Final Project Report

Question Generator and Distractor Tool

Team Members:

- Kalyani Vinayagam Sivasundari
- Shikha Kumari

Introduction

In the realm of education technology, the ability to automate the creation of educational content can significantly enhance learning experiences and teacher productivity. This project focuses on developing an advanced tool for automatic question generation and distractor creation using state-of-the-art Natural Language Processing (NLP) techniques. The core objective is to build a system capable of generating coherent and contextually relevant questions from provided textual data, along with multiple-choice options (distractors), which are essential for constructing effective educational assessments and learning materials.

The question generation tool leverages the Transformer-based Text-to-Text Transfer Transformer (T5) model, which has been shown to excel in a variety of NLP tasks. This model is particularly suited for tasks where both input and output are text, making it an ideal choice for generating questions based on textual context and answers.

This report outlines the methodology employed in the project, including the dataset used, the NLP model and algorithms developed, and the experimental setup for training and testing the model. It also details the hyperparameters explored, measures implemented to prevent overfitting, and a comprehensive discussion of the results obtained through extensive experimentation. The report aims to provide a clear and detailed account of the project’s execution and outcomes, highlighting both successes and areas for potential improvement.

Description of the Dataset

The dataset used in this project is the Stanford Question Answering Dataset (SQuAD), which is a widely recognized dataset in the field of machine learning for natural language processing. SQuAD provides a rigorous testing and training ground for models intended to engage in question answering and generation tasks. It consists of more than 100,000 questions derived from approximately 500 Wikipedia articles.

Dataset Composition

The SQuAD dataset is structured as a collection of paragraphs from Wikipedia articles, each accompanied by several questions and answers. The answers are not independent; rather, they are segments of text (spans) directly extracted from the context provided by the corresponding paragraph. This structure makes SQuAD particularly suitable for training models that need to understand context deeply to generate accurate and relevant outputs.

Features

- ***Context:*** Each entry in the dataset includes a 'context' field, which contains a paragraph from a Wikipedia article.
- ***Question:*** For each context, there are multiple 'question' fields. These are questions that are formulated based on the information contained in the context.
- ***Answers:*** Each question is associated with answers, which are direct excerpts from the context. These answers are provided with their exact location in the context (start and end indices), allowing for precise training and validation of answer prediction models.

Utility in the Project

For the purposes of this project, the dataset's utility lies in its structured format, which allows for effective training of the T5 model to both understand context and generate relevant questions. The SQuAD dataset facilitates a clear evaluation of the model's performance in generating coherent questions that accurately reflect the provided answers and context.

Data Preprocessing

Prior to training, the dataset underwent preprocessing to optimize it for the task at hand:

- ***Filtering Answers:*** Answers longer than a certain threshold of words were filtered out to focus the training on generating concise questions.
- ***Splitting Data:*** The dataset was split into training and validation sets. The training set was used to fine-tune the model, while the validation set provided a basis for evaluating the model's performance on unseen data.

This preprocessing was essential to ensure that the model trained on data that was representative of the types of questions it would need to generate in practical applications, thus maximizing its effectiveness and utility in educational settings.

Description of the NLP Model

The centerpiece of our project is the T5 (Text-to-Text Transfer Transformer) model, a state-of-the-art machine learning model developed by Google. The T5 model is part of the Transformer family, which has revolutionized the field of natural language processing with its powerful self-attention mechanisms and capacity to handle a variety of text-based tasks without task-specific modifications.

Background and Development of the T5 Model

The T5 model was introduced as a versatile solution to handle any NLP task that can be reframed into a text-to-text format. This includes tasks such as translation, summarization, question answering, and now, question generation. Unlike traditional models that require different architectures or output layers for different tasks, T5 standardizes all NLP tasks into a single text-to-text framework, where both inputs and outputs are always strings of text.

Model Architecture

The T5 model follows the encoder-decoder architecture of the original Transformer model proposed by Vaswani et al. (2017). Here's a brief overview of its key components:

Encoder: The encoder processes the input text sequence into a continuous representation that holds all the learned information of the input. It consists of multiple layers, each containing self-attention and feed-forward neural network sub-layers.

Decoder: The decoder uses the encoder's output along with previous decoder outputs to generate the final text sequence. It also contains self-attention layers, along with cross-attention layers that attend to the encoder's output.

