# "Enhancing Question Quality Using Transformer Models: A Self-Paced Research Assessment"

by **Chata.ai**

completed by Pankti Shah

# Abstract

This project explores the use of transformer models, specifically T5-small, for enhancing question quality by rewriting disfluent and unclear questions into refined, concise versions. The study focuses on applying Natural Language Processing (NLP) techniques to remove disfluencies, filler words, and irrelevant content, thereby improving the clarity and precision of questions. The dataset, consisting of paired disfluent and original questions, was preprocessed and analyzed to identify key patterns in word choice and length. The T5-small model was trained and evaluated using metrics such as BLEU, ROUGE, and accuracy, demonstrating its effectiveness in generating clearer, more direct questions. This research contributes to developing automated solutions for question refinement, with potential applications in educational technology, customer support, and AI-driven communication tools.

# Table of Contents

# List of Figures

# 1. Literature Review

The advancements in transformer models have profoundly impacted natural language processing (NLP), especially in tasks such as question rewriting, which is the focus of my project. My work involves leveraging these state-of-the-art models to enhance question clarity and relevance. The paper [1] **"Pre-trained Language Models for Text Generation: A Survey"** underscores the importance of pre-trained models in generating coherent text, aligning with my use of T5 for text rewriting. **"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"** [2] reveals how bidirectional context capture, as demonstrated by BERT, improves task performance, informing my approach to model selection and evaluation. The [3]**"T5: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer"** paper provides insights into using a unified text-to-text framework, which parallels my application of T5 to handle diverse rewriting tasks. Additionally [4], **"PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization"** introduces techniques that, while focused on summarization, are adaptable to question rewriting by generating concise responses. Lastly [5], **"BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension"** highlights how combining bidirectional and autoregressive training enhances sequence generation, which is relevant to my evaluation of BART for improving question rewriting accuracy. These foundational studies provide a robust backdrop for my project, guiding the application and optimization of transformer models to advance the quality and effectiveness of question rewriting.

# 2. Methodology

The aim of this research is to use a **sequence-to-sequence** (Seq2Seq) approach to construct a **question rewrite** model. This model seeks to generate precise and well-structured queries from noisy or partial user inputs in **conversational workflows**. Ensuring appropriate question responding and purpose prediction in the face of input errors resulting from typos or breaks in users' thought processes necessitates these changes.

## 1. Overview of the Dataset

A benchmark dataset called **Disfl-QA** was developed to examine contextual disfluencies in question-answering contexts over Wikipedia passages. Using the accompanying paragraph as a source of distractions, it expands upon the SQuAD-v2 dataset by adding contextual disfluencies, such as corrections or restarts, to each item in the development set. The dataset presents a difficult test for models since it contains about 12,000 pairs of disfluent questions and replies, with over 90% of the disfluencies being corrections or restarts. With Disfl-QA, a standard for assessing model robustness against disfluent inputs is provided, with the goal of bridging the gap between the speech and NLP research communities. There are 7,182, 1,000, and 3,643 questions in the train, test, and development splits, respectively. Experiments have shown that current state-of-the-art

models struggle with disfluent inputs, highlighting the dataset's utility in testing model robustness.

### 1.1 Visualizing Training Data from train.json:

**Sample Disfluent Questions:**
1: petrologists unstable isotope study indicate
2: second level territorial division poland make basic unit territorial division warsaw
3: juvenile platyctenids wow genus lack tentacle sheath

**Sample Original Questions:**
1: unstable isotope study indicate
2: basic unit territorial division warsaw
3: genus lack tentacle sheath

### 1.2 Summary of Data Characteristics

1. **Length Statistics:**
    - **Disfluent Questions:**
        - **Mean Length:** 7.15 words
        - **Median Length:** 7 words
        - **Maximum Length:** 41 words
        - **Minimum Length:** 0 words
    - **Original Questions:**
        - **Mean Length:** 5.11 words
        - **Median Length:** 5 words
        - **Maximum Length:** 17 words
        - **Minimum Length:** 0 words

The disfluent questions are generally longer than the original questions, with a higher mean and maximum length. This reflects the presence of redundant or irrelevant phrases in the disfluent questions that are typically removed in the original versions.

