WordPlay

An Assignment Submitted

in partial Fulfilment of the requirements

for the Degree of

Bachelor of Technology

in

Computer Science and Engineering

As a part of the core elective course

"Natural Language Processing and Text Analytics"

By

Chittepu Rohith Reddy - 200378

Maram Anvitha – 200399

Jugal Ganesh - 200358

BML MUNJAL UNIVERSITY™

Under the supervision of

Dr. Atul Mishra

SCHOOL OF ENGINEERING AND TECHNOLOGY

BML MUNJAL UNIVERSITY GURGOAN

May 3, 2023

WordPlay - A tool for Grammar check, Paraphrase and Summarize built using NLP techniques

Maram Anvitha . Jugal Ganesh . Chittepu Rohith Reddy

Abstract

Language processing tools have become increasingly popular as more and more people rely on digital platforms for communication and content creation. With the vast amount of information available online, it can be challenging to produce high-quality content without the aid of language processing tools. This study describes how to incorporate language processing tools, such as a grammar checker, summarizer, and paraphraser, into a website, named WordPlay. Gramformer, a cutting-edge language model that can spot and fix grammatical problems, was used to build the grammar checker. The Natural Language Toolkit (NLTK), which can produce succinct summaries of lengthy texts, is used to build the summarizer. The paraphraser was created using an encoder-decoder model which has the pipeline architecture and transformers, which can produce paraphrases of text with varying degrees of resemblance. The backbone for building these models is the various concepts of natural language processing like tokenization, stop words, stemming. Users may have a more simplified and effective means of enhancing their writing if these tools are included on a website. While the summarizer can help users create succinct summaries of lengthy texts, the grammar checker can help users find and correct grammatical errors in real-time. The paraphraser will be helpful for producing different versions of the same content or avoiding plagiarism. Users will be able to enter text into the tools on the website. Then, the tools can give the user feedback in real-time, flagging potential mistakes, offering summaries or paraphrases, and recommending different word choices or ways of phrasing. Overall, the incorporation of these language processing tools can improve a website's user experience by giving users access to strong language tools that can help them write better and faster.

Keywords Paraphrase, Summarize, Grammar check, WordPlay, Transformer, Gramformer, NLTK, Tokenization, Encoder-Decoder model, Stopwords, Stemming.

1 Introduction

In the modern Internet age, writing a complete document is difficult since it is time-consuming and has a poor vocabulary that includes unnecessary text and meaningless phrases. The main challenge of processing and comprehending the vast amount of data on the internet is to condense it into textual data; this can be done by using summarization [1]. While writing any documents or papers, one of the biggest problems is spelling and grammatical errors. Here comes the role of grammar checker [2]. Paraphrasing [3] is a technique used to rephrase the given input in order to deliver the same concepts in a clear manner by avoiding unwanted text and meaningless sentences.

WordPlay helps you to write better, faster, and smarter. This tool will paraphrase and summarise your sentence, paragraph, essay, or article according to your preferences with just one click. It includes a wide range of choices to change and improve the modified text. A tool like WordPlay, helps the users to create new content from the existing text in the writing process. WordPlay helps you become more efficient while also ensuring that your language, accent, and style are suitable for any situation. Simply type your words into the input field, and this tool will collaborate with you to generate the finest possible outcome. Other features of WordPlay include a grammar checker that helps in spell check and grammar correction of a sentence.

All the students aim for better marks and never want to lose grades because of linguistic and grammatical errors. Some of the platforms, like Quill Bot [4], cruxe.in, prepostseo.com, and paraphraser.io, are currently available. Through these, we can perform various tasks like grammar check, paraphrase, summarise, translation, and plagiarism check. It works by analysing the input text using its NLP algorithms and identifying key concepts and ideas.

Several years ago, writing skills were completely dependent on the writer's skill level. Now, with the use of all the available platforms, the quality of writing has definitely increased. Before Quill Bot, several technologies existed, such as text-to-speech software, spelling and grammar checkers, and summarising software. Whereas, Quill Bot uses several technologies such as NLP, ML and sentiment analysis [4]. According to the most recent statistics, there are aspects lacking, including difficulties with data privacy, expensive subscriptions, an unfriendly user interface, etc.

The main focus of this paper is to study and propose an NLP-based platform that will enable users to rephrase a sentence or simplify the text into the fewest possible words without changing its meaning while also checking its grammar. And as the research progresses, we would like to put this into practice in a sequential manner. First, we will work on paraphrasing, which will be helpful in rephrasing lengthy sentences or paragraphs by sticking to their original meaning. Then, we would like to include a summariser and a grammar check along with it.

