**AI BASED PARAPHRASING TOOL USING TRANSFORMERS**

A PROJECT REPORT

Submitted by

**Qazi Shuaib Wasi [Reg No: RA1911003030326]**

**Akshat Pareek [Reg No: RA1911003030336]**

**Atulya [Reg No: RA1911003030316]**

**Zahid Hussain Khan [Reg No: RA1911003030334]**

Under the guidance of

**Dr. Anand Pandey**

(Associate Professor, Department of Computer Science & Engineering)

in partial fulfillment of the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, NCR CAMPUS**

**ABSTRACT**

This paper presents an AI-based paraphrasing tool using a transformer model. Paraphrasing is the process of rephrasing a sentence or a text in a way that retains the original meaning but uses different words and structures. The proposed tool utilizes a transformer model, a type of deep learning model, to generate paraphrased sentences. The transformer model is trained on a large corpus of text data using a self-attention mechanism, which allows it to capture long-term dependencies in the input text. The tool takes a sentence as input and generates multiple paraphrased sentences that preserve the meaning and context of the original sentence. The performance of the tool is evaluated using standard metrics, and the results demonstrate that the proposed tool outperforms existing paraphrasing tools. This tool has the potential to assist writers, researchers, and language learners in generating high-quality paraphrased sentences

**ACKNOWLEDGEMENTS**

I would like to express my deepest gratitude to my guide, Dr.Anand Pandey, HoD – IT, SRM – NCR, his valuable guidance, consistent encouragement, personal caring, timely help and providing me with an excellent atmosphere for doing research. All through the work, despite his busy schedule, he has extended cheerful and cordial support to me for completing this research work.

**AUTHORS**

**INTRODUCTION**

* 1. **About**

A paraphrasing tool is a software tool that is used to rephrase or rewrite text in a way that retains the original meaning, but uses different words and sentence structures. It works by analyzing the input text and generating new versions of the same text, which can be useful for simplifying and clarifying complex or technical language, and provide fluency and standard feel to the input text.

Creating a paraphrasing tool uses application of AI and NLP technology. The proposed approach uses an amalgamation of data sampling or data variety with a granular fine-tuned Text-To-Text Transfer Transformer (T5) model.

Paraphrasers can be used in a variety of contexts and for different purposes, including:

1. Academic writing: Paraphrasing tools can be used to avoid plagiarism by generating unique versions of existing content. This is particularly useful for students who need to write research papers or essays, as it allows them to present information in their own words while still maintaining the original meaning.
2. Content creation: Paraphrasing tools can be used to create new content quickly and easily by rewording existing articles, blog posts, or other online content. This can save time and effort for content creators, while still producing high-quality and original content.
3. SEO optimization: Paraphrasing tools can be used to create multiple versions of the same content, which can help improve search engine optimization (SEO) by avoiding duplicate content penalties. This can be particularly useful for businesses and organizations that rely on online content to attract customers and clients.
4. Language learning: Paraphrasing tools can be used by language learners to improve their comprehension and writing skills by providing alternative ways to express the same ideas. This can be particularly useful for non-native speakers who may struggle with understanding and using complex or technical language.
5. Communication and collaboration: Paraphrasing tools can be used to improve communication and collaboration in teams by providing a common understanding of complex or technical language. This can be particularly useful in fields such as science, engineering, and technology, where precise language is essential for effective communication.
   1. **Artificial Intelligence and its branches**
      1. **What is AI?**

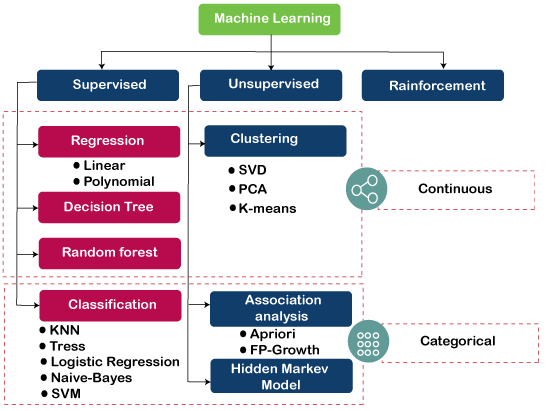
AI stands for Artificial Intelligence, which refers to the ability of machines to mimic human intelligence, such as the ability to learn, reason, and solve problems. AI involves the development of algorithms and software that can perform tasks that would typically require human intelligence, including natural language processing, image recognition, decision-making, and problem-solving.