2. **Most Common Words:**
    - **Disfluent Questions:**
        - Frequently occurring words include conversational fillers or ambiguous terms such as "sorry," "rather," "mean," "make," and "actually." These words often contribute to the disfluency and complexity of the questions.
    - **Original Questions:**

- Common words are more specific and relevant to the context, such as "many," "type," "school," "year," "university," "name," "cell," and "force." These words suggest a focus on specific topics or information, aligning with the goal of producing clearer and more concise questions.

### 1.3 Insights:

The data reveals a clear distinction between disfluent and original questions in terms of both length and word choice (see Fig. 1). Disfluent questions typically contain more unnecessary words or phrases, resulting in longer average lengths. In contrast, original questions are generally shorter and make use of more contextually relevant and specific words, reflecting their goal of clarity and conciseness. This difference highlights the impact of removing disfluencies and optimizing word choice in the rewriting process, contributing to more effective communication.

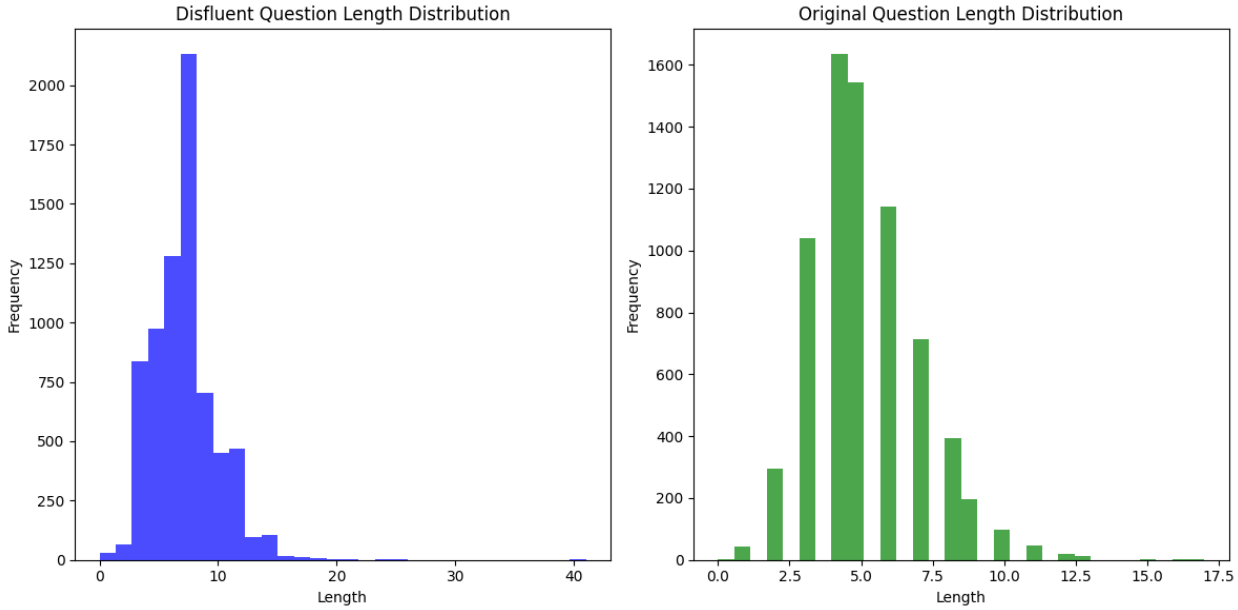


**Fig 1**. Length distribution of data

## 2. Data Cleaning and Preprocessing

To ensure the model effectively handles noisy inputs, several data preprocessing steps were applied as shown in Fig 2. First, disfluency detection and removal targeted common filler words and disfluencies such as "uh," "um," "like," and "you know," which frequently appear in conversational speech but do not contribute to the semantic meaning of a question. These words were filtered out to reduce noise and improve input clarity. Next, tokenization and lemmatization were performed using spaCy and NLTK tools. Tokenization split the text into individual words or tokens, while lemmatization converted these tokens to their base or root forms, helping to standardize different morphological variants of the same word (e.g., "running" to "run"). This process was accompanied by stopword removal,

where commonly occurring words that do not add significant meaning (like "the," "is," "and") were excluded, allowing the model to focus on the essential content of the questions. Further, text normalization was carried out to standardize the input, involving converting text to lowercase to ensure uniformity, expanding contractions (like "can't" to "cannot"), and removing unnecessary characters, punctuation, or extra spaces. These combined preprocessing steps aimed to transform the disfluent inputs into a more structured, clean, and semantically meaningful format, thereby enhancing the model's ability to learn and generalize effectively.