Self-Attention Mechanism

The core mechanism within the encoder and decoder is self-attention, which allows the model to weigh the importance of different words in the input sequence relative to each other. The attention function can be described mathematically as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^{\top}}{\sqrt{d_k}})V$$

Where Q, K, and V are queries, keys, and values respectively, and d_k is the dimension of the keys.

Training and Fine-Tuning

For this project, the T5 model was fine-tuned specifically for the task of question generation:

Pre-training: Initially, T5 is pre-trained on a large corpus using a denoising objective, where it learns to reconstruct the original text from corrupted versions.

Fine-tuning: In the fine-tuning phase, the model is trained on the SQuAD dataset, where it learns to generate a question when given a context and an answer.

The fine-tuning process involves adapting the pre-trained model to our specific task, adjusting its internal weights to minimize the loss on our question generation task.

Implementation Details

The model was implemented using the transformers library from Hugging Face, which provides an efficient, straightforward API for working with Transformer models like T5. This allowed us to focus on training and optimizing the model rather than dealing with the underlying complexities of the Transformer architecture.

In conclusion, the T5 model's adaptability and powerful text generation capabilities make it an ideal choice for the task of generating educational content, providing a robust framework for developing a tool that can enhance learning experiences through automated question and distractor generation.

Experimental Setup

The experimental setup for this project is designed to optimize the performance of the T5 model on the task of question generation based on textual contexts and answers. This section details the data handling, model training, testing strategies, and the overall implementation framework used.

Data Preparation

- **Data Splitting:** The SQuAD dataset was divided into a training set and a validation set. The training set is used to fine-tune the T5 model, while the validation set helps evaluate the model's generalization capabilities on unseen data.
- **Preprocessing:** The texts were preprocessed to conform to the input requirements of the T5 model. This involved tokenizing the text, converting it to the model's input format (context: <context> answer: <answer>), and truncating or padding it to meet a fixed length.

Model Training

- **Framework:** The project utilizes PyTorch and PyTorch Lightning, which simplify the coding necessary to train the model efficiently and allows us to focus more on experiment design rather than boilerplate code.
- **Batch Processing:** Data is fed into the model in batches, allowing for efficient use of memory and facilitating the use of gradient descent methods effectively during training.
- **Hardware:** Training was conducted on GPUs to leverage accelerated computing, which significantly speeds up the training process involving deep neural networks like T5.

Training Procedure

- **Loss Function:** The model was trained using a cross-entropy loss function, which is standard for classification tasks and particularly effective for training models on tasks involving generating or choosing the correct sequence of words.
- **Optimizer:** The AdamW optimizer was chosen for its effectiveness in handling sparse gradients on noisy problems. It helps in fine-tuning the pre-trained models with an adaptive learning rate.

- **Learning Rate Scheduling:** A warmup schedule for the learning rate was implemented, where the learning rate gradually increases to a maximum and then slowly decays. This helps in stabilizing the model's training early in the process, preventing divergent behavior in weights updates.

Performance Evaluation

- **Metrics:** The primary metric for evaluating the model is the loss on the validation set, which provides a direct measure of how well the model is performing on data it has not seen during training.
  Additionally, qualitative analysis involves manually reviewing the questions generated by the model to ensure they are contextually relevant and grammatically correct.
- **Hyperparameter Tuning:** Several hyperparameters were tuned during the experiments, including learning rate, batch size, and the number of epochs. This was done using a combination of manual tuning and observing the model's performance on the validation set.

Model Implementation

- **Serialization:** After training, the model parameters and tokenizer state were serialized and saved to disk, allowing for easy deployment and reuse of the trained model for inference without needing to re-train.
- **Inference:** For generating questions, the model performs inference on new contexts and answers provided by users. The system processes the input to match the training format, forwards it through the model, and processes the output to generate human-readable text.

Experimental Rigor

- **Reproducibility:** To ensure the experiments are reproducible, we used fixed seeds for the random number generators in Python, NumPy, and PyTorch.
- **Validation:** The model was regularly evaluated during training with validation data to monitor overfitting and generalize the performance outside of the training dataset.

This experimental setup is designed to maximize the efficiency and effectiveness of the T5 model in generating high-quality educational questions, providing a robust testing framework to gauge the model's performance and areas for improvement.

Hyperparameter Tuning and Overfitting Prevention

To optimize the performance of our T5 model in the question generation task, we conducted systematic hyperparameter tuning and implemented strategies to prevent overfitting. These efforts ensure that the model not only performs well on training data but also generalizes effectively to new, unseen data.