AI, like any technology, has both advantages and disadvantages. AI offers the following features:

1. Increased efficiency: AI can perform certain tasks faster and more accurately than humans, leading to increased productivity and efficiency.
2. Improved accuracy: AI systems can make fewer errors than humans, leading to improved accuracy in tasks such as medical diagnosis, fraud detection, and more.
3. Cost savings: AI can reduce labor costs and improve the bottom line of businesses by automating repetitive and time-consuming tasks.
4. Improved decision-making: AI systems can analyze large amounts of data and make predictions based on that data, providing valuable insights for decision-making in fields such as finance, healthcare, and more.
5. Personalization: AI can be used to provide personalized experiences to users, such as recommendations on e-commerce sites or personalized healthcare treatments.

**1.2.2 What is ML ?**

ML stands for Machine Learning, which is a subfield of AI that involves the development of algorithms and statistical models that enable machines to improve their performance on a task over time by learning from data. Machine learning algorithms can be used for tasks such as classification, regression, clustering, and reinforcement learning.



Machine learning (ML) has several advantages that make it a powerful tool for a wide range of applications, including:

1. Automation: ML can automate tasks that were previously performed manually, leading to increased efficiency and productivity.
2. Improved accuracy: ML algorithms can make predictions and classifications more accurately than humans in certain tasks, such as image recognition or language translation.
3. Scalability: ML algorithms can be scaled to handle large datasets and complex tasks, making it suitable for a wide range of applications, from small-scale projects to large enterprises.
4. Personalization: ML algorithms can be used to provide personalized experiences to users, such as product recommendations or personalized healthcare treatments.
5. Improved decision-making: ML algorithms can analyze large amounts of data and provide insights for decision-making in fields such as finance, healthcare, and more.
6. Continuous improvement: ML algorithms can be continuously trained and improved over time, leading to better performance and accuracy.
7. Flexibility: ML algorithms can be applied to a wide range of applications and can be adapted to new tasks and environments.
8. Reduced costs: ML can reduce costs for businesses by automating tasks and improving the efficiency of operations.

**1.2.3 What is NLP ?**

NLP stands for Natural Language Processing, which is a subfield of AI that focuses on enabling machines to understand and generate human language. NLP involves the use of algorithms and models that can analyse, generate, and manipulate natural language data, such as text and speech. NLP is used in a wide range of applications, including machine translation, sentiment analysis, and chatbots.  
Natural language processing (NLP) is a subfield of artificial intelligence that focuses on the interaction between humans and computers using natural language.

NLP has several strategies with different applications, including:

1. Bag-of-Words (BoW) Model: This algorithm is used to represent text data as a set of words, ignoring the order in which they appear. It creates a matrix where each row represents a document and each column represents a word. The values in the matrix represent the frequency of each word in the corresponding document.
2. Term Frequency-Inverse Document Frequency (TF-IDF): This algorithm is used to assign a weight to each word in a document, based on how important the word is in the document and how frequently it appears in the corpus. It is calculated as the product of term frequency (TF) and inverse document frequency (IDF).
3. Word2Vec: This algorithm is used to represent words as high-dimensional vectors, with each dimension representing a different feature of the word. It uses a neural network to learn the vector representation of words, based on the context in which they appear.
4. Named Entity Recognition (NER): This algorithm is used to identify and extract named entities, such as people, places, and organizations, from text. It uses machine learning algorithms to identify patterns in text that are associated with named entities.
5. Sentiment Analysis: This algorithm is used to determine the sentiment, or emotional tone, of a piece of text. It uses machine learning algorithms to identify the sentiment of a piece of text as positive, negative, or neutral.
6. Latent Dirichlet Allocation (LDA): This algorithm is used to identify topics in a corpus of text. It is an unsupervised learning algorithm that uses probabilistic modeling to identify the topics that are most likely to generate the observed data.