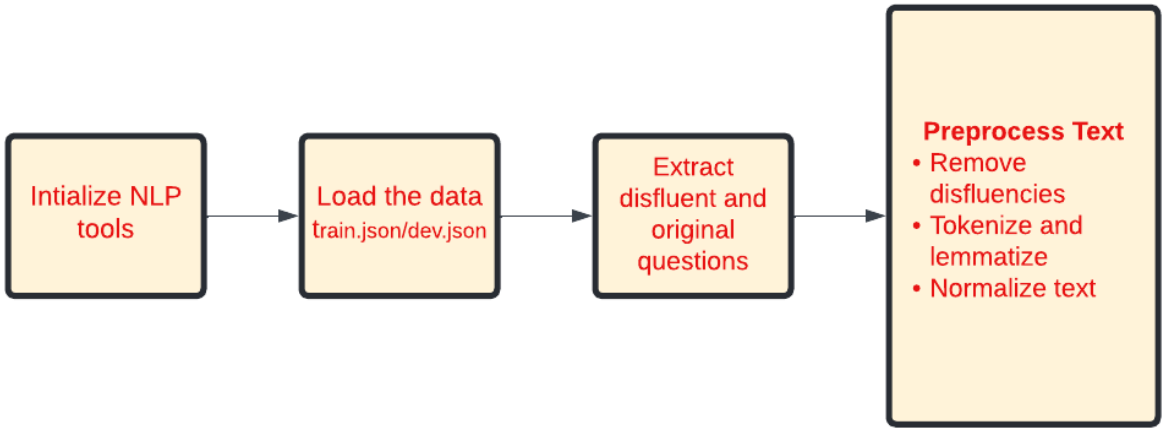


**Fig 2**. Data pre-processing steps

## 3. Model Architecture

The `t5-small` model is a compact version of Google's T5 (Text-to-Text Transfer Transformer) designed for various natural language tasks. With around 60 million parameters, it is the smallest in the T5 family, making it fast and efficient. This model can handle tasks like translating text between languages, summarizing articles, rewriting sentences, or answering questions, all by treating them as a simple text input-to-output problem.

The `t5-small` model learns patterns in text by reading large amounts of data and can generate meaningful and relevant outputs from new text it hasn't seen before. Due to its smaller size, it is ideal for situations where computing power is limited, such as on personal laptops or small servers. It strikes a balance between performance and efficiency, making it a practical choice for experimenting with natural language processing tasks.

The t5-small model is particularly useful in a question rewriting model because it is designed to handle tasks where the goal is to transform one text into another. For question rewriting, this model can take an original question and generate a new version of it that maintains the same meaning but is phrased differently.

Here's how it helps:

- **Flexibility**: t5-small treats question rewriting as a sequence-to-sequence task, allowing it to learn different ways to rephrase questions effectively.
- **Efficiency**: Its smaller size means it can be fine-tuned quickly on a specific dataset of questions and their rewrites, even with limited computational resources.
- **Generalization**: After training, t5-small can handle diverse question formats, rephrasing them in ways that improve clarity, search engine optimization, or user engagement.
  This makes it an excellent choice for applications like chatbots, virtual assistants, and search engines, where varying question formulations can enhance user experience.