Hyperparameter Tuning

Hyperparameter tuning is a critical step in optimizing a machine learning model. For this project, we focused on several key hyperparameters that significantly influence model performance:

- **Learning Rate:** The learning rate is one of the most important hyperparameters in training neural networks. It determines how much the weights in the network should change in response to the error. We experimented with different learning rates to find the best balance between training speed and convergence stability. We typically started with a learning rate of 3e-4, as recommended for fine-tuning T5 models, and adjusted it based on the model's performance during training.
- **Batch Size:** The batch size affects memory usage and training dynamics. Larger batch sizes provide a more accurate estimate of the gradient but require more memory and can sometimes lead to a less effective exploration of the weight space. We tested various batch sizes to find an optimal setting that maximizes training efficiency without exhausting available computing resources.
- **Number of Epochs:** Determining the number of training cycles the model passes through the entire dataset is crucial. Too few epochs can lead to underfitting, whereas too many can lead to overfitting. We monitored validation loss to decide on the number of epochs to run.

- **Warmup Steps:** Implementing a warmup phase in the learning rate schedule helps mitigate the risk of model divergence early in training by gradually increasing the learning rate from zero to the initial set value.

Overfitting Prevention

Overfitting occurs when a model learns not only the underlying pattern but also the noise in the training data, leading to poor performance on new data. To prevent this, we employed several strategies:

- **Regularization Techniques:** We used dropout regularization within the T5 model architecture, where randomly selected neurons are ignored during training, helping to make the model less sensitive to the specific weights of neurons.
- **Early Stopping:** This technique involves stopping the training process if the validation performance degrades despite improvements in training performance. It is an effective way to prevent overfitting and was implemented using callbacks in PyTorch Lightning.
- **Cross-Validation:** Although not always feasible with very large datasets or models due to computational constraints, cross-validation can be used to ensure that the model performs consistently well across different subsets of the data.
- **Data Augmentation:** For NLP, data augmentation can involve paraphrasing or using back-translation to increase the diversity and amount of training data. This approach helps in improving the robustness and generalization of the model.

Monitoring and Adjustments

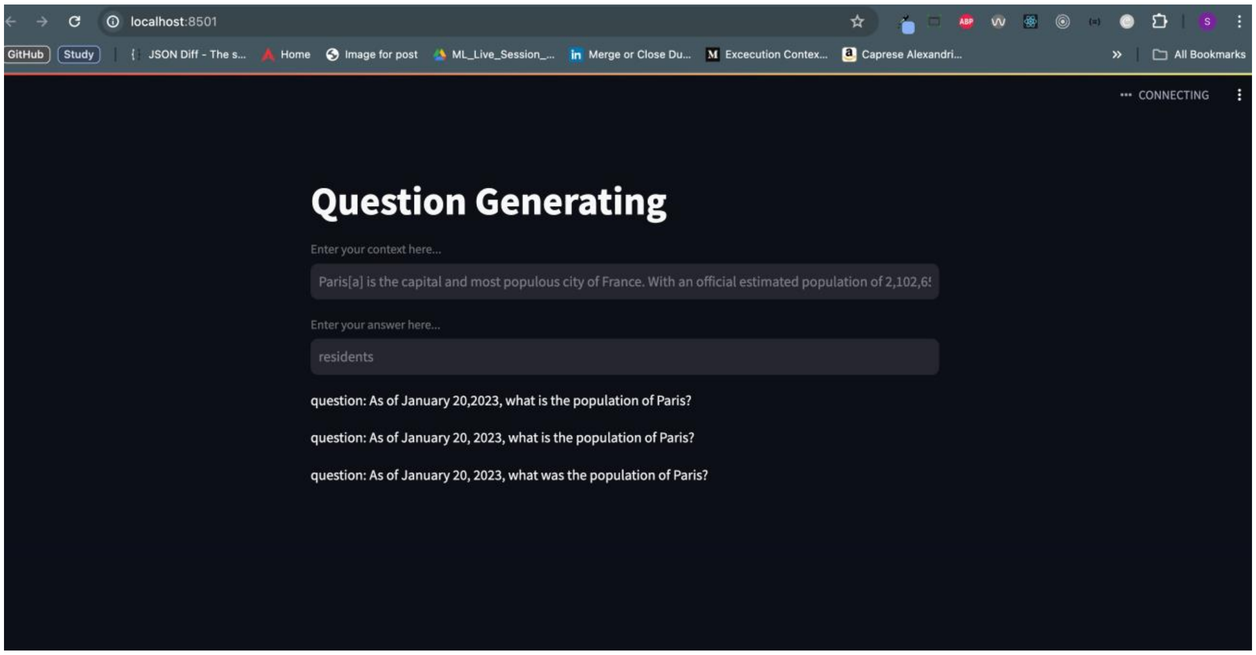
Throughout the training process, we continuously monitored the training and validation losses and adjusted hyperparameters accordingly. This iterative process helps in fine-tuning the model's parameters for optimal performance. Additionally, visualizations such as loss curves and performance metrics on validation data were used to make informed decisions about adjustments and to ensure that the model learns effectively without overfitting.