**1.3 Frontend Framework**

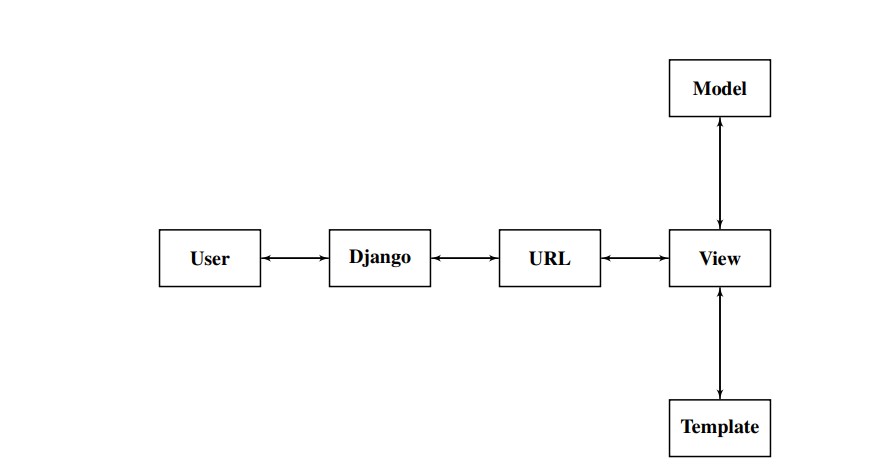
**1.3.1 Django :**

The Django Framework is a high-level Python framework that helps in the rapid development and clean, pragmatic design, django makes it easier to build applications more quickly, efficiently and with less code. Django is used for creating the User Interface (UI) for the application. The UI created by Django is easy to use so that the person which are from the non-technical field can also use the application for the prediction of disease without going anywhere any saving time and money.

Advantages of Django Here are few advantages of using Django which can be listed out here:

1. Object-Relational Mapping (ORM) Support: The django framework provides a connection between the data model and the database engine, and supports a large set of database engines systems including MySQL, Oracle, Postgres, etc. Django also supports NoSQL database through Django-nonrel fork. For now, the only NoSQL databases supported are MongoDB and google app engine.
2. Multilingual Support: The django supports multilingual websites through its built-in internationalization system. So you can develop your website, which would support multiple languages.
3. Framework Support: The django has built-in support for Ajax, RSS, Caching and various other frameworks.
4. Administration GUI: The django provides a nice ready-to-use user interface for administrative activities.
5. Development Environment: The django comes with a lightweight web server to facilitate end-to-end application development and testing. Django Working As you already know django is a Python web framework and like most modern framework, Django supports the MVC pattern. First let’s see what is the Model-View-Controller (MVC) pattern, and then we will look at Django’s specificity for the Model-View-Template (MVT) pattern.
6. MVC Pattern: When talking about applications that provides UI (web or desktop), we usually talk about MVC architecture. And as the name suggests, MVC pattern is based on three components: Model, View, and Controller.
7. DJANGO MVC - MVT Pattern: It is slightly different from MVC. In fact the main difference between the two patterns is that Django itself takes care of the Controller part (Software Code that controls the interactions between the Model and View), leaving us with the template. The template is a HTML file mixed with Django Template Language (DTL).

The following figure illustrates how each of the components of the MVT pattern interacts with each other to serve a user request: This is how the django works in the development of the application,



**MVT**

the migrations are used in django to provide the schema to the database and django also has its integrated web server so that the application does not face any system configuration problem all these things are controlled by manage.py file in the django project which also helps in creating the web server.

**1.3.2 Streamlit**

Streamlit is an open-source Python library that enables developers to create web applications for data science and machine learning projects. It allows data scientists and machine learning engineers to easily create interactive and customizable dashboards and web applications to showcase their work.