For paraphrasing any text, most commonly we use the word-substitution method, in which the algorithm swaps out some words with their similar ones and uses other techniques such as sentence splitting, reordering, etc, without altering the meaning of the original statement. In the text summariser, we keep the core idea of the text while summarising the input document. We can achieve this with the use of abstractive summarisation [5], where we will be able to rephrase words that weren't in the original text and create new phrases to replace lengthy sentences. For texts that do not contain any complex words, we can use an extractive summariser [6], which is determined by linguistic and statistical characteristics. In grammar check, we generally look for misspelt words and grammatical errors in sentences. It can be done by using syntax, statistics, or rule-based models [2].

2 Literature review

Two factors can be used to evaluate the quality of the paraphrased text. When we paraphrase the texts, we frequently just handle two issues: one is its fidelity, and the other is its diversity, we generally solve only fidelity but Song et al. addressed the issue of diversity in scenario of neural conversation models using Determinal point procedures. [7] Similarly, Elhamifar et al's research of subset selection based on Simultaneous Sparse Recovery [8] also addresses the issue of diversity. These models assist us in increasing diversity, but they are constrained in terms of maintaining fidelity to the source sentence.

Pegasus [9] is a pre trained model from the library of Hugging Face [10] Transformers to rephrase words. Instead with the parrot [11] library through Kandi kit which will set up a working sandbox application with all the required pre requisites and resources, where this parrot library uses sequence to sequence method and tokenizers from transformers hugging face. this gives us control over the text's adequacy, fluency and diversity.

Kaiz Merchant et al. [12] proposed a text summarising tool that automatically creates short summary from long sentences. They used Latent semantic analysis (LSA), a method of NLP, to extract concepts from a single document. They created two models, one of them made use of training by enabling the single execution of multi-document criminal summaries into a single programme.

The alternative approach is untrained LSA model which is executed a fresh set of civil summaries iteratively, passing them to the programme for each document. They used ROGUE scores to evaluate the outcomes in part. They were able to obtain average ROUGE-1 (1-gram) and ROUGE-2 (2-gram) scores of 0.5 and 0.15, respectively. However, there are several limitations to this evaluation strategy. Sometimes there is a gap in the overall continuity when using this LSA approach. They must change their suggested system to include new ideas in addition to LSA in order to improve their model.

Abu Kaisar Mohammad Masum et al. [13] being able to produce a concise, fluent summary while minimising training loss. They produced a straightforward summary of the review using the Kaggle-accessible Amazon Fine Food Reviews dataset as their input data. In the decoding layer, they employed an attention model, and in the encoding layer, a bi-directional RNN using LSTMs.

They also employed the sequence-to-sequence approach to successfully complete this work of food descriptions. Text processing, vocabulary counting, missing word counts, word embedding, the effectiveness of the model, and minimising the amount of loss are some of the difficulties that come with employing an abstractive text summarizer. Only a tiny portion of reviews can be accurately predicted by the model, but for the remainder, it delivers great performance and succinct summaries. They have successfully decreased the training loss with a value of 0.036.

Prithiviraj Damodaran [14] created a library Gramformer that presents 3 separate interfaces to a collection of algorithms to detect, highlight, and correct grammar errors. It has a quality estimator to help with correction quality. The type used by the grammar error detector model is a classifier that returns a label, and that of the grammar error corrector model is sequence to sequence that returns the corrected sentence. C4-based synthetic pairs and PIE synthetic pairs were used to generate datasets. Using this library poses challenges as it works at sentence levels and trained on 64 length sentences. So, this is not yet appropriate for long paragraphs.

Grammatical Error Correction (GEC) [15] is a system that performs a sequence-to-sequence task by training a transformer model to receive an incorrect sentence as input and produce a grammatically correct sentence in return. The C4 200M dataset was used to train the grammar checker. Here, they first established the distribution of the relative sorts of grammatical faults that are encountered. Subsequently, the corruptions were conditioned on the error type when they were generated. The T5 model, which is a text-to-text model from Google, is used for training. Tokenizing, sequence to sequence, monitoring, and model evaluation are the steps needed to provide training. Finding data with a good diversity that accurately simulated the errors made in written languages was one of the difficulties they encountered. If the corruptions are random, then the distribution of errors that occur in actual use scenarios would not be represented by them.