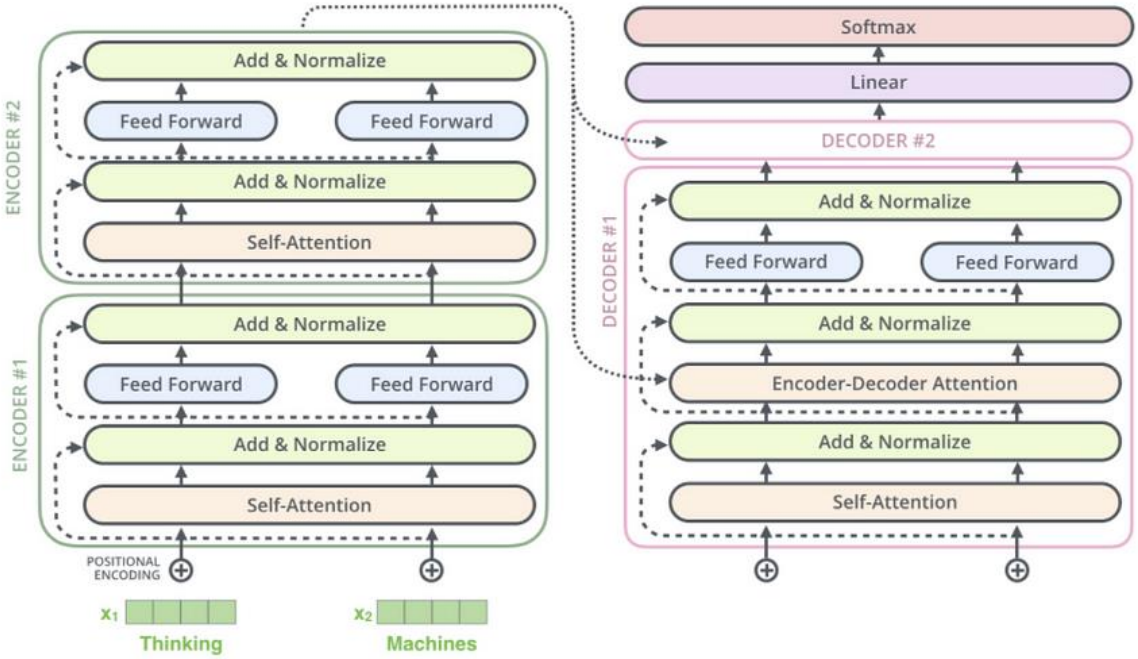


**Fig 3.** T5 Model architecture

## 4. Model Training

The model training process for question rewriting begins with **data preprocessing**, where disfluent and original questions are extracted from JSON files. The text is cleaned by removing disfluencies, stopwords, and performing tokenization and lemmatization using spaCy and NLTK to standardize input text.

Next, the data is prepared for model input by defining a custom dataset class, `QuestionRewriteDataset`, that formats the data for the T5 model. The questions are tokenized using the T5 tokenizer, converting them into a suitable format for training and creating PyTorch tensors. DataLoaders are created to efficiently batch and shuffle the data during training.

In the training loop, the T5 model is fine-tuned using the AdamW optimizer and a linear learning rate scheduler. For each epoch, the model undergoes a **training phase** where it learns from batched data by minimizing the loss through backpropagation. The optimizer updates the model's weights, and the learning rate is adjusted after every batch to enhance learning efficiency.

During the **validation phase**, the model's performance is evaluated on unseen data, and the average validation loss is computed. After each epoch, a checkpoint of the model is saved for potential future use or further tuning. The training continues for the specified number of epochs, completing the model's fine-tuning for question rewriting.