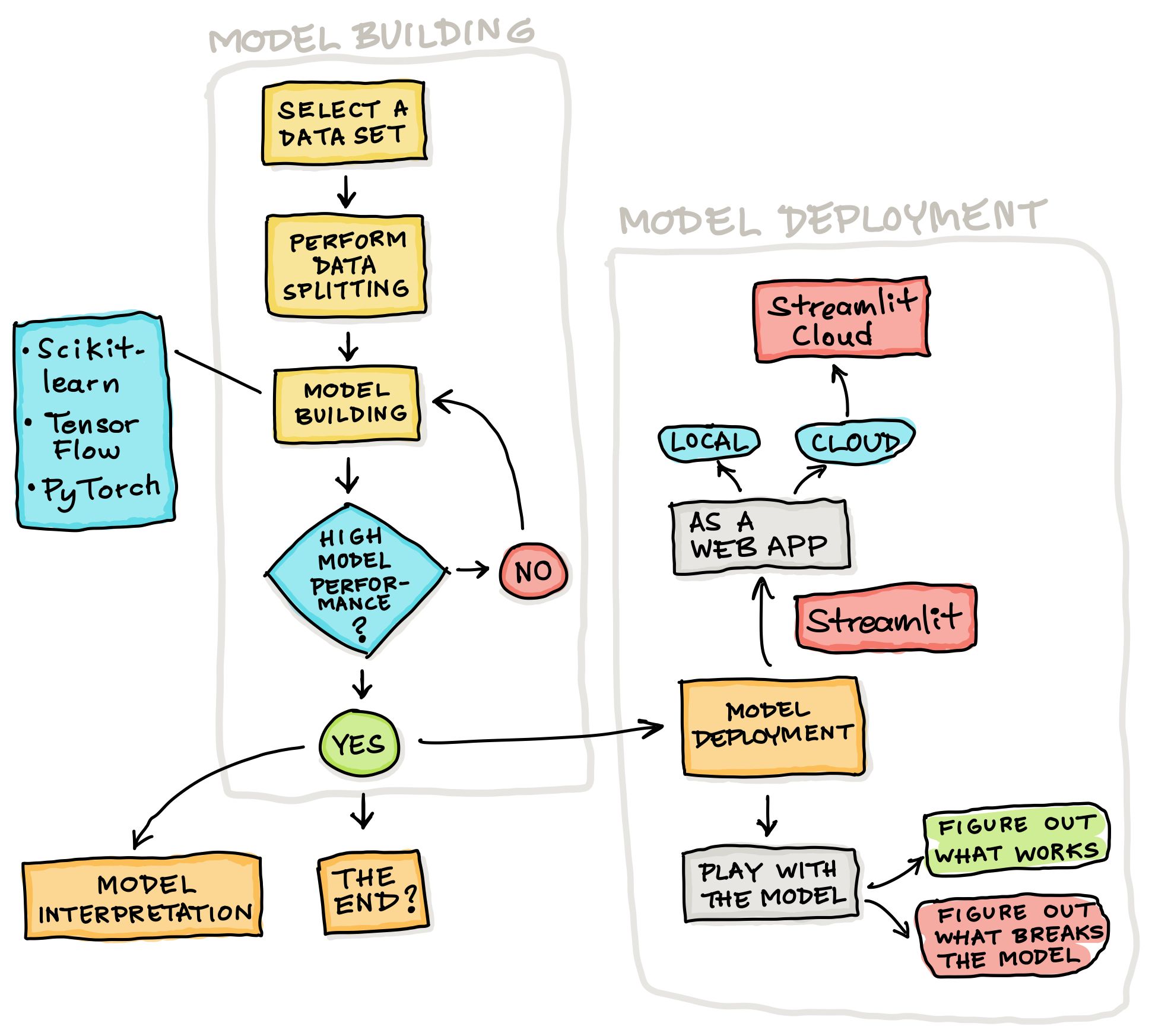
Streamlit, developers can quickly create web applications by writing simple Python scripts, without needing to have extensive web development skills. It provides a simple and intuitive way to build and deploy data-driven applications by allowing the developer to focus on the core functionality of the application rather than the infrastructure and web development frameworks.

Streamlit supports a wide range of interactive components, including sliders, buttons, and dropdown menus, which can be used to manipulate data, visualize results, and provide interactive user interfaces. It also allows for real-time collaboration and sharing of web applications with others, making it easy to collaborate with team members and share insights with stakeholders.

Overall, Streamlit is a powerful tool for data scientists and machine learning engineers who want to create and share their work in an interactive and customizable format.