Using a parrot library is preferable than the other ways for paraphrasing because it allows us to regulate both fluency and diversity. The use of an automated text summary method is preferred since it can produce summaries that are concise. When it comes to grammar checking, using the Gramformer library is preferable because it has been trained on multiple datasets.

3 Methodology

The system we propose will provide an interface for each of the three wordplay characteristics that we want to use. For Paraphrasing we use the pre-trained library Pegasus.

3.1 Dataset

Under these datasets, Pegasus paraphrasing has been tuned, and they are

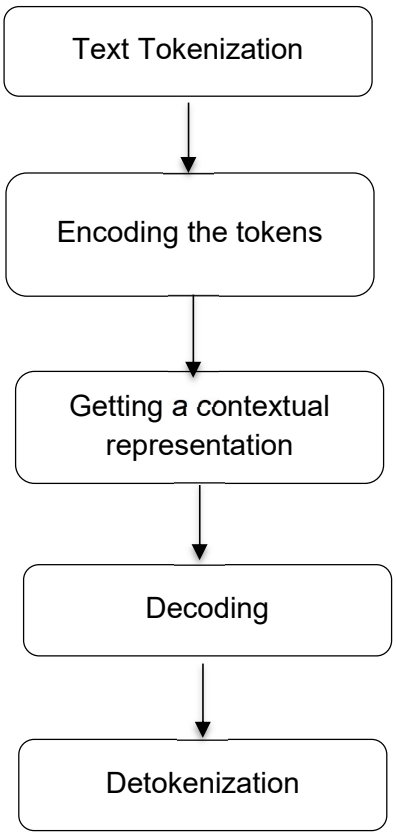
CC-News Dataset this was chosen because it offers a sizable and varied amount of training data that can be used to train high-quality language models like Pegasus, and it covers a wide variety of news articles from many sources and disciplines. A subsection of this dataset that was preprocessed to include sentence pairings with comparable meanings and this subset was used to train the Pegasus paraphrase model.

Pegasus is also trained on the **BooksCorpus Dataset**, which contains more than 11,000 books and more than 800 million words of text, and also with **English Wikipedia Dataset** which contains 2 billion words of text in order to understand a wide variety of linguistic patterns and semantic correlations.

3.2 Model

The Pegasus paraphrase model is a transformer-based model that has been pre-trained to particularly paraphrase natural language material. It employs a sequence-to-sequence model architecture with an encoder-decoder framework. A sizable corpus of sentence pairs, each consisting of a source sentence and a paraphrase of that sentence, is used to fine-tune the Pegasus model.

The input text is initially tokenized, into a series of small meaning units called sub words, which can be prefixes, suffixes, words, or other components of a word. In order to accomplish this, the Pegasus framework's AutoTokenizer module is used. The tokenized input sequence is then fed into the Pegasus model's encoder, which creates a series of hidden representations that capture the context and meaning of each sub word unit. This is accomplished by using the AutoModelForSeq2Seq library in the Pegasus framework. This model is trained to predict the next most probable sentence.



The Pegasus model's decoder then creates a series of output sub word units that correspond to the paraphrased text by predicting the next most probable token at each decoding step and this is done using the hidden representations the encoder produced.

Coming to the next one which is T**ext summarizing**. It is the technique of automatically creating summaries in natural language from an input document while keeping the key ideas. Textual data is getting bigger these days. We need to find a technique to condense this material while keeping its information and meaning. For that, textual information must be summarized.

We use two different types of algorithms to summarize the text that has been given as input. Extractive summarization is a technique used in NLP to extract the most important information or key phrases from a document to create a summary. It is used in applications such as news article summarization. Abstractive summarization is a type of text summarization in which it requires natural language generation and a deeper understanding of the text, capturing the most important information in the original text.

Extractive summarization is faster and easier to implement than abstractive summarization, as it involves selecting sentences or phrases directly from the original text, preserving the exact meaning of the original content and avoiding errors or inaccuracies.

Text summarization using Python libraries: nltk

- Lexical analysis: Word and text tokenizer
- N-gram and collocations
- Parts-of-speech tagger
- Tree model and Text chunker for capturing
- Named-entity recognition

The text summary is generated by selecting the top n sentences with the highest scores, where n is the number of sentences requested by the user. The score of a sentence is determined by the frequency of its words in the entire text. The higher the frequency of words in a sentence, the higher the score of the sentence.

The following steps were taken to generate the text summary:

Tokenization: The text is first tokenized into sentences using the sent_tokenize function from the NLTK package. Each sentence is then tokenized into words using the word_tokenize function.

