TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

MAJOR PROJECT FINAL REPORT

ON

PARAPHRASE GENERATION OF NEPALI LANGUAGE IN DEVANAGARI SCRIPT USING NATURAL LANGUAGE PROCESSING

SUBMITTED BY:

AAJAY SAPKOTA (PUL075BEI001)

ABINASH ACHARYA (PUL075BEI004)

ANISH BADE (PUL075BEI008)

MAHESH UPRETI (PUL075BEI017)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

30 April 2023

Page of Approval

TRIBHUVAN UNIVERSIY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled **"PARAPHRASE GENERATION OF NEPALI LANGUAGE IN DEVANAGARI SCRIPT USING NATURAL LANGUAGE PROCESSING"** submitted by **Aajay Sapkota**, **Abinash Acharya**, **Anish Bade**, and **Mahesh Upreti** in partial fulfillment of the requirements for the Bachelor's degree in Electronics & Computer Engineering.

....	....
Supervisor	Internal examiner
Dr. Arun Kumar Timalsina	
Associate Professor	.
Department of Electronics and Computer Engineering,	Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.	Pulchowk Campus, IOE, TU.

....
External examiner

.
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

Date of approval:

Copyright

Acknowledgements

The completion of this major project has been an arduous journey. We would like to express our heartfelt gratitude to all those who have helped us throughout this project.

First and foremost, we would like to thank our project supervisor **Dr Arun Kumar Timilsina** for always supporting and believing in us since the beginning, whose invaluable guidance, extensive knowledge, and critical feedback were instrumental in shaping the direction and focus of this research work. Similarly, **Dr Umanga Bista's** unwavering support, constant encouragement, and insightful advice have not only challenged us to strive for excellence but also motivated us to overcome the obstacles and difficulties encountered during the research process.

We are ever indebted to our colleagues and friends, who have contributed to the success of this project by providing diverse perspectives, useful feedback, and constructive criticism to make our project stand where it is today.

In addition, we would like to express our gratitude to the faculty and staff of the department of electronics and computer engineering for creating a supportive and intellectually stimulating environment that nurtured our growth as a researcher and scholars.

Ultimately, we would like to acknowledge the invaluable contributions of all those who have supported us throughout this research journey. Without their assistance and encouragement, this project would not have been possible.

Abstract

The project aims to develop a system for generating paraphrases using transformer-based models. Fine-tuning the pre-trained models on a large-scale dataset of sentence pairs, consisting of source sentences and their corresponding paraphrases, and evaluation of their performance on several benchmarks was performed. To accomplish the project's objectives, several tasks were undertaken, such as researching and allocating resources, collecting and translating datasets, sampling, filtering, and analyzing the feasibility of the model. The comprehensive approach employed in the project has enabled the development of a powerful tool for generating high-quality paraphrases, which could enhance the natural language processing and generation capabilities of various applications. Moreover, this model excels in utilizing mathematical and statistical metrics such as BLEU and ROUGE scores to accurately assess paraphrasing. Additionally, the model demonstrated excellent performance on different datasets, showcasing its ability to generalize across different types of test sets. But, the zero-shot evaluation produced a result not so expected, suggesting a low recall score for new sentences which highlighted the need for further improvements in the model. Similarly, this model faces significant challenges such as entity mismatches, semantic and syntactic differences, and exact match problems between the input sentences and their corresponding generated sentences. Furthermore, the implementation of a web application enabled users to input sentences and receive their paraphrases in real time, demonstrating the practicality of our approach. Nonetheless, this research emphasizes the vast potential of advanced language models to enhance natural language processing capabilities in low-resource languages.

Keywords: *Natural Language Processing, Transformer, ROUGE, BLEU.*

Contents

List of Figures

List of Tables

List of Abbreviations

AdamW	Adaptive Moment Estimation Weight Decay
BART	Bidirectional and Auto-Regressive Transformers
Bi-LSTM	Bidirectional Long-Short Term Memory
BLEU	Bilingual Evaluation Understudy
BP	Brevity Penalty
BPE	Byte-Pair Encoding
CSV	Comma Separated Values
HTML	HyperText Markup Language
LSTM	Long Short-Term Memory
MVC	Model-View-Controller
NLTK	Natural Language Toolkit
NLP	Natural Language Processing
ORM	Object-Relational Mapping
POS	Part Of Speech
REPL	Read-Evaluate-Print-Loop
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
RNN	Recurrent Neural Network
T5	Text-to-Text Transfer Transformer
TF-IDF	Term Frequency-Inverse Document Frequency
URL	Uniform Resource Locator
VSCode	Visual Studio Code

1. Introduction

1.1 Background

Paraphrasing is a critical task in natural language processing (NLP) that involves the rewording of a sentence while retaining its intended meaning. This task is particularly important for tasks such as text summarization, machine translation, and data augmentation. With the advent of transformer-based language models, such as BERT[1], BART[2], GPT[3], and T5[4], there has been a significant improvement in the quality of paraphrasing, particularly in the English language. However, paraphrasing in other languages, such as Nepali, remains a challenging task due to the lack of large-scale annotated data and limited research in this area.

In this report, we present our efforts to fine-tune a transformer-based model, specifically the mT5 model, for paraphrasing in the Nepali language. Moreover, we evaluate the quality of the generated paraphrases using both automatic and human evaluation methods and discuss the potential applications of our model in Nepali language processing tasks. The transformer architecture is based on the idea of self-attention, which allows the model to focus on different parts of the input sequence at different timesteps in contrast to previous NLP models, which typically used recurrent or convolutional layers to process input sequences. One of the key advantages of the transformer architecture for paraphrasing is its ability to handle long-range dependencies and capture global context which is often required for a deep understanding of the underlying meaning and context of the original sentence.

In general, this project has been broadly divided into dataset, model, and analysis. Here, the dataset consists of a combination of manually paraphrased data as a gold label and machine-translated and processed datasets like Quora[5] and paraNMT[6] as a bronze label. Further, various levels of sampling and filtering techniques were implemented for the removal of the noise and irrelevant data to optimize the performance of the model on this dataset.

In our project, we undertook extensive fine-tuning and experimentation with the mT5-small and mT5-base models [7] to improve the generation of paraphrases. Furthermore, we employed a back-translation method to conduct a comparative analysis. This approach allowed us to explore various techniques and optimize the performance of our paraphrasing models. Once the fine-tuning task on the pre-trained models was completed, a combination of top-k sampling and top-p (nucleus) sampling techniques was used during the inference to generate the paraphrases.

Regarding the quantitative analysis of our paraphrase models, the implementation of various metrics like BLEU[8] and ROUGE[9] score enabled the possibility to compare the model-generated paraphrase sentences with the dataset contained paraphrases for each input sentence. Moreover, the BLEU score incorporates BLEU-2 and BLEU-4 while the ROUGE score is reflected through ROUGE-1 and ROUGE-2.

In summary, our major goal is to develop a paraphrasing tool using a transformer model which as a black-box model provides a paraphrase of the text(Devanagari script written in the Nepali language) fed into the model.

1.2 Motivation

Paraphrasing is such a tedious job: a large-scale manual restructuring of sentences that best fit the usage scenario is quite hard to execute. With millions of people having access to similar types of research articles, blogs, papers, and generated text, issues like copyright and plagiarism are inevitable. In the case of Nepali being a low-resource language, there is a scarcity of annotated data and language models. This limits the development of high-quality paraphrasing models for Nepali.

Moreover, our main goal is to contribute to the language community by providing valuable resources that can improve language generation and understanding. These resources can be applied in various ways, such as paraphrase generation, plagiarism detection, summarization, and enhancing content quality.

1.3 Problem statement

Despite the growing importance of natural language processing (NLP) techniques in various fields, the Nepali language lacks automated tools for paraphrasing. Unlike other languages such as English and French, there are limited resources available in Nepali for services such as summarization, sentiment analysis, and paraphrasing.

Developing a proper library and dataset in itself is a time-consuming and challenging process, and the lack of open-source contributors in our market makes it even harder. While there have been some projects in Nepali natural language processing, such as sentiment analysis, there has been limited research and implementation on paraphrase generation or more over in text generation.

Even while searching for resources related to Devanagari text, a significant lack of libraries, datasets, and tools to support NLP was felt. This highlights the need for further development and exploration of natural language processing techniques in the Nepali language.

1.4 Objectives

The major objectives of this project are as follows:

- To create a dependable dataset of the Nepali language.
- To make a sophisticated system for paraphrasing Nepali text in Devnagari Script.

1.5 Scope

- It can be used for the production and deployment of accurate, reliable, and user-friendly Nepali paraphrasing tools.
- Paraphrasing models can be used to simplify complex technical jargon or academic language for a non-expert audience without losing its originality.
- Dealing with the plagiarism issue while copying texts or rewriting similar texts by presenting information from a source in a unique way while still giving credit to the original author.
- Summarizing or simplifying for academic and professional purposes such as writing business reports, marketing materials, journalism, etc.
- It can be used in multiple areas such as speech writing, presentation, and personal communication, by making messages clear, concise, or engaging for better conveying or varying messages in accordance with what resonates with the audience.

In short, the scope of paraphrasing is quite vast and it can be applied in various contexts for better-conveying information and enhancing communication.

2. Literature Review

2.1 Related Work

The first-ever attempt in this field was made in 1983 [10].The paraphrasing approach used was a syntactic one with a rule-based approach.In 2009[11], another rule-based approach was proposed,which focused on automatically learning complex paraphrase patterns. Thesaurus-based methods were also explored by Bolshakov and Gelbukh [12], and Kauchak and Barzilay[13]. The first-ever natural language generation-based approach dates back to 2003[14]. In 2004, Quirk, Brockett and Dolan proposed a statistical machine learning theory based on generating sentence-level paraphrases [15].

More recently ,many neural network based techniques have been developed including a stacked LSTM network for paraphrasegeneration[16] and deep generative framework as variational autoencoder-based architecture augmented with sequence to sequence model[17]. The latest findings depict Paraphrase generation and identification using the T5 base model[18].In Nepali, researchers have developed a transformer encoder block inspired by the BERT architecture, which they have named NepBerta[19]. Although sentiment analysis using neural networks has been explored to some extent[20] [21], there has been a lack of progress in the neural network approach to text generation. Despite the numerous advancements in the field, little work has been carried out in the Nepali language, making it the first for the paraphrase generation in the Nepali language.

2.2 Related Theory

2.2.1 Dataset Processing

A dataset in natural language processing (NLP) is a group of written or spoken language examples that have been gathered and organized for the purpose of training and testing different models. This dataset usually includes many documents or texts that can be broken down into smaller units like words or sentences. These datasets are used to train and evaluate machine learning models. To create a dataset for paraphrasing in the Devanagari script, several approaches can be taken. One approach is to collect existing parallel corpora, which consist of pairs of sentences in the source language and their translations in the target language. These parallel corpora can be used to train machine learning models for different NLP tasks such as paraphrase generation, summarization and so on. Another approach here is to use rule-based methods for paraphrase generation where linguistic rules and patterns are used to generate alternative sentences with similar meanings to the original text. For example, synonyms can be substituted for words in the original sentence, or the sentence structure can be modified to create a paraphrase.