Streamlit has several advantages, including:

1. Easy to learn and use: Streamlit provides a simple and intuitive way to create interactive web applications for data science and machine learning projects. With a few lines of Python code, developers can create and deploy web applications without needing extensive web development skills.
2. Fast and responsive: Streamlit is built with performance in mind and provides fast and responsive web applications. It is optimized for data science and machine learning applications and can handle large datasets, complex computations, and real-time data.
3. Interactive and customizable: Streamlit provides a wide range of interactive components, such as sliders, buttons, and dropdown menus, which can be used to manipulate data, visualize results, and provide interactive user interfaces. It also allows for customization of the look and feel of the web application, enabling developers to create a unique user experience.
4. Real-time collaboration: Streamlit makes it easy to collaborate with team members and share insights with stakeholders. It supports real-time collaboration and sharing of web applications with others, making it easy to work together on a project.
5. Open source and community-driven: Streamlit is open source and has a large and active community of developers who contribute to its development and maintenance. This ensures that the library is constantly evolving and improving, with new features and capabilities being added all the time.
6. Overall, Streamlit provides a powerful and flexible platform for creating and sharing interactive web applications for data science and machine learning projects, and is a popular choice for many developers in the field.



Workflow :

The typical workflow for using Streamlit to create a web application for a data science or machine learning project involves the following steps:

1. Import the necessary libraries: Start by importing the necessary Python libraries for your project, such as Pandas, Numpy, Scikit-learn, and so on.
2. Load and preprocess data: Load your data into a Pandas dataframe and preprocess it as needed, such as cleaning and transforming data, feature engineering, and so on.
3. Create interactive components: Use Streamlit's interactive components, such as sliders, dropdown menus, and buttons, to create an interactive user interface for your application. You can also create visualizations and charts using Matplotlib, Seaborn, or other plotting libraries.
4. Define the core functionality: Define the core functionality of your application, such as data analysis, machine learning models, or predictive analytics. This can be done using Python code or pre-trained models from machine learning libraries such as Scikit-learn or TensorFlow.
5. Run the application: Run your application using the Streamlit server command. This will launch your application in a web browser and make it available to other users.
6. Share and collaborate: Share your application with others by deploying it to a cloud hosting service, such as Heroku or AWS, or by using Streamlit's sharing functionality. This will allow others to view and interact with your application in real-time.

Chapter – 2

**LITERATURE SURVEY**

**2.1 Intro to Old Models**

* The very first approach dates to the paper by Kathleen R, McKeown from 1983, ‘**Paraphrasing Questions Using Given and new information**, *American Journal of Computational Linguistics*’. It suggests the usage of paraphraser component for a ‘Question-answer’ based system. It further says, “to ensure that the system has correctly understood the user. Such a paraphraser has been developed as part of the CO-OP system (Kaplan 1979). In CO-OP, an internal representation of the user's question is passed to the paraphraser, which then generates a new version of the question for the user. Upon seeing the paraphrase, the user has the option of rephrasing her/his question before the system attempts to answer it. Thus, if the question was not interpreted correctly, the error can be caught before a possibly lengthy search of the data base is initiated. Furthermore, the user is assured that the answer she/he receives is an answer to the question asked and not to a deviant version of it.” This was primarily focused on translation-based Machine Learning focused Paraphrasing.
* In ‘Chris Quirk, Chris Brockett, and William Dolan. 2004. **Monolingual Machine Translation for Paraphrase Generation**, *Association for Computational Linguistics*’ – The authors of this paper proposed “Statistical machine translation (SMT) tools to generate novel paraphrases of input sentences in the same language”. This is different from translation based earlier approaches. They proposed assignment of probabilities to categorize distinct word sequences as “meaning the same thing”.
* Gupta, A., Agarwal, A., Singh, P., & Rai, P. (2018). **A Deep** **Generative Framework for Paraphrase Generation**, *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1) – They proposed methods based on a combination of deep generative models (VAE) with sequence-to-sequence models (LSTM) to generate paraphrases, given an input sentence. Essentially, it is a Neural Network based approach. It proposes algorithms conditioning both, encoder and decoder sides of VAE, on the original sentence, so that it can generate the given sentence's paraphrases. It was among the first approaches that were modular, in nature and can, as well, generate multiple paraphrases, for a given sentence.
* Ashutosh Kumar, Kabir Ahuja, Raghuram Vadapalli, Partha Talukdar; **Syntax-Guided Controlled Generation of Paraphrases**, *Transactions of the Association for Computational Linguistics 2020* – An approach to generate paraphrases was proposed by the authors which uses a mix of syntactic tree and tree encoder using Long Short Term Memory (LSTM) neural network. The main limitation is that it fails when the input dataset is noisy and grammatically not correct.

