of text-based content with the same meaning and structure. In recent years, significant progress has been made in the development of automated summarization algorithms, with a particular focus on extracting important information from long documents.

There are basically two types of text summarization, i.e, Extractive text summarization and Abstractive text summarization. Extractive text summarization is a type of text summarization that uses an algorithm to select important sentences from the original document and concatenate them to

form a summary. This type of summarization does not require any sophisticated natural language processing algorithms as it does not generate any new sentences. Instead, it relies on the algorithm to identify the most important sentences in the document and extract them for the summary. Extractive summarization is often seen as simpler than abstractive summarization because it does not require any understanding of the context of the text, while Abstractive text summarization is a type of text summarization that uses natural language processing and deep learning algorithms to generate a summary of an original document by creating new phrases and sentences that capture the meaning of the original text. This type of summarization is different from extractive summarization, which uses an algorithm to select important sentences from the original document and concatenate them to form a summary. Abstractive summarization is often seen as more difficult than extractive summarization because it requires understanding the context of the text and using sophisticated natural language processing algorithms to generate new sentences. Extractive summarization methods are easy to implement due to their less complexity compared to abstractive summarization methods.

The task of automatic text summarization is an important problem in natural language processing. In this section, the related work in the field of automatic text summarization is reviewed, specifically in the context of software bug reports.

One approach to text summarization is to use keywords to identify important information in the text. Sun, et al. [9]. proposed an automatic keyword and sentence-based text summarization approach for software bug reports. The proposed method uses a supervised learning approach to identify important keywords and sentences in the text and then generates a summary based on these keywords and sentences. The experimental results show that the proposed method outperforms existing methods in terms of precision, recall, and F-score.

Another approach to text summarization is to use lexicalsemantic keywords to identify important information in the text. Liu, et al. [12] proposed an extractive automatic text summarization method based on lexical-semantic keywords. The proposed method uses a hybrid of unsupervised and supervised learning techniques to identify important keywords and sentences in the text. The experimental results show that the proposed method outperforms existing methods in terms of precision, recall, and F-score.

In addition to using keywords, another approach to text summarization is to use dynamic feature space mapping to identify important information in the text. Chen, et al. [13] proposed an extractive document summarization method based on dynamic feature space mapping. The proposed method uses a neural network approach to learn a dynamic feature space that can capture the most important features of the text, and then generates a summary based on these features. The experimental results show that the proposed method outperforms existing methods in terms of precision, recall, and F-score.

Akhmetov, et al. [8] proposed a method for summarizing scientific articles using a greedy extractive summarization algorithm. Here the text summarization is defined as an optimization problem. ROUGE-1 and ROUGE-2 scores were used to check the uniqueness of the summaries created using different methods. Also, a greedy algorithm was used for the same task which takes sentences that contains a maximum number of words from the abstract of the given text document. One approach to automatic summarization is the use of a combined extractive and abstractive model. Liu et al. [12] proposed a model that uses extractive techniques to identify important sentences from the input text and abstractive techniques to generate a summary. Their model achieves state-ofthe-art performance on several benchmark datasets.

Another recent approach is the use of multimodal techniques to summarize both textual and visual information. Chen et al. [14] proposed a hierarchical recurrent neural network that uses attention mechanisms to jointly summarize both textual and visual information. Their model is able to generate informative and coherent summaries that capture the key information from both modalities.

In addition to these more specific approaches, researchers have also studied various techniques for extractive summarization based on lexical and semantic analysis. Zare and Montazeri [15] proposed a method that uses lexical and semantic analysis to identify important keywords and phrases from the input text, which are then used to generate a summary.

Finally, a recent survey by Mridha, et al. [16] provided a comprehensive overview of the recent advancements and challenges in the field of automatic text summarization. The survey covers a wide range of techniques, including both extractive and abstractive methods, as well as the use of deep learning and multimodal techniques.

In conclusion, automatic text summarization is a challenging problem that has received a great deal of attention in recent years. Researchers have proposed a variety of techniques, including extractive and abstractive methods, as well as multimodal approaches that incorporate both textual and visual information. While significant progress has been made in the field, there are still many challenges to overcome, including the need for more accurate and efficient models that can handle the complexity of real-world text data.

III. Proposed Method

This section describes the details of the proposed method, which is a model to create a basic structure of a presentation based on the text report given by the user.