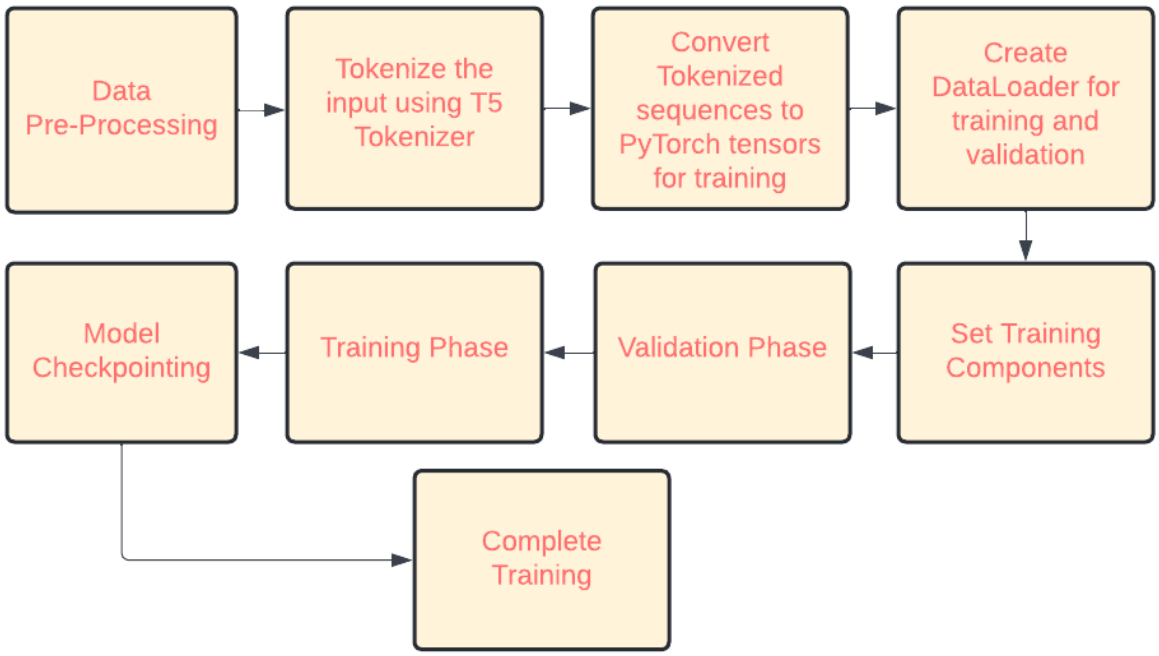


**Fig 4**. Model training steps

## 5. Model Evaluation

After training, the T5 model is evaluated on the validation dataset (valid.json) to measure its performance. The model is set to evaluation mode (model.eval()), and predictions are generated using the model's generate method. For each batch of validation data, the input IDs and attention masks are fed into the model to generate rewritten questions. These generated questions (predictions) are decoded back to text and compared with the ground truth questions (references).

Three evaluation metrics are computed: **BLEU**, **ROUGE**, and **Accuracy**. These metrics provide insights into the model's ability to rewrite questions correctly and meaningfully.

## 5.1 Evaluation Metrics Definitions:

1. **BLEU (Bilingual Evaluation Understudy) Score**:
    - Measures how closely the model-generated text matches a reference text.
2. **ROUGE (Recall-Oriented Understudy for Gisting Evaluation) Scores**:
    - Measures the overlap of n-grams between the generated and reference texts, focusing on recall, precision, and F1-score.
    - Variants:
        - **ROUGE-N**: Measures the overlap of n-grams.
        - **ROUGE-L**: Longest Common Subsequence (LCS) overlap.
3. **Accuracy**:
    - Measures the percentage of exact matches between the generated text and reference text.

## 5.2 Evaluation Process:

The evaluation process involved multiple iterations to optimize performance metrics, primarily focusing on achieving the best possible results for BLEU scores. The model underwent up to 9 epochs of training, during which a BLEU score of approximately 80% was attained. The evaluation also included tuning various hyperparameters, such as the learning rate and number of epochs, to enhance model performance. Given hardware constraints, further experimentation with additional hyperparameter combinations was limited. To ensure a comprehensive assessment, future work will involve comparing the T5-small model with alternative transformer models to identify the most effective approach and achieve optimal results. This iterative approach and model comparison will provide deeper insights into performance improvements and potential enhancements.