Stopword Removal: Stopwords are common words such as "the", "and", "in", etc., which do not contribute much to the meaning of a sentence. The NLTK package provides a list of stopwords for the English language, which are removed from the list of words using a for loop.

Stemming: The words are then stemmed using the Porter Stemmer algorithm from the NLTK package. Stemming is the process of reducing words to their root form, which helps to group together words with similar meanings.

Word Frequency Calculation: The frequency of each word in the entire text is calculated using a dictionary data structure. If a word is not already in the dictionary, it is added with a frequency of 1. If the word is already in the dictionary, its frequency is incremented by1.

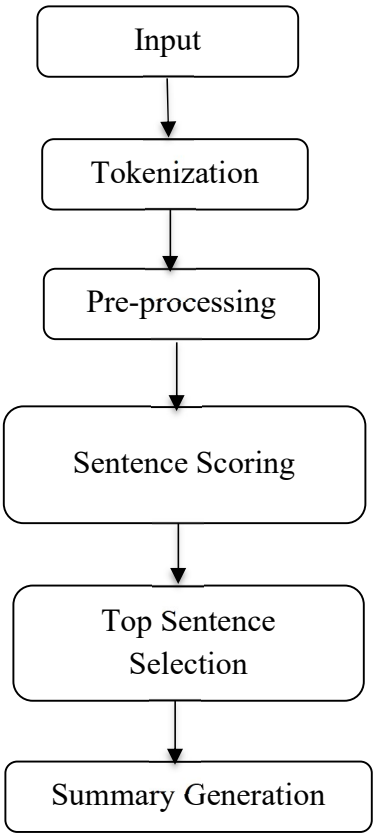
Sentence Score Calculation: For each sentence, the scores of its words are added together to get the sentence score. The score of a word is determined by its frequency in the entire text. The score of a sentence is the sum of the scores of its words.

Selection of Top Sentences: The sentences with the highest scores are selected based on the requested number of sentences. The sentence scores are sorted in descending order using the sorted function, and the top n sentences are selected.

Summary Generation: The selected sentences are joined together to form the text summary using the join function.

In conclusion, the described above outlines the steps taken to generate a text summary using natural language processing techniques. The code is written in Python and uses several packages from the NLTK and Rouge packages for evaluation. The quality of the generated summary can be evaluated using the Rouge package.

Model Working:



Abstractive summarization is an NLP task that generates a condensed version of a long text while retaining important information. Pre-trained transformer models, such as PEGASUS, have achieved state-of-the-art performance on this task.

Purpose of Libraries: The three libraries that we are using - torch, transformers, and sentencepiece - serve different purposes in our text summarization task.

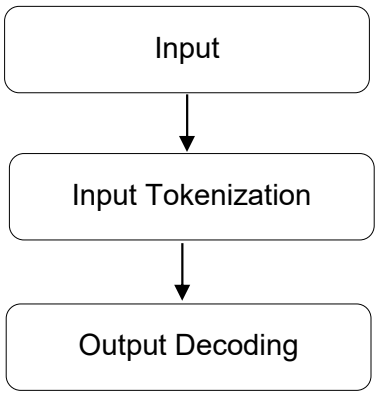
- torch is a machine learning library that provides support for training and inference of deep neural networks. We will use it to fine-tune the pre-trained PEGASUS model on our dataset.
- transformers are a library that provides access to pre-trained transformer models, such as PEGASUS. It also provides tools for fine-tuning these models on new datasets.
- sentencepiece is a library for tokenization, which we will use to tokenize our input text into smaller, manageable pieces that the PEGASUS model can process.

Purpose of Pre-trained PEGASUS Model: We are using the pre-trained PEGASUS model because it has been specifically designed for abstractive text summarization tasks. It was trained on a large corpus of web pages, news articles, and scientific papers using a denoising autoencoder objective. The objective of the PEGASUS training process is to reconstruct the original text from a corrupted version, which forces the model to learn a general representation of the input text that is capable of capturing its most salient features. This pre-training process allows the PEGASUS model to achieve state-of-the-art performance on a wide range of summarization tasks without requiring extensive fine-tuning on new datasets.

Tokenization: Before we can summarize our input text using the PEGASUS model, we need to tokenize it using the PegasusTokenizer class from the transformers library. Tokenization involves breaking up the input text into smaller pieces, or tokens, that the model can process. We can choose from different tokenization methods, such as byte-pair encoding (BPE) or WordPiece, depending on our specific use case.