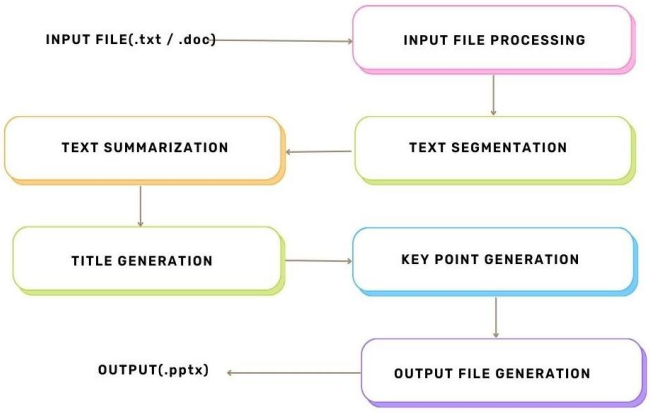

Fig. 1. Block diagram of proposed method

A. Dataset

1) *ArXiv Dataset*: The ArXiv dataset is used to train and test our model. For more than 30 years, ArXiv has benefited the general public and the research communities by providing free access to scholar articles in a wide range of fields, including math, statistics, electrical engineering, quantitative biology, economics, and the many aspects of physics. The 1.7 million articles in the arXiv dataset include relevant details including article titles, authors, categories, abstracts, full-text, PDFs, and more. With the use of this data set, researchers can explore more sophisticated machine learning approaches that combine multi-modal variables for applications including trend analysis, paper recommendation systems, category prediction, co-citation networks, knowledge graph creation, and semantic search interfaces.

B. Functional Components

1) *Input File Processing*: The input file refers to any document that is supposed to be converted to a presentation. A text file or a pdf file can be given as input to the model. The first step is to check if the file is a valid one and is not corrupted. If the file is valid, the text is extracted from the file. If the document is a structured document, with titles and subtitles, the cleaning process can be avoided. This text can be directly converted into segments based on the structure. If the text is unstructured, it is preprocessed where the text is converted to vectors. All the unwanted data is cleaned and this data is then passed to the next step.

2) *Text Segmentation*: This section is to separate the document into different segments based on content. Structured and unstructured documents can be identified by looking for possible headings in the document, which can be i is structured, each section can be treated as a segment. Each segment can then be fed to the summarization model.

If the document is unstructured, this model will be getting the data as vectors. In this case, the data will be treated differently. The document will be read from the beginning to the end and each sentence will be analyzed. The segmentation model uses a sliding window that divides the document into segments, in which each segment contains sentences that have a common topic. Each segment thus created is fed into the summarization model.

3) *Text Summarization*: Summarization is done in order to get the most important information from the input text. Each segment is summarised separately to ensure no data is lost. GPT-2 is used to achieve this task.

GPT-2 is pre-trained on a large dataset of text and can then be fine-tuned on specific tasks, such as translation, question answering, or text summarization. For text summarization, the model is trained on a dataset of input-summary pairs, where the input is a long piece of text and the summary is a shorter version that captures the main points of the input. To generate a summary, the model takes the input text and processes it through its layers of neural networks, which consist of attention mechanisms and transformer blocks. The attention mechanisms allow the model to focus on specific parts of the input text and the transformer blocks help the model understand the relationships between the words and sentences in the text. The model then generates a summary by predicting the next word in the sequence, starting from the beginning of the summary. It does this using the probabilities of the next word given the previous words, which are learned during training. The model continues generating words until it reaches the end of the predetermined summary length.

4) *Title Generation*: For each segment that is summarized, a title is generated. Each summarized segment is fed to another GPT-2 model. The model then generates a title by predicting the next word in the sequence, starting from the beginning of the title. It does this using the probabilities of the next word given the previous words, which are learned during training. The model continues generating words until it reaches the end of the title or a predetermined title length. To train GPT-2 for generating titles, a large dataset of input-title pairs is used, where the input is a text document and the title is a short phrase that summarizes the main content of the document. The model is then fine-tuned on this dataset to learn to generate titles that accurately reflect the content of the input text. It can generate multiple titles based on the contents and selects the most suitable title.

5) *Key Point Generation*: Key point generation refers to the conversion of summaries to points. In the presentation, different slides are created dynamically based on the amount of data. The titles generated before and their respective content is placed on each slide in order.