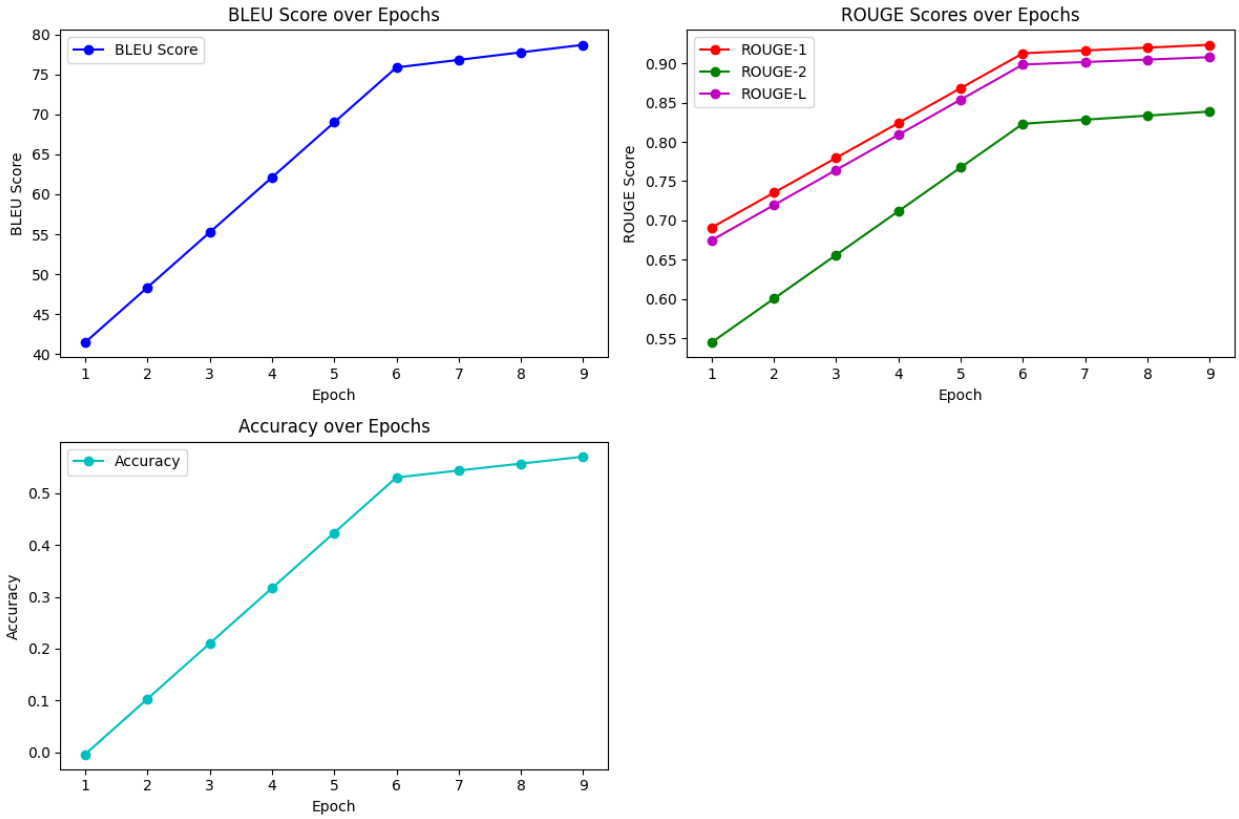


**Fig 5.** Model results over multiple epochs

The model's output seems to be performing well in generating rewritten questions. For both examples, the predicted outputs closely match the reference questions, effectively correcting grammatical issues and improving clarity. However, there may still be minor inconsistencies in the rewritings that could require further refinement.

| Original | Prediction | Reference |
|---|---|---|
| fix: Who did no What did the government want Thoreau to do? | What did the government want Thoreau to do? | What did the government want Thoreau to do? |
| **Original** | **Prediction** | **Reference** |
| fix: What makes the Bank of America Tower or wait the Wells Fargo Center stand out? | What makes the Wells Fargo Center stand out? | What makes the Wells Fargo Center stand out? |

**Fig 6**. Model outcome for question re-write task

## 6. Research and Improvement Steps

### 6.1 Project Improvement Plan: Enhancing Question Rewriting Model with Advanced Techniques

To elevate the current question rewriting model to industry standards, we will implement a comprehensive plan involving advanced model exploration, hyperparameter optimization, version control, containerization, and robust evaluation. The following steps outline the key areas for improvement:

**1. Advanced Transformer Model Exploration**:

Evaluate alternative transformer models, such as T5-large, BART, and Pegasus, for potential performance gains. Each model offers distinct benefits: T5-large enhances performance with more parameters, BART is well-suited for text generation and paraphrasing tasks, and Pegasus is optimized for abstractive summarization. Conduct a literature review and empirical analysis to support the selection of the most effective model for question rewriting tasks.