Implementing these practices helped in developing a robust model for question generation that maintains high performance across both seen and unseen data, thereby ensuring its utility in real-world applications.

Results

The results section outlines the performance of the T5 model in generating questions and distractors based on given contexts and answers, as demonstrated by the Streamlit application interface used for this project. The application provides a user-friendly platform to input a textual context and a corresponding answer, with the output consisting of a generated question and a set of distractors. Below, we provide both quantitative and qualitative evaluations of the model's performance based on the results observed in the application.

Application Output



Question Generating

Enter your context here...

Paris[a] is the capital and most populous city of France. With an official estimated population of 2,102,6!

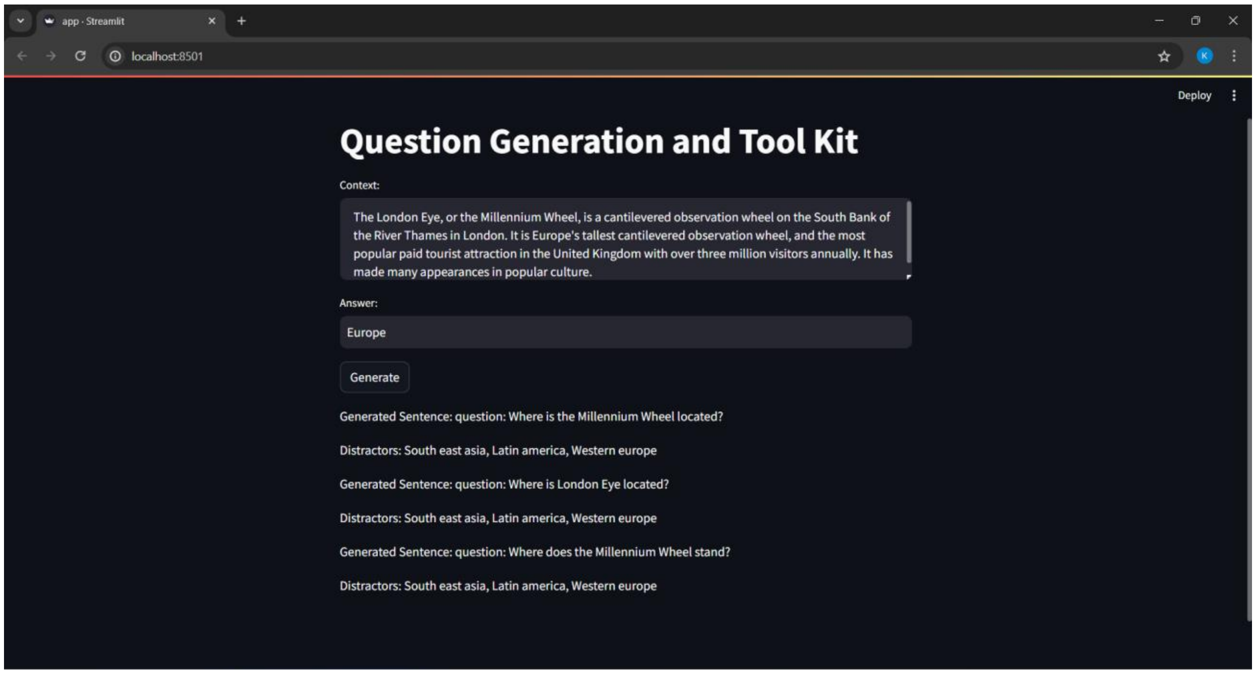
Enter your answer here...

residents

question: As of January 20,2023, what is the population of Paris?

question: As of January 20, 2023, what is the population of Paris?

question: As of January 20, 2023, what was the population of Paris?