2.2.2 Pytorch lightning

PyTorch Lightning is a lightweight wrapper library for PyTorch. It provides a high-level interface for organizing PyTorch code into reusable and modular components. Its primary aim is to simplify training deep learning models by providing a standard structure for writing PyTorch code that is easy to read, debug, and scale. PyTorch Lightning offers a set of abstractions for everyday deep-learning tasks such as training loops, validation, and testing. It also handles low-level details such as distributed training, checkpointing, and logging. PyTorch Lightning is designed to make it easier for researchers and practitioners to focus on building and experimenting with deep learning models rather than worrying about the details of implementing and managing the training process.

2.2.3 Transformer

A transformer is a type of deep learning model that was introduced in the paper "Attention Is All You Need" by Vaswani et al. in 2017. It is an architecture that primarily uses self-attention mechanisms to process sequential data, such as natural language text.

Traditionally, RNNs were used for sequential data processing, but they suffer from issues like vanishing gradients and difficulty in parallelization. Transformers overcome these limitations by using a self-attention mechanism that allows the model to attend to different parts of the input sequence while processing each token.

The core idea behind self-attention is to compute a weighted sum of the input sequence tokens based on their importance with respect to a specific token. These importance weights are learned during the training process and are used to compute a context vector for each token. The context vectors are then fed through a feedforward neural network to obtain the final output.

Transformers have been used in various natural language processing tasks such as language modeling, machine translation, and text classification. They have also been applied in other domains, such as computer vision and speech recognition.

2.2.4 Language Models

A language model is an AI system that can comprehend and generate human language. It is trained on large datasets of text and employs statistical and machine learning methods to predict the most likely words or phrases to follow in a given sentence or text.

Language models have diverse applications such as speech recognition, natural language processing, machine translation, and text generation. They enhance the capabilities of search engines, chatbots, and virtual assistants to comprehend and respond to user inquiries.

Various types of language models exist, including n-gram models, recurrent neural networks (RNNs), and transformers, each with its unique strengths and weaknesses. These models are deployed in specific contexts based on the application and the requirements at hand.

Here are some examples of language models:

- GPT-3 (Generative Pre-trained Transformer 3) developed by OpenAI.
- BERT (Bidirectional Encoder Representations from Transformers) developed by Google.
- ELMo (Embeddings from Language Models) developed by the Allen Institute for Artificial Intelligence.
- ULMFiT (Universal Language Model Fine-tuning) developed by Jeremy Howard and Sebastian Ruder.
- XLNet (eXtreme MultiLingual Language Model) developed by Carnegie Mellon University and Google Brain.
- RoBERTa (Robustly Optimized BERT Pre-training Approach) developed by Facebook AI Research.
- ALBERT (A Lite BERT) developed by Google Research and Toyota Technological Institute in Chicago.
- T5 (Text-to-Text Transfer Transformer) developed by Google.
- MT5 (Multilingual Text-to-Text Transfer Transformer) developed by Google.

These models are among the most advanced and widely used language models today, and they have contributed significantly to the field of natural language processing. Among these, T5 and MT5 are the models of our concern.

2.2.4.1 T5

T5 is a transformer-based language model created by Google AI Language. It is a highly flexible model that can perform a range of natural language processing tasks, including text classification, question answering, language translation, and summarization.

T5's ”text-to-text” approach is one of its most notable features. This means that the input and output text is transformed into a standard format before processing, allowing the model to learn a diverse range of text-to-text transformations.

The T5 model utilizes a transformer architecture, which is a neural network design ideal for sequential data such as text. It is comprised of several layers of self-attention and feedforward neural networks, enabling it to capture intricate patterns and dependencies within the input data.

To pre-train T5, a technique called ”masked language modeling” is utilized. This involves randomly masking words in the input text and training the model to predict the masked words based on the context.

Once pre-trained, T5 can be fine-tuned for specific natural language processing tasks such as machine translation or text classification. During fine-tuning, the model is trained on a smaller dataset that is specific to the target task, enabling it to learn task-specific patterns and improve its performance on that task.

T5 is a high-performing language model that has achieved state-of-the-art results on a range of natural language processing benchmarks. As a result, it is widely used across many natural language processing applications.

2.2.4.2 mT5

mT5 (Multilingual Translation Transformer) is a natural language processing (NLP) model developed by Google. It is based on the Transformer architecture, which is a deep learning model designed to process sequential data, such as text. MT5 is specifically designed for machine translation and can translate between 100 different languages.

mT5 is a multilingual model, meaning it can translate between any of the 100 supported languages. This makes it a powerful tool for communication and information exchange across linguistic boundaries. Additionally, MT5 can be fine-tuned for specific domains or languages, allowing for even more accurate translations.

One of the key features of mT5 is its ability to handle complex sentence structures and idiomatic expressions. This is achieved through the use of contextual embeddings, which capture the meaning of words based on their context in the sentence.

mT5 also utilizes a pre-training and fine-tuning approach to improve its translation accuracy. The model is first trained on a large corpus of text in multiple languages, allowing it to learn the nuances of each language. It is then fine-tuned on specific translation tasks, further improving its accuracy.

Overall, mT5 is a powerful tool for machine translation that can handle a wide variety of languages and complex sentence structures. Its accuracy and flexibility make it a valuable tool for communication, research, and business in multilingual contexts.

2.2.4.3 mBART

The mBART model is a multilingual extension of the BART model developed by Facebook AI. It is a language model capable of performing various natural language processing tasks, such as machine translation and text generation, in multiple languages simultaneously.

The mBART model is trained on a large corpus of text data in multiple languages using a combination of techniques, including masked language modeling and denoising autoencoding. It uses a transformer-based architecture, which is a neural network architecture designed to handle sequential data such as text.

One of the key advantages of the mBART model is its ability to handle multiple languages simultaneously. This makes it particularly useful for multilingual applications, such as cross-lingual transfer learning and machine translation.

The mBART model has achieved state-of-the-art results on various machine translation benchmarks, outperforming previous multilingual models. It is also available in the Hugging Face Transformers library, making it accessible to developers and researchers for a wide range of natural language processing tasks.

2.2.4.4 Back Translation

Back-translation is a technique used in machine translation to improve the quality of translations by translating a text from the target language back into the source language. This technique involves training a neural machine translation model on a parallel corpus (a set of texts in the source and target languages) and then using the model to translate a text from the target language back into the source language. The resulting text is then compared to the original source text to identify errors and improve the quality of the translation. Back-translation is a simple and effective way to generate additional training data for machine translation models and has been shown to improve the quality of translations in several studies. It is commonly used in both academic research and commercial machine translation systems.

2.2.5 Tokenizer

A tokenizer is a program or a library that breaks text into tokens which are in fact the smaller pieces of the text. Different types of tokenizers are available. Word tokenizers and character tokenizers are the most common ones. Some of the widely used tokenizers by NLP researchers are TreebankWorkTokenizer, TweetTokenizer, MWETokenizer, T5Tokenizer, mT5Tokenizer, MBartTokenizer, AutoTokenizer, etc.

2.2.5.1 T5 Tokenizer

The T5 tokenizer is a type of tokenizer that is specially designed for use with the T5 transformer model. It uses subword tokenization to split words into smaller subword units, which helps the model to process rare and out-of-vocabulary words more efficiently.

One of the unique features of the T5 tokenizer is that it is a text-to-text tokenizer. This means that it converts both input and output text into a standardized text format prior to processing. This approach enables the T5 model to perform a wide range of text-to-text transformations, which allows it to generalize better to new domains and tasks.

The T5 tokenizer can handle various input and output formats, including sequence-to-sequence, text classification, and language modelling tasks. It also includes specialized tokens that help the model to perform specific tasks, such as padding, masking, and delimiting the input and output sequences.

Overall, the T5 tokenizer is an essential component of the T5 model's high accuracy and efficiency in performing various natural language processing tasks.

Here's an example of how the T5 tokenizer works:

First, we initialize the T5 tokenizer:

Let's say we want to tokenize the following sentence:
"I love using T5 for natural language processing."

First, we initialize the T5 tokenizer:

```
from transformers import T5Tokenizer
tokenizer = T5Tokenizer.from_pretrained(’t5-small’)
```

Then, we use the tokenizer.encode method to tokenize the sentence:

```
text = ”I love using T5 for natural language processing.”
tokens = tokenizer.encode(text)
```

The output of tokens will be:

```
[216, 3, 129, 1218, 93, 10751, 15, 847, 125, 3094, 531, 424, 1]
```

This is a list of token IDs, where each ID corresponds to a particular token in the T5 vocabulary. We can use the tokenizer. decode method to convert these token IDs back into text:

```
decoded_text = tokenizer.decode(tokens)
```

The output of decoded_text will be:
’I love using T5 for natural language processing.</s>’ Here, the </s> token is added by the T5 tokenizer to indicate the end of the sequence.

2.2.5.2 mT5 Tokenizer

mT5Tokenizer is a tokenizer designed specifically for the multilingual transformer model called ”mT5” (multilingual T5), which is part of the family of T5 models developed by Google. mT5 Tokenizer is used to preprocess text data before feeding it into the mT5 model for tasks such as language translation, summarization, and question answering.

mT5Tokenizer is based on the Byte-Pair Encoding (BPE) algorithm, which is a data compression technique that can also be used for tokenization. However, mT5Tokenizer uses a variation of BPE called SentencePiece, which is designed specifically for multilingual models. SentencePiece is able to handle multiple languages by generating a shared vocabulary of subword units that can be used across different languages.

mT5Tokenizer is implemented in Python using the Hugging Face Transformers library, which provides a unified API for working with various transformer models. This tokenizer can be used for both training and inference, and it can handle text data in multiple languages. Overall, mT5 Tokenizer is a useful tool for researchers and developers working with multilingual natural language processing tasks using the mT5 model.

Here’s an example of how the mT5 tokenizer works with the input sentence:

”उद्धारकर्मीहरूले जीवित मानिसहरूको खोजी गरिरहेका छन्।”

First, we initialize the mT5 tokenizer:

```
from transformers import MT5Tokenizer

tokenizer = MT5Tokenizer.from_pretrained(’google/mt5-small’)
```

Then, we use the tokenizer. encode method to tokenize the sentence:

```
text = ”उद्धारकर्मीहरूले जीवित मानिसहरूको खोजी गरिरहेका छन्।”

tokens = tokenizer.encode(text)
```

The output of tokens will be:

```
[22169, 19, 4250, 145, 6, 1336, 5957, 2686, 1747, 26, 186, 97, 309, 1]
```

This is a list of token IDs, where each ID corresponds to a particular token in the mT5 vocabulary. We can use the tokenizer.decode method to convert these token IDs back into text:

```
decoded_text = tokenizer.decode(tokens)
```

The output of decoded_text will be:

’उद्धारकर्मीहरूले जीवित मानिसहरूको खोजी गरिरहेका छन्।

.</s>’ Here, the </s> token is added by the mT5 tokenizer to indicate the end of the sequence.

2.2.5.3 mBART Tokenizer

MBartTokenizer is a special type of tokenizer developed by Facebook AI. It is a multilingual tokenizer specifically designed to work with the mBART model, which is a multilingual extension of the BART (Bidirectional and Auto-Regressive Transformers) model.

The mBART Tokenizer is designed to process text data in multiple languages and can handle both monolingual and multilingual text. It is a subword tokenizer that uses byte pair encoding (BPE) to split words into smaller subword units, allowing it to handle out-of-vocabulary and rare words more efficiently.