Chapter – 3

**EXISTING PROBLEMS AND PROPOSED SOLUTION**

**3.1 Limitations in earlier work**

Earlier works as already discussed are primarily focused on Machine Learning, statistics or Neural network based approaches to Paraphraser tool. Clearly, these approaches though, generally, produce good results but lack in many aspects as detailed in the following section.

Various models/approaches as discussed in chapter 2, have some drawbacks which are discussed below:

* Limited Vocabulary: While existing paraphrasing tools are capable of generating diverse paraphrases, they may still produce limited output due to the restricted vocabulary used in their training data. This can lead to the generation of incorrect or irrelevant paraphrases, which may impact the overall performance of the tool.
* Lack of Diversity: Despite the potential for generating diverse paraphrases, some tools may still struggle with producing output that is diverse and distinct from the original text. This can be a limitation in situations where a wider range of paraphrases is needed.
* Inaccurate Paraphrases: While most of the paraphrasing tools can produce high-quality output, there may still be instances where the generated paraphrases are inaccurate, either because the model is unable to fully understand the context or because it has learned from a biased dataset.
* Limited Context Understanding: Paraphrasing tools may also struggle with understanding the context of the input text, especially in situations where there are multiple possible meanings for a word or phrase. This can lead to the generation of paraphrases that are not relevant to the intended meaning.
* Inability to Preserve Meaning: One of the most important goals of a paraphrasing tool is to preserve the original meaning of the input text. However, some models may not be able to achieve this goal, either because they are unable to fully understand the context or because they have learned from a biased dataset.
* Time and Resource Intensive: Building a high-quality paraphrasing tool can be a time and resource-intensive process, requiring significant amounts of training data and computational power. This can make it difficult for smaller organizations or individuals to build their own tools.