Summarization: Once we have tokenized our input text, we can pass the resulting tokens to the PegasusForConditionalGeneration class from the transformers library to generate a summary using the PEGASUS model. The PEGASUS model uses an encoder-decoder architecture, where the encoder processes the input tokens and generates a hidden representation, and the decoder uses this representation to generate the summary tokens. We can choose from different decoding methods, such as beam search or sampling, to control the quality and length of the generated summary.

Decoding: Finally, we can use the PegasusTokenizer.decode method from the transformers library to decode the summary tokens back into human-readable text. The decode method maps the summary tokens back to their corresponding words and reconstructs the original text with appropriate punctuation and capitalization.



Coming to the next one, **Grammar Checker**. To complete this task, we will use a library called Gramformer. These days, grammatical errors while writing have been the most common issue. In order to overcome that, a grammar checker plays an important role.

For fine-tuning the model, the following techniques were used:

- Use, modify, and filter WikiEdits.

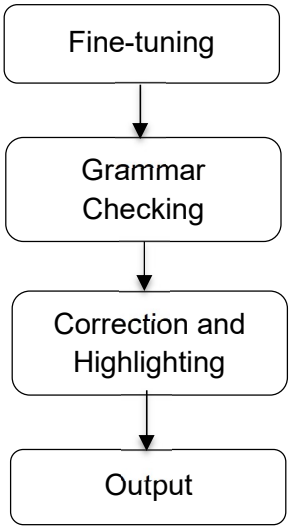
WikiEdits is a group of scripts for automatically extracting altered sentences from test edition histories, such as Wikipedia revisions.

- By using C4 - based synthetic pairs.

C4: For the purpose of correcting grammatical errors, this dataset contains artificial training data.

- Use PIE synthetic pairs.

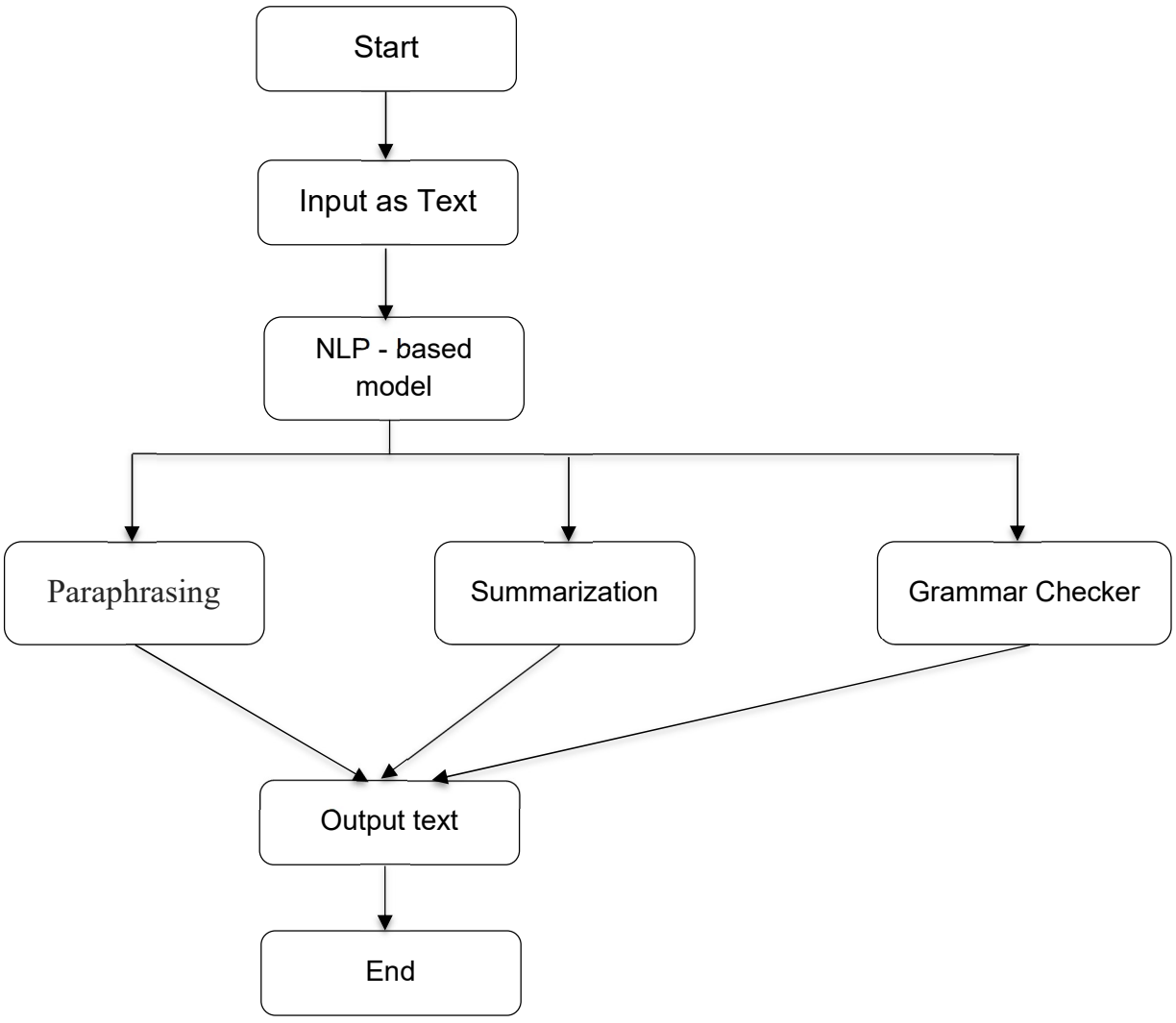
PIE (Parallel Iterative Edit Models for Local Sequence Transduction) is an architecture for tasks like Grammar Error Correction.