The mBART Tokenizer supports various input and output formats, including sequence-to-sequence, text classification, and language modelling tasks. It also includes specialized tokens for tasks such as padding, masking, and delimiting the input and output sequences.

One of the key advantages of the mBART Tokenizer is its ability to handle multiple languages simultaneously. This makes it particularly useful for multilingual applications, such as cross-lingual transfer learning and machine translation.

In summary, the mBART Tokenizer is a powerful multilingual tokenizer designed to work with the mBART model, and its ability to process text data in multiple languages makes it a valuable tool for various natural language processing tasks.

Here's an example of how the mBART tokenizer works with the given input sentence in Nepali:

First, we initialize the mBART tokenizer:

```
from transformers import MBartTokenizer

tokenizer = MBartTokenizer.from_pretrained('facebook/mbart-large-cc25')
```

Then, we use the tokenizer. encode method to tokenize the sentence:

```
text = ”प्रभावितहरूका लागि एक प्रमुख अन्तर्राष्ट्रिय राहत प्रयासले गति लिएको छ।”

tokens = tokenizer.encode(text)
```

The output of tokens will be:

```
[100, 11618, 1548, 36317, 55, 1447, 1383, 307, 84, 1681, 1132, 37951, 32987, 98]
```

This is a list of token IDs, where each ID corresponds to a particular token in the mBART vocabulary. We can use the tokenizer.decode method to convert these token IDs back into text:

decoded_text = tokenizer.decode(tokens)

The output of decoded_text will be:

'»ne«"प्रभावितहरूका लागि एक प्रमुख अन्तर्राष्ट्रिय राहत प्रयासले गति लिएको छ।"

Here, the »ne« token is added by the mBART tokenizer to indicate that the language of the input sentence is Nepali.

2.2.6 Inference

In machine learning and artificial intelligence, the inference is the process of using a trained model to predict or classify new, previously unseen data. During inference, the model takes in input data and produces an output based on the patterns and relationships it has learned from the training data. This output can be a predicted value, a label, or a sequence of values.

Inference is a crucial step in the machine learning workflow, as it allows models to be applied to real-world problems. For example, a model trained to recognize objects in images can be used to identify objects in new images, and a model trained to predict customer churn can be used to identify which customers are likely to leave a business.

Inference can be performed in real-time, such as for live video or audio processing, or offline, such as for batch processing of large datasets. Regardless of the use case, the objective of inference is to use the trained model to generate accurate predictions on new data.

To summarize, inference is the process of using a trained model to make predictions on new data and is an essential component of machine learning applications.

2.2.6.1 Top-k sampling

Top-k sampling is a popular text generation method used in natural language processing, particularly in language modeling tasks. It is a variation of the more conventional approach to text generation, known as beam search.

In top-k sampling, the model selects the k most likely next tokens from the probability distribution that the model generates for the current input. These k tokens are then utilized to construct the probability distribution for the next token, from which the model samples generate the next token in the sequence. This method enables more diverse and captivating text generation, unlike beam search, which typically produces more conservative and repetitive outputs.

The selection of k can vary depending on the desired level of diversity in the generated text. Using a higher value of k results in a more diverse output but may also increase the likelihood of nonsensical or ungrammatical text. Conversely, using a smaller value of k produces more conservative output but may also lead to repetitive or predictable text.

Overall, top-k sampling is an effective technique for generating diverse and engaging text output in natural language processing tasks.

The mathematical formula for top-k sampling is as follows:

- Compute the probability distribution over the vocabulary for the next word given the input and previous words. Let P be the probability distribution.

- Sort the probability distribution in descending order.

- Let S be the set of indices of the top k words with the highest probabilities. Let s(i) be the i-th index in the set S.

- Normalize the probability distribution over the set of words in S, so that the probabilities sum up to 1. Let Q be the normalized probability distribution.

  $$Q(s_i) = \frac{P(s_i)}{\sum_{j=1}^{k} P(s_j)}$$

- Sample the next word from the normalized distribution Q.

  The parameter k controls the size of the reduced set of words to sample from. A larger k value increases the diversity of the generated text, while a smaller k value results in more conservative and repetitive output.

  In summary, the formula for top-k sampling is:

  $$Q(s_i) = \frac{P(s_i)}{\sum_{j=1}^{k} P(s_j)}$$

  where P is the probability distribution over the vocabulary, S is the set of indices of the top k words with the highest probabilities, s_i is the i^{th} index in the set S, and Q is the normalized probability distribution over the set of words in S.

2.2.6.2 Top-p (nucleus) sampling

Top-p sampling is a text generation technique used in natural language processing, similar to top-k sampling. This method is also known as nucleus sampling, where the model selects from the most probable next tokens whose cumulative probability exceeds a certain threshold value, p.

In contrast to top-k sampling, which selects the top-k tokens with the highest probability, top-p sampling considers the cumulative probability of the tokens, where the least probable tokens are discarded until the cumulative probability reaches the threshold p.

The selection of p determines the number of possible tokens that the model will consider for generating the next token in the sequence. A smaller value of p leads to more conservative and predictable outputs, while a larger value of p results in more diverse and unpredictable outputs.

Top-p sampling is a useful method in scenarios where it's important to maintain a balance between diversity and coherence in generated text, such as dialogue generation or summarization tasks. By limiting the selection of tokens based on cumulative probability, top-p sampling allows for more control over the quality and diversity of generated text compared to traditional methods like greedy search. The formula for top-p sampling is as follows:

- Compute the probability distribution over the vocabulary for the next word given the input and previous words.
- Sort the probability distribution in descending order.
- Compute the cumulative probabilities of the sorted words.
- Select the smallest possible set of words whose cumulative probability exceeds p. Let this set be denoted as S.
- Normalize the probability distribution over the set of words S so that the probabilities sum up to 1.
- Sample the next word from the normalized distribution.

The parameter p controls the size of the set of words to sample from. A larger p-value increases the diversity of the generated text, while a smaller p-value results in more conservative and repetitive output.

Here is the mathematical formula for computing the set S:

Let P be the sorted probability distribution, with p_i being the probability of the i^{th} word.

Let F(i) be the cumulative distribution function defined as:

$$F(i) = \sum_{j=1}^{i} p(j)$$

Then, the set S is defined as:

$$S = i : F(i) > p$$

where i denotes the index of the i^{th} word in the sorted probability distribution P.

2.2.7 AdamW OPtimizer

AdamW is an optimizer that builds upon the popular Adam optimizer. It was proposed by Loshchilov and Hutter in their 2017 paper ”Fixing Weight Decay Regularization in Adam”. The main difference between Adam and AdamW is the way they handle weight decay regularization.

In Adam, weight decay is added to the gradient update during each iteration, which is equivalent to L2 regularization. However, this can lead to suboptimal solutions, especially in deep learning models. On the other hand, AdamW separates weight decay from the gradient update, which can lead to better performance.

The AdamW optimizer adds weight decay to the loss function after each iteration instead of directly modifying the gradient update. This allows the optimizer to update the weight decay more effectively and prevent overfitting.

The formula for updating the parameters in AdamW is:

$$\theta_t = \theta_{t-1} - \alpha \left(\frac{m_t}{\sqrt{v_t} + \epsilon} + \lambda \times \theta_{t-1} \right)$$

where:
θ_{t-1} is the parameter in the previous time step
α is the learning rate
m_t and v_t are the first and second moments of the gradients, respectively, computed using exponentially decaying averages
ϵ is a small value to prevent division by zero

λ is the weight decay coefficient
The main benefit of using AdamW is the improved generalization performance, especially in deep neural networks with many layers. Additionally, AdamW has been shown to be more robust to changes in hyperparameters and training conditions compared to other optimizers.

Overall, AdamW is a useful optimizer to consider for deep learning models, especially when regularization is a concern.

2.2.8 Cross Entropy Loss

Cross entropy loss refers to the difference between two random variables. It is measured in order to evaluate the difference in the information that they contain. This loss function is used to calculate the accuracy of the machine learning or deep learning model by defining the difference between the desired outcome and the estimated probability.

Cross-entropy loss is a commonly used loss function in classification tasks. It measures the difference between the predicted probabilities and the actual probabilities of the classes. This loss function is popular in classification tasks because it penalizes the model more heavily for predictions that are farther from the true probabilities and less heavily for predictions that are closer to the true probabilities, making it a good choice for achieving high accuracy.

In contrast, regression tasks involve predicting continuous numerical values such as the price of a house or the age of a person. Mean squared error (MSE) loss is commonly used in regression tasks to measure the difference between the predicted value and the actual value.

In classification tasks, the output is usually a set of probabilities assigned to each category. The predicted class is then determined by selecting the class with the highest probability. In contrast, the output of regression tasks is a continuous numerical value.

The formula to calculate cross-entropy loss is

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right)$$

where:
L is the cross entropy loss
N is the number of samples in the dataset
y is the actual target class (0 or 1)
p is the predicted probability of the target classs

2.2.9 Visualization tools

2.2.9.1 Pandas

Pandas is a Python library that is widely used for data manipulation, analysis, and visualization. It provides easy-to-use data structures and data analysis tools for working with structured data, such as tabular data, time series, and other types of data. Pandas is built on top of the NumPy library and can efficiently handle large datasets with millions of rows and columns. Some of the key features of pandas include data cleaning, data transformation, merging and joining of datasets, grouping and aggregating data, and data visualization. Pandas is a powerful tool for data scientists, analysts, and researchers who work with data in Python.

2.2.9.2 CSV logger

As the name suggests itself, a CSV logger is a tool that logs data in Comma Separated Values(CSV) format. CSV is a file format used to store tabular data, for instance: spreadsheets, database data, etc. A CSV logger takes the data from some input sources such as an ML model in our case and writes this data to a CSV file. Each row in such a file represents a single data point, where each value is separated by a comma. CSV loggers are most commonly used in scientific and engineering applications to track data over the time domain. The resulting CSV files can be easily used for data analysis through software like Excel or python.

2.2.10 Score analysis

2.2.10.1 NLTK

NLTK is a very popular python library used for working with human language data. It contains libraries and programs for statistical language processing tasks. It is a powerful NLPlibrary, which contains packages to make machines understand human language and reply to it with an appropriate response. It is widely used in the education and research sectors. It is widely used for exploring and analyzing text data. It also provides various language resources for different languages. Such resources include text collections, lexicons, and grammar for those languages. It also contains the libraries to calculate the bleu score and glue score which are useful for the evaluation of NLP tasks which will be further discussed in later stages.

2.2.10.2 BLEU score

BLEU score is a metric mostly used to measure the quality of machine-generated text by comparing it to one or more reference texts. It works by comparing the n-gram overlap between the machine-generated text and the reference text, where n refers to the number of consecutive words in a sequence. The higher the BLEU score, the closer the machine-generated text is to the reference text. BLEU scores range from 0 to 1, with 1 indicating a perfect match. While BLEU scores can provide a useful measure of machine generation quality, they are not always an accurate reflection of how well a text generation model performs for a given task or in a specific domain and should be used in conjunction with other evaluation methods.

In our project, we have used the BLEU score metrics as one of the evaluation metrics to compare the n-gram overlap between the labeled paraphrase with the generated paraphrase.