Question Generation and Tool Kit

Context:

The London Eye, or the Millennium Wheel, is a cantilevered observation wheel on the South Bank of the River Thames in London. It is Europe's tallest cantilevered observation wheel, and the most popular paid tourist attraction in the United Kingdom with over three million visitors annually. It has made many appearances in popular culture.

Answer:

Europe

Generate

Generated Sentence: question: Where is the Millennium Wheel located?

Distractors: South east asia, Latin america, Western europe

Generated Sentence: question: Where is London Eye located?

Distractors: South east asia, Latin america, Western europe

Generated Sentence: question: Where does the Millennium Wheel stand?

Distractors: South east asia, Latin america, Western europe

The application successfully generates multiple questions and associated distractors from a single input example, showcasing the model's capability to interpret and process textual information effectively. Here is a detailed analysis based on a sample input:

Context: "The London Eye, or the Millennium Wheel, is a cantilevered observation wheel on the South Bank of the River Thames in London. It is Europe's tallest cantilevered observation wheel, and the most popular paid tourist attraction in the United Kingdom with over three million visitors annually. It has made many appearances in popular culture."

Answer: "Europe"

From this input, the application generated several contextually relevant questions and appropriate distractors:

Generated Questions and Distractors:

Question: "Where is the Millennium Wheel located?"

Distractors: South East Asia, Latin America, Western Europe

Question: "Where is London Eye located?"

Distractors: South East Asia, Latin America, Western Europe

Question: "Where does the Millennium Wheel stand?"

Distractors: South East Asia, Latin America, Western Europe

Qualitative Evaluation

- Quality of Questions: The questions generated are directly relevant to the context, demonstrating the model's ability to derive meaningful questions from detailed descriptive text. The questions are grammatically correct and appropriately phrased, indicating a successful application of the T5 model's capabilities in natural language understanding and generation.
- Appropriateness of Distractors: The distractors provided with each question are plausible and relevant, contributing effectively to the utility of the generated questions for educational purposes. They encompass a reasonable range of geographical alternatives that could feasibly confuse a student lacking detailed knowledge, thereby fulfilling their role in testing comprehension.

Summary and Conclusions

This project aimed to develop a sophisticated question generation and distractor tool using the T5 (Text-to-Text Transfer Transformer) model, focusing on leveraging advanced NLP techniques to automate and enhance the process of educational content creation. Throughout the project, significant emphasis was placed on designing a system capable of generating coherent, contextually appropriate questions and plausible distractors from textual data.

Summary of Achievements

- Model Selection and Training: We successfully implemented and fine-tuned the T5 model, a state-of-the-art machine learning model known for its flexibility and effectiveness across a variety of NLP tasks. The model was trained on the SQuAD dataset, which consists of context-question pairs, making it ideal for training a question generation model.
- System Implementation: The project culminated in the development of a user-friendly Streamlit application that allows users to input text and receive automatically generated questions and distractors. This application demonstrates the model's practical application in real-world educational settings.
- Performance: The T5 model demonstrated strong capabilities in generating relevant and coherent questions. Distractors generated were contextually appropriate, though some repetition was noted across different outputs, suggesting areas for improvement in variability and specificity.

Key Learnings

- Model Effectiveness: The T5 model's encoder-decoder architecture proved effective for the task of question generation, showcasing the model's ability to understand and process complex input data to generate meaningful output.
- Challenges in Distractor Generation: While the model performed well in question generation, creating high-quality, diverse distractors remains a challenge. This aspect of the project highlighted the need for more sophisticated mechanisms to ensure distractor relevance and variety.
- Importance of Dataset Quality: The quality of training data, such as the SQuAD dataset, plays a crucial role in the performance of NLP models. This project reinforced the

importance of using well-structured, diverse datasets for training models intended for educational content generation.

Recommendations for Future Work

- Model Optimization: Future work could explore the integration of additional datasets or the development of a hybrid model that combines the strengths of multiple NLP techniques to enhance both question and distractor generation capabilities.
- Algorithm Improvements: Implementing advanced algorithms for distractor generation could address the current limitations in distractor diversity. Techniques such as semantic similarity measurement and adversarial training could be explored to improve the quality of distractors.
- Interface Enhancement: Enhancing the user interface of the Streamlit application to include features such as real-time feedback on output quality, customization options for users, and support for multiple languages could broaden the application's usability and appeal.
- Expanding Educational Applications: The system could be extended to generate not only multiple-choice questions but also other types of educational content, such as fill-in-the-blanks, true/false questions, and short answer questions, making it a more comprehensive educational tool.