Gramformer's training uses data augmentation methods such noise addition, artificial mistake generation, and reverse translation. These methods aid in broadening the training set's diversity and enhancing the model's adaptability to various sorts of input text.

Building a grammar checker using the Gramformer library over other systems like grammatical error correction is preferable as it exposes separate algorithms for detection, highlighting, and correction. In general, all generative models sometimes have the tendency to produce inaccurate text, which is something we cannot control. Since Gramformer is a generative model, there are high chances of producing spurious text. So, a quality estimator to make sure the quality of recommended corrections and highlights will be added.

3.3 Architecture / Framework:



4 Results

The proposed system aims to provide an interface for three wordplay characteristics - paraphrasing, text summarization, and grammar correction. To evaluate the effectiveness of the tool, we took a sentence and calculated the accuracy of the various sentence lengths. We calculated the paraphrasing accuracy for various sentence lengths. For the grammar check, we calculated the performance-built model by calculating three parameters: accuracy, average distance, and average error rate. For the summary, we calculated the Rouge package using multiple metrics, including ROUGE-1, ROUGE-2, and ROUGE-L, which evaluates the degree to which the generated summary and the reference text overlap.

The user can see the efficiency of the website as the user can select different options, grammar-checker, summarizer, paraphraser in the home page only. Learnability, memorability can be experienced by the user.The flow of the website attracts the user promoting its usability. The below fig.1 shows the home page layout of the website.

Fig.1 Home Page of the Website

To check the accuracy of the paraphrased text, we are comparing the evaluated text to the original text and calculating the proportion of the sentence that has been modified by this.

We will know how well the model is working, and to evaluate the model, we are checking it for multiple length sentences.

Sentence Length	**Percentage Changed**
10	86.6
12	89.7
15	93.2

Table. 1 : Accuracy values of the paraphrase tool

For example, if we have given the text

“`nlp is tedious task and to learn nlp one needs to have a lot patience and confidence in himself and needs to pay attention`”

Here, the sentence goes through the model in the following ways. First, it gets tokenized, and then, through the encoder and decoder, we get the following output.

```
'One needs to pay attention and have a lot of patience to learn nlp.'
```

To evaluate the performance of the generated summary the Rouge package is used by comparing it to a reference text provided by the user. The Rouge package calculates several metrics such as ROUGE-1, ROUGE-2, and ROUGE-L, which measure the overlap between the generated summary and the reference text.

Sentence Length	ROUGE-1	ROUGE-2	ROUGE-L
34	65.5%	43.7%	65.5%
69	72.3%	48.5%	0.53%
87	71.42%	49.6%	63.4%

Table. 2 : Accuracy values of the summarizing tool

For example, from the above table let’s consider the sentence length 69 for deeper analysis

Input text :

text = "The benefits of exercise are well-known and include improved physical health, increased mental clarity, and reduced risk of chronic diseases such as heart disease and diabetes. However, many people struggle to find the motivation to exercise regularly. Finding an enjoyable form of physical activity, setting achievable goals, and enlisting the support of friends and family can all help to make exercise a more consistent part of one's daily routine."

The referenced text is:

ref_text = "Regular exercise has many benefits, including improved physical and mental health and reduced risk of chronic diseases. To stay motivated, try finding a physical activity you enjoy, setting achievable goals, and seeking support from friends and family."