The formula for computing the BLEU score is as follows:

$$BLEU = \text{BP} \times \exp\left(\frac{1}{n}\sum_{i=1}^{n} w_i \log(p_i)\right)$$

where:
BP is a penalty term to account for the fact that shorter texts are favored over longer ones. It is defined as:

$$BP = 1, \text{if MT\_output\_length} \geq \text{reference\_length}$$
$$BP = \exp(1 - (\frac{reference\_length}{MT\_output\_length}))$$

otherwise, n is the length of the n-gram used for comparison (typically n = 4)
w_i is the weight assigned to the i^{th} n-gram
p_i is the precision of the i^{th} n-gram, which is defined as the number of times the i^{th} n-gram appears in the model output that also appears in any of the reference text, divided by the total number of i^{th} n-grams in the model output. The weights w_i are usually set to 1/n so that all n-grams are weighted equally. However, other weighting schemes can also be used.

BLEU-2 score BLEU-2 (also known as BLEU-2gram) measures the precision of 2-grams (pairs of adjacent words) in the machine-generated output compared to the reference text.

BLEU-4 score BLEU-4 (also known as BLEU-4gram) is similar to BLEU-2, but measures the precision of 4-grams (sequences of four adjacent words) in the machine-generated text output compared to the reference text.

In simple words, BLEU-2 uses a sequence of 2 words while Bleu-4 uses a sequence of 4 words. Generally, using longer n-grams in the calculation of the BLEU score tends to give a more accurate measure of the quality of the machine translation. However, using longer n-grams also requires more data and computational resources, and may not be feasible in all situations.

2.2.10.3 ROUGE score

ROUGE score is a commonly used metric for evaluating the quality of text summarization systems. It measures the similarity between a generated summary and a set of reference summaries by computing the overlap of n-grams (contiguous sequences of words) between them. The n-grams can be of different lengths, and typically ROUGE scores are reported for different values of n (e.g., ROUGE-1 for unigrams, ROUGE-2 for bigrams, etc.). Higher ROUGE scores indicate better summary quality, as the generated summary is more similar to the reference summaries. ROUGE scores are widely used in research on text summarization and have also been adopted as an evaluation metric in various competitions and challenges. Such characteristics of Rouge score make it eligible to be used as one of the evaluation metrics for our paraphrase generation model. ROUGE-1 and ROUGE-2 are two variants of the ROUGE metric.

Here are the mathematical formulas for calculating the ROUGE scores:

ROUGE-N:

Let R be the set of n-grams in the reference summary, and G be the set of n-grams in the generated summary. Then, the ROUGE-N score is computed as:

$$ROUGE-N = \frac{number\_of\_overlapping\_n-grams\_between\_R\_and\_G}{total\_number\_of\_n-grams\_in\_R}$$

ROUGE-L:

Let LCS be the length of the longest common subsequence between the reference and generated summaries, and let R be the number of words in the reference summary. Then, the ROUGE-L score is computed as:

$$ROUGE-L = \frac{LCS}{R}$$

ROUGE-W:

Let WLCS be the weighted length of the longest common subsequence between the reference and generated summaries, and let RW be the total weight of words in the reference summary. Then, the ROUGE-W score is computed as:

$$ROUGE-W = \frac{WLCS}{RW}$$

The weight of each word in the reference summary is typically set to 1 for consecutive matches and 0.5 for non-consecutive matches. The weight of each word in the generated summary is set to 1.

ROUGE-1 and ROUGE-2 are specific variants of the ROUGE-N score, where N is equal to 1 and 2, respectively.

ROUGE-1 score
ROUGE-1 measures the overlap of unigrams (single words) between the machine-generated paraphrase and the reference paraphrase.

Let R be the set of unigrams (single words) in the reference summary, and G be the set of unigrams in the generated summary. Then, the ROUGE-1 score is computed as:

$$ROUGE-1 = \frac{number\_of\_overlapping\_unigrams\_between\_R\_and\_G}{total\_number\_of\_unigrams\_in\_R}$$

ROUGE-2 score
ROUGE-2 measures the overlap of bigrams (pairs of adjacent words) between the two paraphrases.

Let R be the set of bigrams (pairs of consecutive words) in the reference summary, and G be the set of bigrams in the generated summary. Then, the ROUGE-2 score is computed as:

$$ROUGE-2 = \frac{number\_of\_overlapping\_bigrams\_between\_R\_and\_G}{total\_number\_of\_bigrams\_in\_R}$$

In both formulas, the number of overlapping n-grams refers to the count of n-grams that appear in both the reference and generated summaries. The total number of n-grams in the reference summary refers to the total count of n-grams in the reference summary, including repeated n-grams.

2.2.11 Developent tools

2.2.11.1 HTML

The preferred markup language for documents intended to be viewed in a web browser is HTML or HyperText Markup Language. It frequently benefits from tools like CSS and programming languages like JavaScript.Instead of displaying the HTML tags, browsers use them to interpret the content of the page.

2.2.11.2 CSS

A style sheet language called CSS is used to describe how a document is produced in a markup language like HTML or XML. The purpose of CSS is to make it possible to separate content from presentation, including layout, colors, and fonts.

2.2.11.3 Django

Django is a high-level Python web framework that follows the MVC architectural pattern. It provides a robust set of tools and features for building web applications quickly and easily. Some of the key features of Django include an ORM for database management, a built-in admin interface for managing web application content, a templating engine for creating dynamic web pages, and a URL routing system for mapping URLs to views. Django also has a strong emphasis on security, including protection against common web attacks. Django is a popular choice for web developers due to its ease of use, scalability, and strong community support.

2.2.11.4 VSCode

A source-code editor called Visual Studio Code (VSCodeworks with many different programming languages, such as C++, Fortran, Go, Java, JavaScript, Node.js, Python, etc. Its fundamental support consists of configurable snippets, code folding, bracket matching, and syntax highlighting.

2.2.11.5 Jupyter Notebook

Jupyter Notebook is an interactive computational environment used for creating notebook documents. The application runs on the browser and presents a REPL (Read-Evaluate-Print Loop) that contains an ordered list of input/output cells. These cells can contain code, text, math formulas, plots, and other media. Jupyter Notebook can colloquially refer to two different concepts, either the user-facing application to edit code and text, or the underlying file format which is interoperable across many implementations.

2.2.11.6 Kaggle

Kaggle is an online platform that hosts a community of data scientists and machine learning experts. It is a subsidiary of Google LLC that offers a range of tools and resources for working with data, including access to published data sets, a web-based environment for exploring and building models, collaboration features for working with other data scientists, and opportunities to participate in data science competitions.

2.2.12 Model Deployment

Model deployment means putting a machine learning model into action by integrating it into a system or application so that it can make predictions on new data. This process involves preparing the model to work efficiently and accurately in a production environment, and ensuring that it can handle many requests. Model deployment is an important step in the machine learning process and requires careful planning, testing, and monitoring to ensure that the model performs well and provides value to the business.

For the deployment of the paraphrase generation model, Joblib will be used. It is a powerful Python library that can efficiently serialize and deserialize Python objects, including machine learning models. By using Joblib, it is possible to save the trained model to disk and quickly and easily load it back into memory when needed.

3. Methodology

3.1 Data preparation

3.1.1 Collection

As the first step of our project, Data collection was done by analyzing and identifying various datasets in English, Russian, and other languages including MSCOCO[22], ParaNMT, Quora, MSRP[23], ParaSCI-ACL ParaSCI-Axiv[24], and Twitter URL, for modelling our dataset.

3.1.1.1 Quora:

The Quora dataset is composed of question pairs in the English language. Quora dataset consists of 404,351 rows and six columns id, qid1, qid2, question1, question2, and is_duplicate. (i.e. shape of 404351,6).

- Id: The serial id number of the dataset.
- qid1:The id number of question1.
- qid2:The id number of question2.
- question1:Different questions that were asked on the Quora website.
- question2: Questions similar to question 1 that are asked on the quora website.
- is_duplicate: Binary representation of the dataset for duplicates on a row, where '0' represents that they are not duplicates and '1' represents they are duplicates.

question1	question2
What can make Physics easy to learn?	How can you make physics easy to learn?
How should I prepare for CA final law?	How should one know that he/she is completely prepared for the CA final exam?

Table 3.1: Sample Data of Quora Dataset

Out of the 404,351 question pairs in our dataset, 255,045 (63.08%) are labelled as negative (0), while 149,306 (36.92%) are labelled as positive (1). There is a single missing value in the 'question1' column and two in the 'question2' column. Additionally, each question pair is unique within the dataset, with 97.66% consisting of unique questions and 2.34% consisting of repeated questions.

After conducting manual sampling to assess the quality of the data, we found that 'ParaNMT', 'Quora', and 'MSRP' were the best fit for our model. We also collected small quantities of data manually to compare human and computer paraphrasing.

3.1.1.2 ParaNMT:

ParaNMT is a dataset consisting of 50 million sentential paraphrase pairs in English. It is divided into three sections: sentence, paraphrase, and metric value. The metric value indicates the quality score of a sentence and its paraphrase, ranging from 0.35 to 1.

Example from ParaNMT dataset:

Sentence	Paraphrase	Metric
"Only yesterday Sammie sent that across with my supper," he said.	"Yesterday Sammie sent me out with dinner," he said.	0.8452678124
I was there.	I was there and I saw it.	0.83601537078

Table 3.2: Sample Data of ParaNMT Dataset

3.1.1.3 Microsoft Research Paraphrase Corpus (MSRP):

This dataset consists of pairs of paraphrased sentences sourced from various web news sources. Each sentence in the corpus ranges from 7 to 35 words in length, with an average length of 21 words per sentence. Approximately 67% of the sentence pairs are marked as paraphrased. It's worth noting that no more than one sentence was extracted from any given news article to ensure diversity in the dataset.

After conducting manual sampling to assess the quality of the data, we found that 'ParaNMT', 'Quora', and 'MSRP' were the best fit for our model. We also collected small quantities of data manually to compare human and computer paraphrasing.

Sentence	Paraphrase
PCCW's chief operating officer, Mike Butcher, and Alex Arena, the chief financial officer, will report directly to Mr So.	Current Chief Operating Officer Mike Butcher and Group Chief Financial Officer Alex Arena will report to So.
The world's two largest automakers said their U.S. sales declined more than predicted last month as a late summer sales frenzy caused more of an industry backlash than expected.	Domestic sales at both GM and No. 2 Ford Motor Co. declined more than predicted as a late summer sales frenzy prompted a larger-than-expected industry backlash.

Table 3.3: Sample Data of MSRP Dataset

3.1.2 Labelling

Different data collection methodologies, sources, and translations were utilized to obtain the data present in the dataset. Further, to ensure a clear and concise representation of the data, two major labelings were given to the data. Here, the human paraphrased data is labeled as 'Gold,' while the machine-converted datasets that have been filtered are labeled as 'Bronze.'

3.1.2.1 Gold:

Gold data are the exact thousand data that were collected and paraphrased manually. This dataset is divided equally into four specific domains(politics, sports, entertainment and economics).