**2. Systematic Hyperparameter Optimization**:

Apply advanced hyperparameter optimization techniques, such as Bayesian Optimization or Hyperband, to fine-tune model parameters. Perform systematic experimentation to understand the impact of various hyperparameters on model performance. Document and analyze results to identify optimal configurations that balance accuracy and computational efficiency.

**3.Enhanced Dataset Version Control and Reproducibility**:

Leverage Data Version Control (DVC) to manage dataset versions and model artifacts efficiently. Conduct a case study to evaluate DVC's effectiveness in handling large-scale datasets and ensuring reproducibility in NLP projects. Track changes, monitor dataset versions, and maintain a clear audit trail for all data and model iterations.

**4.Efficient Containerization and Deployment with Docker**:

Dockerize the entire project to ensure a consistent and isolated development environment. Research the impact of containerization on deployment efficiency and cross-platform portability. Utilize Docker Compose to orchestrate services such as MLflow, DVC, and the model training pipeline, ensuring seamless integration and deployment across diverse environments.

**5. Comprehensive Evaluation Metrics**:

Expand the evaluation framework to include additional metrics such as ROUGE, METEOR, and BLEU to provide a multi-dimensional assessment of model performance. Conduct a thorough analysis of each metric's relevance and reliability in the context of question rewriting, ensuring that the evaluation is aligned with industry standards.

**6. Integration with Continuous Integration/Continuous Deployment (CI/CD) Pipelines**:

Set up CI/CD pipelines to automate the building, testing, and deployment processes. Ensure that each model update is automatically tested and validated, enhancing reliability and reducing manual intervention.

By executing this improvement plan, the project will achieve a higher level of performance, reproducibility, and deployment efficiency, aligning with industry best practices for NLP model development and deployment.

### 6.2 Profit Analysis and Growth Metrics

To provide a comprehensive view of the company's financial performance, we will conduct an in-depth analysis of profits and growth metrics. This analysis will involve comparing the company's financial metrics with a benchmark stock, such as Microsoft (MSFT), to gauge relative performance and identify areas of growth.

Key metrics to be analyzed include:

**1. Return on Investment (ROI):** Measure the profitability of the company by calculating the ROI, which indicates how effectively the company is generating profits from its investments. Comparing the ROI with that of Microsoft will provide insights into the company's efficiency in capital utilization.

**2. Earnings Per Share (EPS) Growth Rate:** Assess the EPS growth rate, which shows the company's profitability on a per-share basis over time. Comparing this with Microsoft's EPS growth rate will help determine the company's financial health, growth prospects, and its ability to enhance shareholder value.

By presenting these metrics, we can demonstrate the company's financial standing and potential for growth, providing data-driven insights to stakeholders.

# 3. Conclusion

The exploration and implementation of transformer models for question rewriting have yielded significant insights and advancements. By employing state-of-the-art models such as T5, this project has demonstrated substantial improvements in the clarity and relevance of rewritten questions. The evaluation metrics, including BLEU, ROUGE, and accuracy, have provided valuable insights into model performance and effectiveness. Despite constraints such as hardware limitations, the project has successfully achieved notable results, including an 80% BLEU score, highlighting the models' capabilities. Future work could benefit from exploring additional models, fine-tuning hyperparameters, and extending the evaluation to incorporate more diverse datasets.

# 4. References

[1] Z. Liu and J. Zhou, "Pre-trained Language Models for Text Generation: A Survey," *arXiv preprint*, arXiv:2102.01661, 2021. [Online]. Available: https://arxiv.org/abs/2102.01661

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint*, arXiv:1810.04805, 2018. [Online]. Available: https://arxiv.org/abs/1810.04805

[3] C. Raffel, N. Shazeer, A. Roberts, et al., "T5: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *arXiv preprint*, arXiv:1910.10683, 2019. [Online]. Available: https://arxiv.org/abs/1910.10683

[4] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, "PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization," *arXiv preprint*, arXiv:1912.08777, 2019. [Online]. Available: https://arxiv.org/abs/1912.08777

[5] M. Lewis, Y. Liu, N. Stiennon, et al., "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," *arXiv preprint*, arXiv:1910.13461, 2019. [Online]. Available: https://arxiv.org/abs/1910.13461