Output is :

```
Finding an enjoyable form of physical activity, setting achievable
goals, and enlisting the support of friends and family can all help to
make exercise a more consistent. The benefits of exercise are well-
known and reduced risk of chronic diseases such as heart disease and
diabetes.
```

The reported scores are as follows:

1. ROUGE-1 score: 0.7234042503214125 - This is the overlap between unigrams (single words) in the generated summary and the reference summary.
2. ROUGE-2 score: 0.4859813034326142 - This is the overlap between bigrams (sequences of two words) in the generated summary and the reference summary.
3. ROUGE-L score: 0.5319148886192849 - This is a variant of ROUGE that considers the longest common subsequence between the generated summary and the reference summary, with the length of the subsequence weighted by the length of the reference summary.

To evaluate the performance of the proposed model of grammar-checker, different parameters like ***accuracy***, ***average distance***, and ***average error rate*** were used. Accuracy represents the percentage of sentences that were corrected accurately compared to the correct sentences provided. The average distance represents the average edit distance between the corrected sentence and the corresponding correct sentences. The average error rate represents the percentage of errors in the corrected sentences relative to the total number of characters in the correct sentences.

Number of words	Accuracy	Average distance	Average error rate
10	60%	2.60	5.80%
15	80%	0.60	0.99%
20	66.7%	0.33	0.29%

Table. 3 : Accuracy values of the grammar-checker tool

The example of the given input text and corrected text is shown below.

The incorrect sentences are:

```
influent_sentences = [
  "I do not watches TV as much as i used to when I was an kids.",
  "There is a lot of things I need to do before I can go homes.",
  "I have a lot of work to finishes upp before the end of the weak.",
  "I wondr whether or not Tom knew it was a bad idea to do that. ",
  "I seen the look on your face when Tom asked you to waited for Mary.",
]
```

The grammatically correct sentences are:

```
correct_sentences = [
  "I do not watch TV as much as I used to when I was a kid.",
  "There are a lot of things I need to do before I can go home.",
  "I have a lot of work to finish up before the end of the week.",
  "I wonder whether or not Tom knew it was a bad idea to do that. ",
  "I saw the look on your face when Tom asked you to wait for Mary.",
]
```

The corrected sentences for given influent sentences are:

[Correction] I do not watch TV as much as I used to when I was a kid.
[Correction] There are a lot of things I need to do before I can go home.
[Correction] I have a lot of work to finish up before the end of the week.
[Correction] I wondered whether or not Tom knew it was a bad idea to do that.
[Correction] I saw the look on your face when Tom asked you to wait for Mary.

The layouts of the grammar checker and summarizer on the website are shown below in Figs. 2 and 3.