वाक्य	वाक्यांश	टिप्पणीहरू
आमाबुबासँग नागरिकता छ, छोराछोरीहरूसँग नागरिकता छैन ।	आमाबुबासँग नागरिकता भएता पनि छोराछोरीसँग भने छैन।	politics
रिपोर्टअनुसार विश्वव्यापी मन्दीको डरको बीचमा रोजगारीमा ठूलो खतरा छ।	मन्दि को खतरा को बिचमा रोजगारीको समस्या आउने देखिएको छ।	economics
अर्को समूहमा बलियो टिम भारतसहित बंगलादेश, पाकिस्तान र माल्दिभ्स छन्।	बलियो टोली भारतसँगै बंगलादेश, पाकिस्तान र माल्दिभ्स अन्य विधामा छन् ।	sports
राम्रोसँग बोली नफुटेकी बच्चीलाई हिन्दी तथा नेपाली नयाँ–पुराना गीतहरूमा ‘लिपसिंक’ गर्न लगाइएको हुन्छ।	हिन्दी तथा नेपाली पुराना नयाँ गीतहरूमा ‘लिपसिंक’ गर्न लगाइएको हुन्छ ती राम्रोसँग बोली नफुटेकी नानिलाई ।	entertainment

Table 3.4: Sample of Gold Dataset

3.1.2.2 Bronze:

The Bronze label data consists of previously available data with minor modifications. For instance, we used readily available English paraphrase data from ParaNMT, Quora, and MSRP as our Bronze data. To obtain clean data, we further performed machine translation and filtration on this data. This process helped to refine the Bronze data and improve its overall quality.

3.1.2.3 Translation

To translate, the non-duplicate positive labelled (1) data from the Quora dataset and sentences with metric values ranging from 0.6 to 0.99 from ParaNMT were selected. Here, the translation was done using the Google Translate API.

प्रश्न १	प्रश्न २
के कुराले भौतिक विज्ञान सिक्न सजिलो बनाउन सक्छ?	तपाईं कसरी भौतिक विज्ञान सिक्न सजिलो बनाउन सक्नुहुन्छ?
मैले CA को अन्तिम कानूनको लागि कसरी तयारी गर्नुपर्छ?	सीए फाइनल परीक्षाको लागि आफू पूर्णतया तयार भइसकेको कसरी थाहा पाउने?

Table 3.5: Sample Translation of Quora Dataset

वाक्य	वाक्यांश	मेट्रिक
”हिजो मात्र सामी त्यसलाई मेरो खानासँगै पठाइदिएँ,” उनले भने।	”हिजो सामीले मलाई डिनर गरेर बाहिर पठाउनुभयो,” उनले भने।	०.८४५२६७८१२४४
म त्यहाँ थिए।	म त्यहाँ थिएँ र मैले यो देखेँ।	०.८३६०१५३७०७८१

Table 3.6: Sample Translation of ParaNMT Dataset

वाक्य	वाक्यांश
PCCWका प्रमुख अपरेटिङ अफिसर माइक बुचर र एलेक्स एरिना, मुख्य वित्तीय अधिकारी, सीधै श्री सोलाई रिपोर्ट गर्नेछन्।	वर्तमान प्रमुख अपरेटिङ अफिसर माइक बुचर र समूह प्रमुख वित्तीय अधिकारी एलेक्स एरेनाले सो रिपोर्ट गर्नेछन्।
विश्वका दुई ठूला अटोमेकरहरूले भने कि उनीहरूको अमेरिकी बिक्री गत महिना अनुमान गरिएको भन्दा बढी घटेको छ किनभने गर्मीको अन्त्यमा बिक्रीको उन्मादले उद्योगलाई अपेक्षा गरेभन्दा बढी प्रतिक्रिया दियो।	GM र No. 2 Ford Motor Co. दुवैको घरेलु बिक्री अनुमान गरिएको भन्दा बढी घट्यो किनभने गर्मीको अन्त्यमा बिक्रीको उन्मादले अपेक्षा गरेभन्दा ठूलो उद्योग प्रतिक्रियालाई प्रेरित गऱ्यो।

Table 3.7: Sample Translation of MSRP Dataset

3.1.3 Sampling:

3.1.3.1 Quora:

To prepare the dataset for human evaluation, the dataset was divided into three parts: [0:50,000), [50,000:100,000), and [100,000:149,307), based on an index. Further, random thirty samples were selected from each part.

After analyzing the samples, it was found that 24, 22, and 27 question pairs from the first, second, and third parts, respectively, were suitable for paraphrasing. This indicates that the dataset remained usable even after translation (which accounted for 80.11% of the data) and had yet to be filtered.

3.1.3.2 ParaNMT

To evaluate ParaNMT, we sampled it based on four metrics values, with 30 samples in each range. The metric value ranges were [0.6-0.7), [0.7-0.8), [0.8-0.9), and [0.9-0.99), which resulted in four small subsets.

We accepted 17, 11, 19, and 24 samples from the consecutive ranges, respectively. After observing the statistics, we found that 77 samples were accepted, while 43 samples were rejected, resulting in an overall acceptance rate of 64.2%.

Upon comparison, we noticed that the acceptance rate for the first two ranges ([0.6-0.7) and [0.7-0.8)) was lower than the next consecutive range, and the paraphrases in the metrics range of over 0.95 were too similar to each other, so we rejected the data within that range.

In conclusion, the dataset consists of Nepali-converted sentences with metrics values ranging from 0.8 to 0.95.

3.1.3.3 MSRP

For evaluation of the MSRP dataset, 30 samples were taken randomly from 74260 data out of which 18 were selected while 12 were. were rejected. Our analysis concluded that this can be used in our dataset.

वाक्य	वाक्यांश	टिप्पणीहरू
बोलजानो प्रान्त, ट्रेन्टो, क्षेत्र भेनेटो:	जिल्ला Trentino-Alto Aldige: Bolzano प्रान्त, Trento ।	बकवास
म १६ वर्षको भएँ। मैले के गर्नुपर्छ?	म १६ वर्षको भएँ। मैले के गर्नुपर्छ	समान
२ हप्तामा ५ पाउण्ड तौल घटाउन मैले कस्तो प्रकारको आहार लिन सक्छु?	2 हप्तामा 5 पाउन्ड गुमाउने केहि वैकल्पिक तरिकाहरू के हुन्?	संख्यात्मक त्रुटि

Table 3.8: Example of Random Samples With Remarks

3.1.4 Filtration

During the sampling process, we implemented filtering procedures to detect and remove various elements such as characters, hyperlinks, HTML tags, alphabets, and digits from our dataset. This helped to improve the quality and reliability of our data.

3.1.4.1 First Filtration

id	sentence 1	sentence 2	Remarks
1081112	MSN मेसेन्जर 6 बुधबार साँझ 6 बजे GMTमा http://messen ger. msn. com/download/ v6preview.asp बाट डाउनलोडको लागि उपलब्ध हुनेछ।	MSN मेसेन्जर 6 सफ्टवेयर बुधबार बिहान 11 बजे PST बाट उपलब्ध हुनेछ, माइक्रोसफ्टका अनुसार।	Hyperlink, English Digit, English Characters
1725788.	$21.6बिलियनको बिक्री, एक वर्ष अघि $ 19.7 बिलियनको 10 प्रतिशतले, वाल स्ट्रीटको $ 21.4 बिलियनको अनुमान भन्दा अगाडि थियो।	माइक्रोसफ्टले $८.१ बिलियनको बिक्री पोष्ट गरेको छ, जुन एक वर्ष अघिको ७.३ बिलियन डलरको तुलनामा बढेको छ र विश्लेषकहरूको अनुमानभन्दा $७.९ बिलियन अगाडि छ	Special Characters, English Digits

Table 3.9: Sample of Data Before Filtering from MSRP Dataset

The filtering process removed special characters, including [+-/#*%@.$'():;><!|_•=], as well as alphabets (A-Z, a-z) and digits (0-9). However, a certain character such as "?" was retained since they serve as sentence-ending punctuation.

Furthermore, only sentences with a length between 4 and 24 words were included in the dataset to ensure that it contains sentences of adequate length.

3.1.4.2 Evaluation and Sampling

Sampling fifty data points from the MSRP, Quora, and ParaNMT datasets resulted in only six, eighteen, and nine rejections, respectively. The most common reason for rejection across all three datasets was the presence of sentences that were either meaningless, too similar to other sentences, contained multiple sentences, or had significant deviations from the intended meaning.

From MSRP, Quora and ParaNMT 779, 105550, and 71810 data remained after the first level of filtration.

3.1.4.3 Final Filtration

Initially, we believed that we had successfully removed all unwanted characters from the data. However, upon closer inspection, we discovered that certain unsupported characters, such as `\u200d` and `\u200f`, remained hidden in the data. These characters appeared to be fine in applications like Excel and Google Sheets, as well as within the data frame itself. However, upon individual inspection of the data frame values, we detected these unsupported characters and proceeded to remove them at this stage.

Later on, it was discovered that certain sentences with similar meanings but different wording were translated as duplicates by Google API. To address this issue, similarity metrics were used to identify and remove the duplicates.

3.1.5 Partition

To ensure the accuracy and generalization of the model, the dataset was divided into distinct training, validation, and testing sets. This partitioning strategy enables an unbiased evaluation of the model's performance on unseen data. For the Quora dataset, 4053 pairs (3.8% of the total data) were reserved as the test set, while the remaining data was partitioned into a training set (80%) and a validation set (20%). This partitioning strategy ensures that the model is trained on a sufficiently large dataset while also being evaluated on unseen data to prevent overfitting.

Similarly, the ParaNMT dataset had 71,810 sentence pairs, with 2786 pairs reserved as the test set. The test set was further partitioned into a training and validation set to ensure an unbiased evaluation of the model's performance on unseen data. Overall, this dataset partitioning approach ensures that the models are trained and evaluated on a diverse dataset while preventing overfitting.

To prepare the datasets for model training and evaluation, the training and testing sets for both Quora and ParaNMT were merged separately. This created a bronze train and test set, with the bronze train set being used for fine-tuning the model.

Since the MSRP dataset is small, consisting of only 779 data points, it was used as the bronze test set. Additionally, the gold dataset as a gold test set and bronze test set was used for zero-shot evaluation.

Finally, the Quora and ParaNMT bronze test sets were split and merged for in-shot evaluation. This approach allowed us to comprehensively evaluate the model's performance across various datasets and prepare it for deployment in real-world scenarios.

3.2 Fine tuning models

3.2.1 mT5

mT5 is a multilingual large pre-trained transformer model that was trained on a dataset (mC4 corpus) that contained text in 101 different languages including Nepali. As of now, there are five variants of the mT5 model. mt5-small, mt5-base, mt5-large, mt5-xl, mt5-xxl.

We will be using mT5 over T5 as our model but why ? T5 and mT5 have the same basic transformer architecture. The difference between these two is that T5 is trained on the monolingual corpus (English) while mT5 is trained on the multilingual corpus(over 101 languages). Most importantly mT5 is useful for Nepali language sequence-to-sequence generation. That's why we have used the mT5 model over the T5 model in our project.

In this project, we first fine-tuned the mT5 model so that we could validate our dataset and see the outcomes. Fine-tuning entails taking a pre-trained model and training it on a smaller dataset specifying task. For the purpose of our project, we are mainly using a small and base version of the mT5 model due to the limitation of hardware resources.mT5-small is the smallest lightweight version of the mT5 model mainly suitable for low-resource environments where computational resources are limited.mT5 small presents 300 million trainable parameters in contrast to 580 million parameters for the mt5-base model which is a slightly heavier version of the mT5 model compared to the small model.