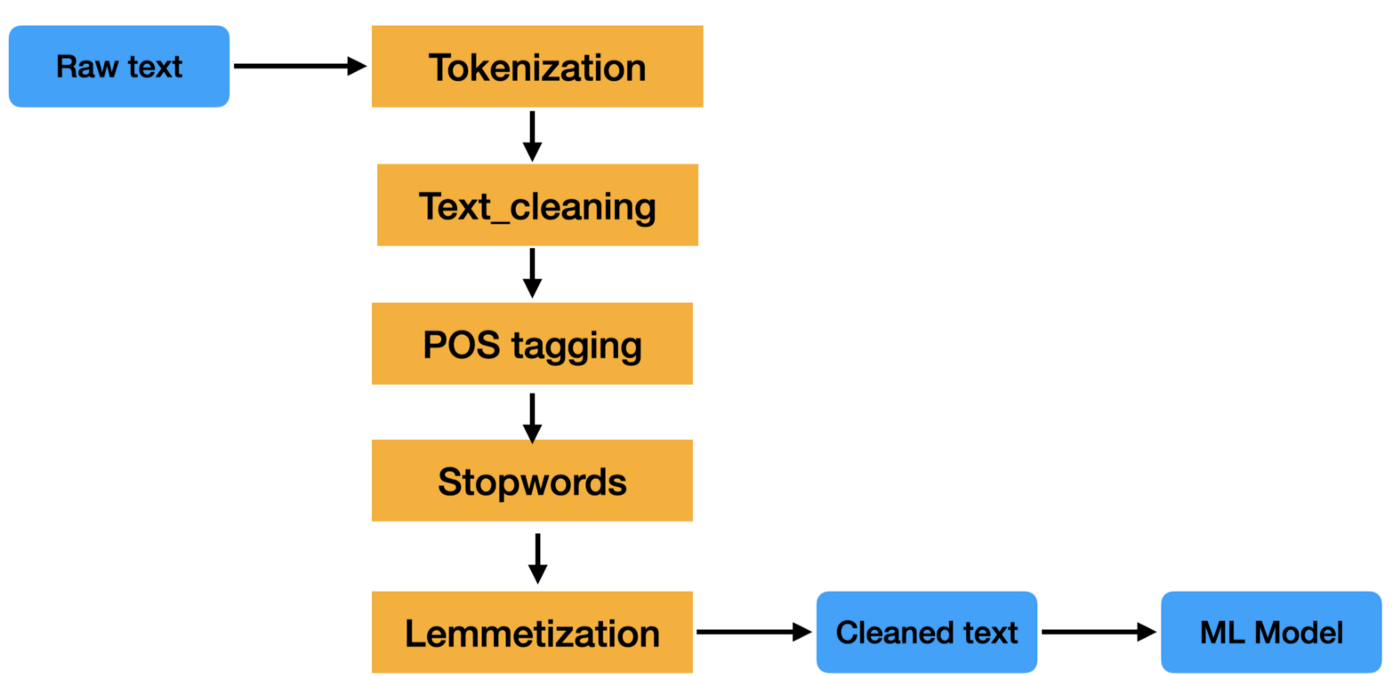
**3.2 Proposed Solution**

Our proposed solution uses the following pipeline:

1. Data collection: Collect a large corpus of text data that the paraphrasing tool can use to learn from. The data should be diverse and cover a wide range of domains and topics. The more data you have, the better your model will be at generating high-quality paraphrases. You can use web scraping, APIs, or pre-existing datasets to collect the data.
2. Pre-processing: Pre-process the text data to clean it and prepare it for use in the paraphrasing tool. This step involves tasks such as tokenization, stop word removal, stemming, and lemmatization. You may also need to perform sentence segmentation to separate the text into individual sentences.
3. Data augmentation: Augment the pre-processed text data to create additional training examples for your model. You can use techniques such as back-translation, synonym replacement, and paraphrasing to create more diverse and varied training data.
4. Training a language model: Train a language model on the pre-processed and augmented text data using techniques such as recurrent neural networks (RNNs) or transformer-based models like BERT or GPT. The language model should be trained to understand the meaning of sentences and generate coherent paraphrases. You can use frameworks like TensorFlow or PyTorch to train your model.
5. Fine-tuning: Fine-tune the language model on a smaller dataset of human-generated paraphrases to improve its paraphrasing ability. You can use transfer learning to adapt a pre-trained model to a specific task. Fine-tuning involves updating the weights of the pre-trained model using the human-generated paraphrases.
6. Integration: Integrate the paraphrasing model into a user-friendly interface such as a web app, a browser extension, or a command-line tool. The interface should allow users to input a sentence or paragraph and generate paraphrases.
7. Evaluation: Evaluate the performance of the paraphrasing tool by comparing its generated paraphrases against human-written paraphrases using metrics such as BLEU, ROUGE, or other evaluation methods. You can also use crowdsourcing to get feedback from users on the quality of the generated paraphrases.

The main model in our approach uses Text-To-Text Transfer Transformer (T5).

Transformers are a type of deep learning model that have been highly successful in natural language processing tasks, including paraphrasing. NLP’s Transformer is a new architecture that aims to solve tasks sequence-to-sequence while easily handling long-distance dependencies. Computing the input and output representations without using sequence-aligned RNNs or convolutions and it relies entirely on self-attention.



The Transformer model is based on the encoder-decoder architecture. The encoder is the gray rectangle on the left and the decoder is on the right. The encoder and decoder consist of two and three sublayers, respectively. Multi-head self-awareness, fully connected feedforward network, and encoder decoder self-awareness in the case of decoders (called multi-head attention) with the following visualizations).

**Encoder:**The encoder is responsible for stepping through the input time steps and encoding the entire sequence into a fixed-length vector called a context vector.  
**Decoder:** The decoder is responsible for stepping through the output time steps while reading from the context vector.