6) *Output File Generation*: The output file will be in .pptx format which can be further edited by the user if needed. Here, Google App Script is used to write the generated content into Google Slides. Google Apps Script is a scripting platform developed by Google for lightweight application development in the Google Workspace platform. Google Apps Script lets you programmatically create and modify Google Slides presentations using the Slides service. You can use Apps Script to add custom menus, dialogs, and sidebars to Google Slides. You can also integrate Slides with other Google services like Calendar, Drive, and Gmail. This can be edited using Google Slides, download as a file with .pptx extension or save into Drive.

REFERENCES

[1] P. Mishra, C. Diwan, S. Srinivasa and G. Srinivasaraghavan, **"Automatic Title Generation for Text with Pre-trained Transformer Language Model,"** 2021 IEEE 15th International Conference on Semantic Computing (ICSC), Laguna Hills, CA, USA, 2021, pp. 17-24.

[2] A. Ramisa, F. Yan, F. Moreno-Noguer and K. Mikolajczyk, **"BreakingNews: Article Annotation by Image and Text Processing,"** in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 5, pp. 1072-1085, 1 May 2018.

[3] P. Mahalakshmi and N. S. Fatima, **"Summarization of Text and Image Captioning in Information Retrieval Using Deep Learning Techniques,"** in IEEE Access, vol. 10, pp. 18289-18297, 2022.

[4] M. Y. Saeed, M. Awais, R. Talib and M. Younas, **"Unstructured Text Documents Summarization With Multi-Stage Clustering,"** in IEEE Access, vol. 8, pp. 212838-212854, 2020.

[5] D. Shirafuji, R. Rzepka and K. Araki, **"Argument Extraction for Key Point Generation Using MMR-Based Methods,"** in IEEE Access, vol. 9, pp. 103091-103109, 2021.

[6] S. G. Jindal and A. Kaur, **"Automatic Keyword and Sentence-Based Text Summarization for Software Bug Reports,"** in IEEE Access, vol. 8, pp. 65352-65370, 2020.

[7] S. Ghodratnama, A. Beheshti, M. Zakershahrak and F. Sobhanmanesh, **"Extractive Document Summarization Based on Dynamic Feature Space Mapping,"** in IEEE Access, vol. 8, pp. 139084-139095, 2020.

[8] I. Akhmetov, A. Gelbukh and R. Mussabayev, **"Greedy Optimization Method for Extractive Summarization of Scientific Articles,"** in IEEE Access, vol. 9, pp. 168141-168153, 2021.

[9] X. Sun and H. Zhuge, **"Summarization of Scientific Paper Through Reinforcement Ranking on Semantic Link Network,"** in IEEE Access, vol. 6, pp. 40611-40625, 2018.

[10] Shusheng Xu, Xingxing Zhang, Yi Wu, Furu Wei, and Ming Zhou. 2020. **"Unsupervised Extractive Summarization by Pre-training Hierarchical Transformers"**. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1784–1795, Online. Association for Computational Linguistics.

[11] Hananto, Valentinus Roby, Uwe Serdult, and Victor Kryssanov. 2022.¨ **"A Text Segmentation Approach for Automated Annotation of Online Customer Reviews, Based on Topic Modeling"** Applied Sciences 12, no. 7: 3412.

[12] W. Liu, Y. Gao, J. Li and Y. Yang, **"A Combined Extractive With Abstractive Model for Summarization,"** in IEEE Access, vol. 9, pp. 43970-43980, 2021.

[13] Chen, Jingqiang & Zhuge, Hai, (2017). **"Automatic generation of related work through summarizing citations."** Concurrency and Computation: Practice and Experience. 31. e4261. 10.1002/cpe.4261.

[14] Jingqiang Chen and Hai Zhuge, 2018. **"Abstractive Text-Image Summarization Using Multi-Modal Attentional Hierarchical RNN."** In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4046–4056, Brussels, Belgium. Association for Computational Linguistics.

[15] A. Hern´andez-Casta´neda, R. A. Garc˜ıa-Hernandez, Y. Ledeneva and C.´ E. Millan-Hern´andez, **"Extractive Automatic Text Summarization Based on Lexical-Semantic Keywords,"** in IEEE Access, vol. 8, pp. 4989649907, 2020.

[16] M. F. Mridha, A. A. Lima, K. Nur, S. C. Das, M. Hasan and M. M. Kabir, **"A Survey of Automatic Text Summarization: Progress, Process and Challenges,"** in IEEE Access, vol. 9, pp. 156043-156070, 2021.