3.2.1.1 mT5 small

Here for the purpose of our project, first we divided the fine-tuning task for paraphrasing into two parts: one with prefix and one without prefix. Each is further trained three different times for different datasets named quora, paranmt, and combine data of quora and paranmt.For the training purpose we keep a similar hyperparameter for all training runs to ensure consistency across the experiments. Here, after processing the data as previously stated in the document, we tokenize our data using the mt5 small tokenizer. We set the input maximum length for the tokenizer to 256 tokens, indicating that any input text larger than that will be truncated. Along with padding with the maximum length, we have also configured the addition of special tokens. The tokenizer aids in ensuring that the input to the model is consistent and manageable by limiting the maximum input length and applying truncation.

Hyperparameter choices were the following :

- Learning rate:1e-4
- Weight decay:0.1
- Train batch size:16
- Validation batch size:8
- Gradient accumulation steps:16
- Input max length:256
- Output max length:128
- Adam epsilon:1e-8

The training process was optimized with careful consideration of the hyperparameters used. A learning rate of 1e-4 and a weight decay of 0.1 were selected to ensure optimal performance. Considering the limitation of resources, a batch size of 16 was utilized for training purposes, and a batch size of 8 was adopted for validation. To speed up the training process and reduce memory usage, a gradient accumulation technique was employed with a step size of 16. The maximum length for the input and output sequences was set to 256 and 128, respectively. Finally, to optimize the model weights during training, the Adam optimizer was used with a small epsilon value of 1e-8. Overall, these hyperparameters were chosen carefully to balance model performance and training efficiency. Finally, after training the model for a specified number of epochs we saved the best checkpoint based on the validation loss and used it for evaluating the performance of the fine-tuned model on the test set.

3.2.1.2 mT5 base

To fine-tune the mT5 base model, a similar set of hyperparameters to those used for the mT5 small model above was selected, with the exception of adjusting the learning rate to 3e-4, which was chosen based on careful consideration of the mT5 base model's characteristics. The maximum lengths for input and output sequences were kept the same as before at 256 and 128, respectively. Following the same procedure as before,the best checkpoint of the model was saved, and evaluation was performed using it.

Train step loss:
Initially, the train step loss starts at 4 and subsequently decreases with each step, although there are occasional increases. By the final step of our model training, the train step loss converged to 0.6. The graph of the train step loss over the course of the training shows a clear convergence of the loss values, which were initially relatively high but steadily decreased as the model improved. Overall, these results suggest that the model has successfully learned and adapted to the training data, resulting in improved performance. The fact that the train step loss curve continues to decrease without flattening suggests that the loss is converging even further. However, due to hardware limitations, we were forced to stop the training before the loss could converge completely. Continuing training beyond 5 epochs could potentially result in further convergence of the loss. Nonetheless, the curve clearly shows that the model was continuing to learn and improve over time, indicating that further training could result in even better performance.

3.2.2 Back-Translation

Back-translation is a technique used in machine translation to improve the quality of translations by translating a text from the target language back into the source language. This technique involves training a neural machine translation model on a parallel corpus (a set of texts in the source and target languages) and then using the model to translate a text from the target language back into the source language. The resulting text is then compared to the original source text to identify errors and improve the quality of the translation. Back-translation is a simple and effective way to generate additional training data for machine translation models and has been shown to improve the quality of translations in several studies. It is commonly used in both academic research and commercial machine translation systems.

In this project, we have used the back translation method as an alternative/optional method for the generation of paraphrases of the Nepali language. Basically, back translation in

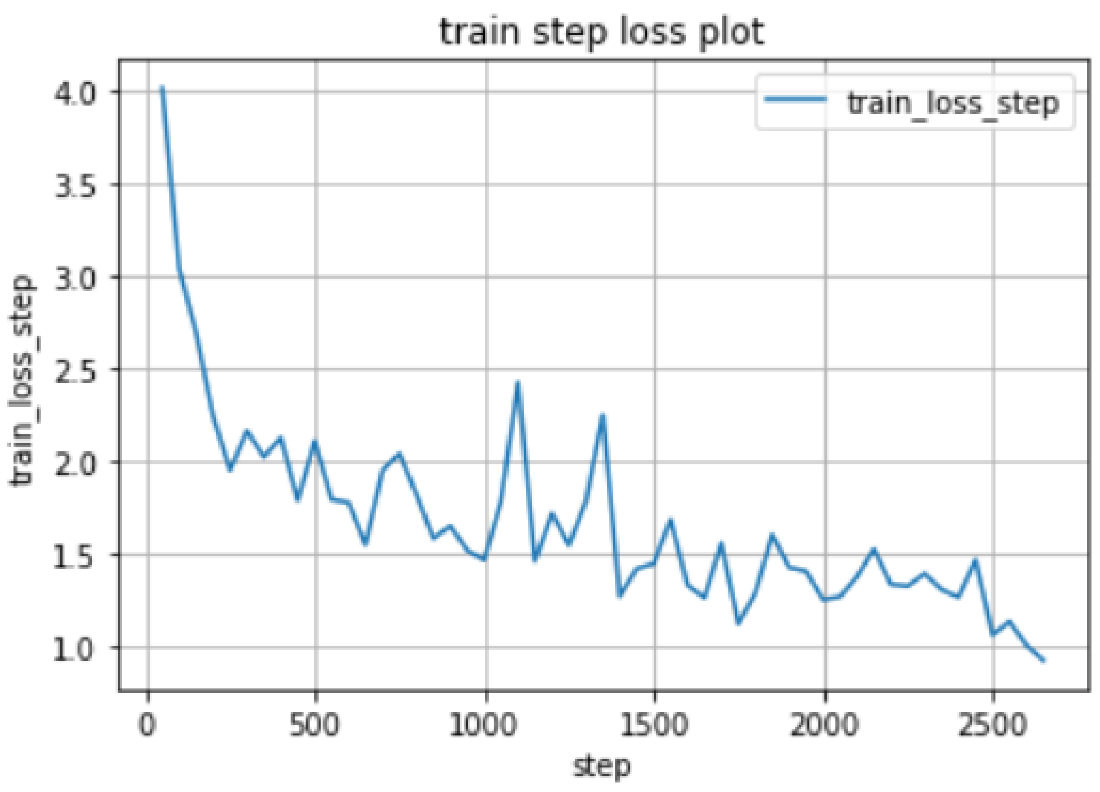

Figure 3.1: Train step loss

our scenario means that we will convert our Nepali sentences into English sentences. The obtained English sentences are then paraphrased using one paraphrasing tool. Then this paraphrased English sentence is translated back into the Nepali language.

The basic idea here is that, when we convert the English language to Nepali in the second phase, the sentence obtained is syntactically different from the initial Nepali sentence fed into the back translation model. It is because when the English sentence is paraphrased, the syntax and sentence structure of an English sentence is different from the non-paraphrased one (original one). And the back-translation also follows the same sequence.

Translation model: For the translation from a Nepali sentence to an English sentence and vice versa, we used the 'translate' method available in the transformers.

Paraphrase model: For the English sentence paraphrase mode, we searched for the best paraphrase model available in the hugging face library and found that 'Parrot Paraphraser' is the model with the highest number of users in the NLP society. That's why we used the parrot paraphraser for paraphrasing our Nepali to English-translated sentences.

3.3 Deployment

While using the large model for our paraphrasing task, considered utilizing joblib to dump the model to disk and reload it into memory as necessary. It is important to save the model for future use to make a prediction on unseen data. The saving of data is called Serialization while restoring the data is called Deserialization.

Save : joblib.dump to serialize an object hierarchy

Load : joblib.load to deserialize a data stream

Inference is a crucial step in the machine learning workflow, as it allows models to be applied to real-world prIoblems. In our project, we have used the combination of top-k and top-p samplings in order to generate the paraphrase of the input sentences.

While in theory, Top-p seems more elegant than Top-K, both methods work well in practice. Top-p can also be used in combination with Top-K, which can avoid very low-ranked words while allowing for some dynamic selection. That's why we have used top-p in combination with top-k in our inference code with the following values:

$$Top - k = 40$$

$$Top - p = 0.95$$

After dumping the large model, we loaded the model and passed the unseen sentences through the inference code to generate the paraphrase. The frontend interface was built in HTML and CSS while the backend was coded in Django. The sentence was accepted on one side of the text area and the other text area was coded to generate the paraphrase of the sentence. The pictorial representation can be seen below:

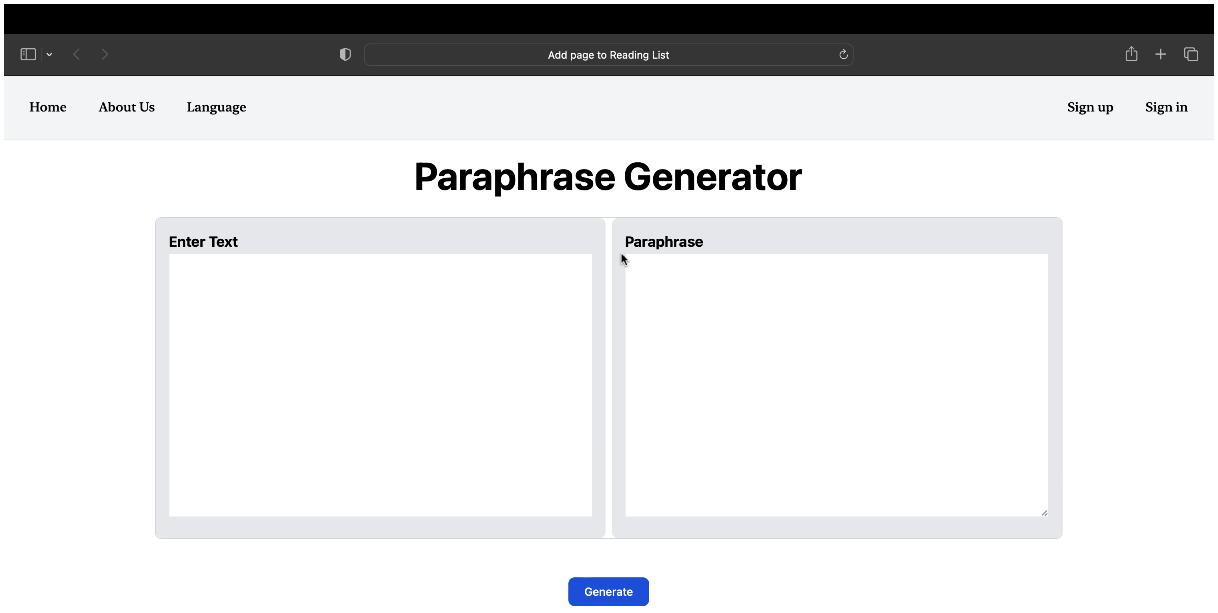

Figure 3.2: User Interface

4. System Block Diagram

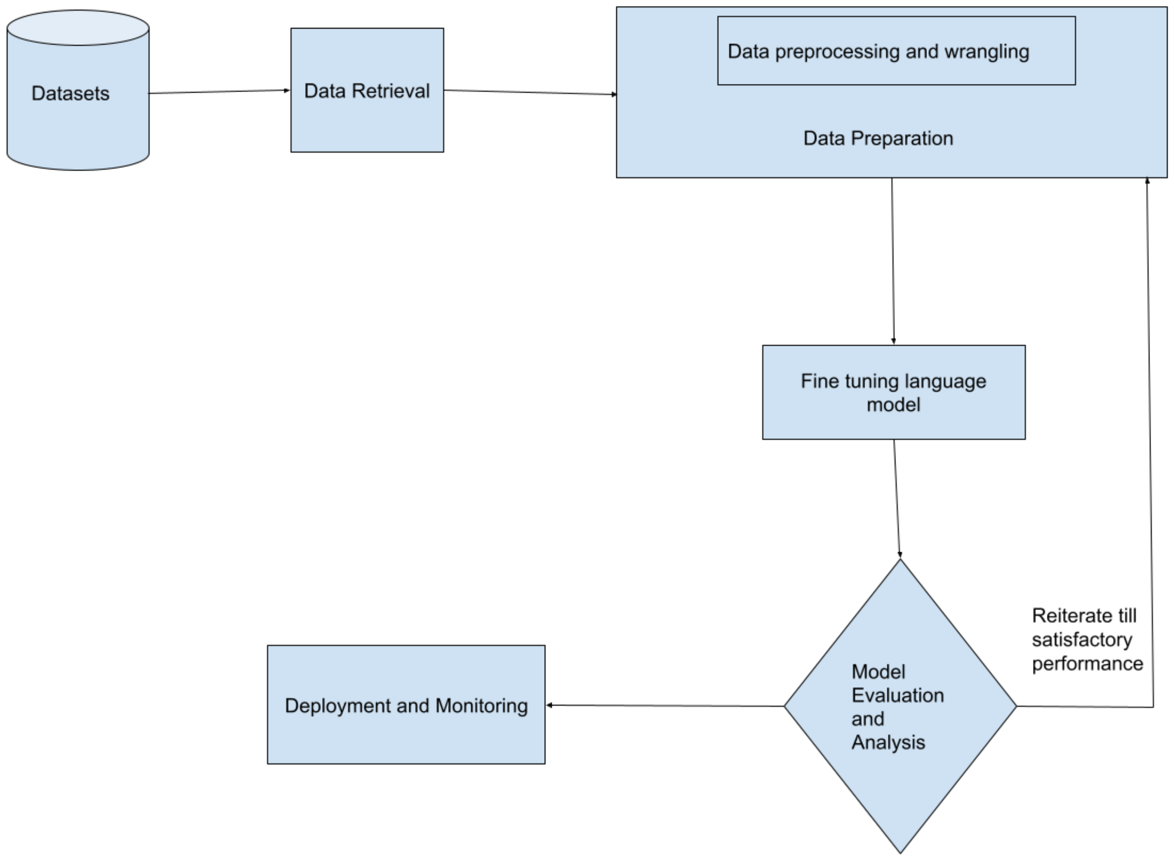
Datasets
Data Retrieval
Data preprocessing and wrangling
Data Preparation
Fine tuning language model
Model Evaluation and Analysis
Reiterate till satisfactory performance
Deployment and Monitoring

Figure 4.1: System Block Diagram

The above picture depicts our whole system.

5. Results

After the model is trained, comparative analyses of different models with different test data were performed. Two types of analysis called quantitative and qualitative analysis were performed. The overall sum of this two analysis are:

5.1 Quantitative analysis:

Quantitative analysis is a research method that employs mathematical and statistical techniques to analyze and understand complex phenomena. In the realm of NLP, quantitative analysis has become an essential tool for analyzing large datasets of text and extracting valuable insights that can be used to build intelligent systems that can understand and generate human-like language.

5.1.1 Bleu2 score analysis

Set Identity	TestSet	MT5							Backtranslation
		Small-without-prefix			Small-with-prefix			Base	
		Quora	Para	All	Quora	Para	All	All	All
From set	Quora	0.49	-	*0.54*	0.50	-	*0.54*	**0.55**	0.46
	Para	-	0.51	0.57	-	0.48	*0.58*	**0.63**	0.49
Zero shot	Bronze	0.36	0.40	0.50	0.36	0.37	0.51	**0.58**	*0.52*
	Gold	0.42	0.46	*0.54*	0.43	0.42	*0.54*	**0.58**	0.50

Table 5.1: Bleu2 score analysis table

For instance, let's consider the Quora test set. In this particular evaluation, the model achieved an impressive BLEU-2 score of 0.55, indicating its remarkable ability to generate accurate rephrasings of the sentences that were given as input. Additionally, the model demonstrated excellent performance on the ParaNMT (referred to as Para onwards and in every table), Bronze, and Gold sets, showcasing its ability to generalize across different types of test sets. Out of all the test sets, the mT5 base model performed the best on the Para set. Overall, when compared to other models, the mT5 base model showed superior precision across various test sets.

Set Identity	TestSet	MT5							Backtranslation
		Small-without-prefix			Small-with-prefix			Base	
		Quora	Para	All	Quora	Para	All	All	All
From set	Quora	0.40	-	*0.45*	0.41	-	*0.45*	**0.46**	0.33
	Para	-	0.43	*0.49*	-	0.40	*0.49*	**0.54**	0.37
Zero shot	Bronze	0.26	0.32	**0.49**	0.26	0.29	0.42	*0.48*	0.38
	Gold	0.31	0.37	*0.44*	0.32	0.40	0.43	**0.46**	0.35

Table 5.2: Bleu4 score analysis table

5.1.2 Bleu4 score analysis

The bleu4 score analysis was conducted to evaluate the performance of different test sets in the mT5 base and small models. The results showed that overall, the mT5 base model performed better than the small model, except for the bronze set, where the small model outperformed the base model. It is important to note that the bronze set without a prefix in the small model contributed to its higher score.

Despite this exception, the mT5 base model demonstrated superior performance on all other test sets. In particular, the para-test set yielded impressive results, indicating that the model was able to accurately generate paraphrases of the sentences in the set.

Overall, these results highlight the impressive capabilities of the mT5 base model in generating high-quality paraphrases of various sentences across multiple test sets. While the small model may have outperformed the base model on the bronze set, it is clear that the base model is the more robust and reliable option for generating paraphrases to a diverse range of sentences.

5.1.3 Rouge1 score analysis

After analyzing the BLUE-2 and BLEU-4 scores, we conducted a Rouge-1 score analysis to evaluate the performance of different models on various test sets. The Quora test set consistently performed well across all three models, including the mT5 small model with and without prefix, and the mT5 base model, suggesting that the model's performance on this set is stable and reliable. The back-translation method did not perform well on any of the test sets, which may indicate a low recall of words in generated paraphrases compared to input sentences. However, the para test set achieved commendable scores in the Rouge1 score analysis, demonstrating exceptional model performance on this particular set. The

Set Identity	TestSet	MT5							Backtranslation
		Small-without-prefix			Small-with-prefix			Base	
		Quora	Para	All	Quora	Para	All	All	All
From set	Quora	*0.40*	-	**0.43**	*0.40*	-	**0.43**	**0.43**	0.27
	Para	-	0.43	*0.46*	-	0.40	*0.46*	**0.48**	0.28
Zero shot	Bronze	0.23	0.36	0.38	0.24	0.35	*0.39*	**0.42**	0.25
	Gold	0.23	*0.35*	**0.36**	0.24	0.32	**0.36**	**0.36**	0.21

Table 5.3: Rouge1 score analysis table

zero-shot evaluations produced underwhelming results, suggesting a low recall score for new sentences and highlighting the need for further improvements in the model.

5.1.4 Rouge2 score analysis

Set Identity	TestSet	MT5							Backtranslation
		Small-without-prefix			Small-with-prefix			Base	
		Quora	Para	All	Quora	Para	All	All	All
From set	Quora	*0.18*	-	**0.20**	*0.18*	-	**0.20**	**0.20**	0.08
	Para	-	0.21	*0.23*	-	0.20	*0.23*	**0.24**	0.08
Zero shot	Bronze	0.09	0.18	0.19	0.09	0.17	*0.20*	**0.21**	0.07
	Gold	0.08	*0.13*	**0.14**	0.08	0.11	**0.14**	*0.13*	0.04

Table 5.4: Rouge2 score analysis table

In addition to the recall score for Rouge 1, we also evaluate the quality of our paraphrase by measuring its recall score for bigram words using Rouge 2. This helps us determine how well the generated paraphrase captures the meaning of the input sentence by looking at pairs of consecutive words. The results show that there is a slight variation in the performance of different models on different test sets. Specifically, the combined training data of Quora and Para datasets performed well in both the mT5 small model with and without prefixes on the gold test set. This indicates that the combined training data may be particularly effective in generating paraphrases for various sentences in the gold test set.

However, the mT5 base model consistently outperformed other models on most test sets, particularly on the Para test set.

In conclusion, it is important to consider a range of evaluation metrics when assessing the performance of different models. While the Rouge2 score analysis highlighted some variation in performance, the overall results suggest that the mT5 base model is the best choice for generating paraphrases.

5.2 Qualitative analysis

In the context of NLP and text-based evaluation, qualitative analysis refers to the examination and interpretation of the content, style, and context of machine-generated translations or summaries. Unlike quantitative analysis, which relies on numerical metrics such as BLEU and ROUGE scores, qualitative analysis involves analyzing the language itself and looking for patterns, themes, and other meaningful insights.

Owing to the shortcomings of the mathematical model, quantitative analysis alone cannot determine the quality of the generated paraphrases. Hence, in order to compare and evaluate our finetuned models, we used human evaluation. Here, we conducted a human evaluation on all three models developed: mT5-small without a prefix, mT5-base, and back-translation method.

We run into a variety of errors during the assessment, including entity mismatch problems, semantic and syntactic differences, and exact match problems between input sentences and their respective generated sentences. The table below summarizes the human evaluation of all the models on the test sets.

Testset	Input	Paraphrase	Remarks
Quora	हामी प्रेममा पर्दैछौं भनेर कसरी थाहा पाउन सक्छौं?	तपाईं प्रेममा पर्दैछौं भनेर कसरी थाहा पाउन सक्छु?	Paraphrase with change in pronoun.
	म कसरी धाराप्रवाह अंग्रेजी बोल्न सक्छु?	म कसरी बोल्न सक्छु?	Full meaning of sentence not captured.
ParaNMT	उनीहरुको हत्या भएको छ ।	उनीहरुको हत्या छ।	Slight semantic dissimilarity.
	के तपाईंले ग्यारेटसँग कुरा गर्नुभयो?	ग्यारेटसँग कुरा गर्नुभयो?	Paraphrased well.
Bronze	विदेशी पशु व्यापार ठूलो छ र यो नियन्त्रण बाहिर सर्पिल जारी छ।	तर विदेशी पशु व्यापार ठूलो छ भने यो नियन्त्रण जारी छ।	Paraphrase obtained with slight semantic difference.
	पार्क भित्र वा नजिकै तीन अन्य आगो बुधबार नियन्त्रण गरियो।	पोक भित्र वा नजिकै तीन अन्य आगो नियन्त्रण गरियो।	Not captured the full meaning of original sentence but slightl change in sentence structure which gives sense of parphrase.
Gold	उनलाई त्यहाँ अनुकुल हुन लामो समय लाग्यो ।	उनले त्यहाँ अनुकुल हुन लामो समय लाग्यो ।	Slight change in pronoun.
	बार्सिलोनाका लागि राम्रो प्रदर्शन गरिरहेको छु ।	मैले बार्सिलोनाका लागि राम्रो प्रदर्शन गरिरहेको छु।	Addition of pronoun but semantic and syntactic error are preserved.