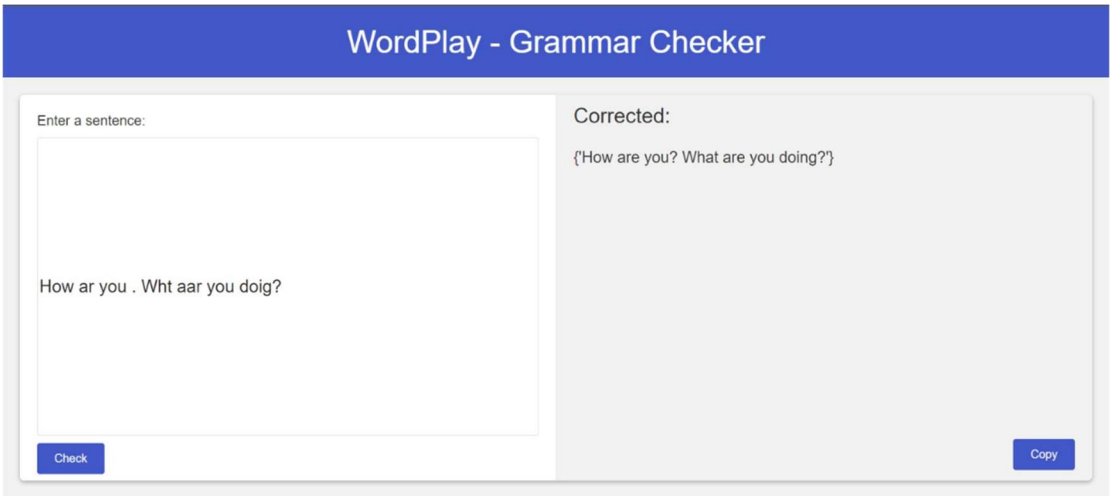


Fig. 2 Layout of the Grammar Checker

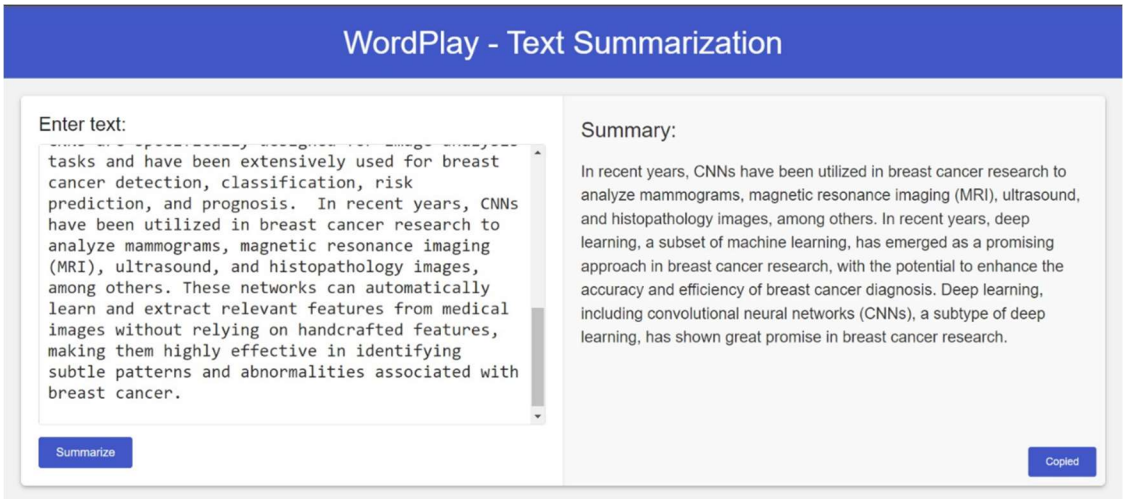


Fig. 3 Layout of the Text Summarization

5 Conclusion and future scope:

Paraphrasing, summarizing, and grammar checking serve one of the major purposes. So, the application must be error-free with accurate outputs. The paraphraser shows an accuracy of 86.6%, 89.7%, and 93.2% for sentences of word lengths 10, 12, and 15, respectively. Summarizer gives Rouge-1 and Rouge-2 values of 72.3% and 48.5%, respectively, and Rouge-L values of 0.53% for a sentence length of 69. The grammar checker shows an accuracy of 60%, 80%, and 66.7% for sentences of word lengths 10, 15, and 20, respectively. When summarizing, the model must comprehend the meaning of the text to be summarized. However, at the moment, our model only functions in extractive summarizing mode. At the moment, our website is functional for extractive summarizing and grammar checking. In the future, we will be integrating the paraphrasing tool into this website, which we couldn't do due to the models that have been used in paraphrasing. Also, in the future, we will be implementing abstractive summarizing and a translator.

6 References

[1] C. A. H. Pokharkar and D. P. P. Smita, “Text Summarizer using NLP,” pp. 25-27, 2022.

[2] N. Y. L. K. M. S. and N. L. T. , “Developing a Chunk-based Grammar Checker for,” pp. 245-254.

[3] V. Madaan, “A NOVEL APPROACH TO PARAPHRASE ENGLISH SENTENCES USING NATURAL LANGUAGE PROCESSING,” pp. 5117-5128, 2016.

[4] E. Y. K. and F. R. , “Post-Graduate Students’ Perceptions of Quillbot,” *JELTL (Journal of English Language Teaching and Linguistics),* p. 16, 2022.

[5] S. G. and S. K. G. , “Abstractive summarization: An overview of the state of the art,” vol. Volume 121, pp. Pages 49-65, 1 May 2016.

[6] Y. Liu, “Fine-tune BERT for Extractive Summarization,” 5 Sep 2019.

[7] A. K. M. Mohammad, S. S. A. M. A. I. Talukder and S. A. Akhter Hossain, “Abstractive method of text summarization with sequence to sequence RNNs,” *IEEE,* 2019.

[8] K. M. and Y. P. , “NLP Based Latent Semantic Analysis for Legal Text Summarization,” *IEEE,* pp. 1803-1807, 2019.

[9] P. Damodaran, “Github,” 27 December 2022. [Online]. Available: https://github.com/PrithivirajDamodaran/Gramformer.

[10] P. Dwivedi, “deep-learning-analytics,” 1 April 2022. [Online]. Available: https://deeplearninganalytics.org/nlp-building-a-grammatical-error-correction-model/.