Table 5.5: Sample analysis for mT5-small without prefix trained on all data

Testset	Input	Paraphrase	Explanation
Quora	म कसरी विद्यालयमा बढी केन्द्रित हुन सक्छु?	म कसरी विद्यालयमा केन्द्रित हुन सक्छु?	Paraphrased well.
	शान्ति सेनाको अनुभव कस्तो रह्यो ?	शान्ति सेनामा सेवा गर्न कस्तो लाग्छ?	Paraphrased well.
ParaNMT	मेरो भगवान रायन मलाई धेरै माफ गर्नुहोस्।	मेरो भगवान रायन मलाई माफ गर्नुहोस्।	Paraphrased with one word removal.
	शगुनको तरवार अब मलाई असफल नगर्नुहोस्।	शगुनको तरवार मलाई असफल गर्न सक्नुहोस्।	Semantically opposite sentence obtained.
Bronze	कुन चेनलाई लक्षित गर्ने संघले अहिलेसम्म खुलाएको छैन।	विश्वविद्यालयले अहिलेसम्म कहिल्यै कुन चेनलाई लक्षित गर्ने संघलाई खुलाएको छैन।	Paraphrased good.
	उनको एउटै सर्त थियो कि उनको मृत्युसम्म पुस्तक प्रकाशित हुँदैन।	उसको एउटै सर्त थियो कि ऊ मेरो जन्म सम्म प्रकाशित हुँदैन।	Semantic meaning not captured.
Gold	फिल्ममा रणवीर कपुर विशेष भूमिकामा देखिनेछन्।	विशेष भूमिकाका साथ फिल्ममा रणवीर कपुर देखिनेछन्।	Perfectly paraphrased
	अमेरिकी डलरको तुलनामा जापनिज येन पनि कमजोर बन्द गएको छ।	जापानको मुद्रा ऐन पनि अमेरिकी डलरको तुलनामा कमजोर हुँदै गएको छ।	Perfectly paraphrased.

Table 5.6: Sample analysis for mT5-base trained on all data

Testset	Input	Paraphrase	Explanation
Quora	मैले किन डोनाल्ड ट्रम्पलाई भोट दिनु पर्दैन?	मानिसहरूले किन तुरहीको लागि मत नदिने?	Entity mismatch problem but meaning is retained along with sentence structure.
	तपाई अब जवान हुनुहुन्न भन्ने के थाहा पाउनु भयो?	जब तिमी जवान छैनौ, तिमी कसरी जान्दछौ?	Paraphrase obtained but change in semantic structured.
ParaNMT	काउन्टी न्यायाधीश टिम हार्लीले निर्णायकको निर्णयमा पुग्न नसकेका कारण मिस्ट्रियल घोषणा गरे।	काउन्टी न्यायाधीश टिम हार्लीले निर्णायकको निर्णयमा पुग्न नसकेका कारण मिस्ट्रियल घोषणा गरे।	Generated paraphrase matches exactly with the input.
	फायर डिपार्टमेन्टका बटालियन प्रमुख जोन जेकोबी आइपुगे।	जोन जाकोबी, हतियार विभागका प्रमुख आइरहेका छन्	Entity mismatch problem.
Bronze	कादिरोभ घाइते भएनन् तर मारिनेमा उनका चार अंगरक्षक थिए।	कदिरोभलाई चोट लागेको थिएन तर उसको चार अंगरक्षकलाई चोट लागेको थियो	Perfect paraphrase obtained.
	प्रतिरक्षाले रुसको फैसलाको सुनुवाइ नभएसम्म अपील गर्न सक्दैन।	जबसम्म त्यो निर्णय सुनाइने छैन तबसम्म प्रतिवादीलाई अपील गर्ने कुनै विकल्प छैन	Good paraphrase obtained with few additional words.
Gold	अहिले कुरा सम्भदै गर्दा एउटा भिडिओ याद आयो।	मैले सोचिरहेको एउटा भिडियो सम्भन्छु	Paraphrased obtained with slight semantic difference.
	अहिले एक्दमै पेचिलो भागबन्डाको विषय बनेको छ ।	यो अहिले अति जटिल भागब्यान्डको विषय हो	Paraphrased .

Table 5.7: Sample analysis for backtranslation

Testset	mT5				mT5-base				Backtranslation			
	EM^*	SE^*	MM^*	GE^*	EM^*	SE^*	MM^*	GE^*	EM^*	SE^*	MM^*	GE^*
Quora	2	*3*	0	**4**	*3*	2	0	0	0	*4*	**3**	2
Para	**9**	*7*	*1*	*1*	**6**	**3**	0	0	0	**5**	*1*	*3*
Bronze	*3*	5	0	*1*	2	**3**	0	0	**1**	2	0	*3*
Gold	*3*	**10**	**2**	*1*s	2	**3**	0	0	0	*4*	*1*	**5**

Table 5.8: Error analysis table

*EM = Exactly Matched
*SE = Semantic Error
*MM = Entity Mismatch
*GE = Grammatical Error

As from the table, the mT5-small model contains a significant number of unwanted results (EM*, SE*, MM*, GE*) whereas the mT5-base model has a drastic improvement in this aspect. The main difference between those is that the mT5-small model produced a total of 3 entity mismatches and 7 syntactic errors whereas the mT5-base model reduced those errors. Also, there are a lot of exact match problems and semantic dissimilarity issues in all of the mT5 small test sets, however, these issues are less prevalent in the mT5 base but continue to exist there. For the mT5 base method, it was found that we can alleviate semantic problems by training on a large language model with a high number of trainable parameters, as the language understanding of these models is better than that of smaller models.

Human evaluation has shown that the back translation method has fewer errors than other methods when an exact match is taken into account. However, it has a higher number of grammatical errors when compared to other methods, indicating that our trained model performs well in this aspect. The back-translation method performs worse than the base model in all areas except for exact match errors, while compared to the mT5 small model, the back-translation method performs significantly better.

Additionally, the exact match problem can be mitigated by running the generation process multiple times for a given sentence until a satisfactory paraphrase is generated. During testing, it was found that by running the generation process an average of three times for given input in the base model, it is guaranteed to generate a paraphrase. Regarding the other human evaluation data, it was found that the quality of paraphrases generated by the back translation method and the mT5 base model were similar in quality, while the quality of mT5 small model was not up to mark with the other two.

6. Conclusion

In summary, we've developed a paraphrasing tool for the Nepali language using an MT5 base model that underwent training with diverse datasets and evaluation through BLEU and ROUGE scores.

This powerful tool aims to enhance the writing and communication abilities of Nepali speakers by providing them with accurate and effective paraphrasing options. Further, our findings indicate that our tool can produce high-quality paraphrases in Nepali, making it an excellent resource for multiple natural language processing applications. With this tool at their fingertips, Nepali speakers can communicate more effectively and achieve greater success in their writing endeavours.

Additionally, we conducted several experiments using both the MT5 small model and a back-translation model. Our findings revealed that while the MT5 base model successfully overcame various errors present in the MT5 small model, it still fell short of the well-developed back-translation model in terms of performance. Despite this, our research has provided valuable insights into the strengths and weaknesses of these different models and highlighted the potential for future improvements.

Ultimately, our research illuminates the vast potential of advanced language models to enhance natural language processing capabilities in low-resource languages and break down linguistic barriers. By demonstrating the effectiveness of our approach in overcoming linguistic hurdles, we hope to inspire further innovation in this field and promote greater language access and inclusion for all.

7. Limitation

While doing this project, we faced many challenges and we have overcome many of them. Some of the major challenges we faced during the accomplishment of our goal in this project are as follows:

- Lack of good amount of data. Since we are one of the very few who have worked in the field of Nepali language NLP field, it was a very tough job to find a sufficient amount of data. However, we are proud that we will be counted among the 'Data Creators' in this heavily unexplored field.

- Lack of computational power to train the model As our model is very heavy with more than a billion parameters, the lack of enough computational power to train the model was a huge challenge. We still feel that, with the availability of a better computational engine, we can achieve even better results in this project.

  There are some limitations of our project as well. We have created a good paraphrase model for the Nepali language but we are yet to apply this model in a beneficiary way. It will take some time for us to come up with an application-ready model for our project. As this project is a sequence-to-sequence generation model, there is no exact way to evaluate the performance of the model by a machine or a program. So, human evaluation is the optimal way of evaluation. Also, we are yet not able to compare the input sentence and output paraphrase semantically, which would be the best way for the evaluation of the model.

8. Future work

Efforts are currently underway to enhance the evaluation metrics for measuring semantic and syntactic proficiency in the Nepali language. Future plans include the addition of new data to the existing dataset and fine-tuning larger models to improve accuracy. To generate more contextually appropriate paraphrases that account for the surrounding text, controlled paraphrase generation is being developed. In addition, new tasks such as paraphrase identification and plagiarism detection are being introduced to enhance language processing capabilities.

Bibliography

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[3] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[4] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[5] Lakshay Sharma, Laura Graesser, Nikita Nangia, and Utku Evci. Natural language understanding with the quora question pairs dataset. *arXiv preprint arXiv:1907.01041*, 2019.

[6] John Wieting and Kevin Gimpel. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *arXiv preprint arXiv:1711.05732*, 2017.

[7] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

[8] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[9] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[10] Kathleen McKeown. Paraphrasing questions using given and new information. *American Journal of Computational Linguistics*, 9(1):1–10, 1983.

[11] Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. Application-driven statistical paraphrase generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 834–842, 2009.

[12] Igor A Bolshakov and Alexander Gelbukh. Synonymous paraphrasing using wordnet and internet. In *International Conference on Application of Natural Language to Information Systems*, pages 312–323. Springer, 2004.

[13] David Kauchak and Regina Barzilay. Paraphrasing for automatic evaluation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 455–462, 2006.

[14] Raymond Kozlowski, Kathleen F McCoy, and K Vijay-Shanker. Generation of single-sentence paraphrases from predicate/argument structure using lexico-grammatical resources. In *Proceedings of the second international workshop on Paraphrasing*, pages 1–8, 2003.

[15] Chris Quirk, Chris Brockett, and Bill Dolan. Monolingual machine translation for paraphrase generation. 2004.

[16] Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. Neural paraphrase generation with stacked residual lstm networks. *arXiv preprint arXiv:1610.03098*, 2016.

[17] Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. A deep generative framework for paraphrase generation. In *Proceedings of the aaai conference on artificial intelligence*, volume 32, 2018.

[18] Hemant Palivela. Optimization of paraphrase generation and identification using language models in natural language processing. *International Journal of Information Management Data Insights*, 1(2):100025, 2021.

[19] Sulav Timilsina, Milan Gautam, and Binod Bhattarai. Nepberta: Nepali language model trained in a large corpus. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*, pages 273–284, 2022.

[20] Milan Tripathi. Sentiment analysis of nepali covid19 tweets using nb svm and lstm. *Journal of Artificial Intelligence*, 3(03):151–168, 2021.

[21] Bishwo Prakash Pokharel. Twitter sentiment analysis during covid-19 outbreak in nepal. *Available at SSRN 3624719*, 2020.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[23] Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[24] Qingxiu Dong, Xiaojun Wan, and Yue Cao. Parasci: A large scientific paraphrase dataset for longer paraphrase generation. *arXiv preprint arXiv:2101.08382*, 2021.