Final Projects

Project 1: AI Teaching Assistant

Project Overview

The goal of this project is to create an AI Teaching Assistant (AI TA). The AI TA aims to provide an interactive platform for students to ask questions and receive guidance on course materials, mirroring the support traditionally offered by human teaching assistants. The project emphasizes the application of NLP techniques to create an educational tool that enhances the learning experience through dynamic interaction and content understanding.

Project Objectives

- **Automate FAQ Responses:** Develop the AI TA to autonomously address frequently asked questions related to course logistics, assignments, and content.
- **Content Understanding and Query Handling:** Implement NLP strategies enabling the AI TA to interpret and respond to inquiries about the uploaded course material accurately.
- **Dynamic Interaction:** Design the AI TA to interact with students conversationally, providing information and resources pertinent to their queries.

Technical Approach

NLP Engine Components

- **Language Model Integration:** Leverage pre-trained NLP models (e.g., BERT, Llama) for robust text understanding and generation, tapping into their vast linguistic knowledge.
- **Natural Language Understanding (NLU):** Employ models like BERT for NLU tasks to accurately parse student inquiries, determining their intent and extracting relevant details for precise responses.
- **Retrieval-Augmented Generation (RAG):** Central to the NLP Engine, RAG combines retrieval methods (to access information within a structured database of course materials) with generative capabilities (to formulate coherent and context-aware responses). This approach ensures the AI TA delivers informative answers, directly pulling from course content and synthesizing it into helpful responses.
- **Response Generation:** Utilize the designed AI TA to generate responses that are not only accurate but also engaging and pedagogically effective.

Interaction Interface

- **Simplified Interface Development:** Construct an intuitive web or command-line interface that allows straightforward question submission and response retrieval, ensuring ease of use for student interactions with the AI TA.

Project 2: TV Show Script Generation

Project Overview

For this project, you will harness the power of Natural Language Processing (NLP) and Machine Learning to create a script for a new episode of a TV show you enjoy. The challenge is to design a model that captures the essence, style, dialogue, and storyline of the original series, thereby generating a script that feels authentic and could seamlessly fit into the show.

Project Objectives

- **Data Collection:** Gather scripts from existing episodes of the chosen TV show to serve as training data for the model.
- **Text Processing:** Implement text preprocessing techniques to prepare the data for model training.
- **Model Development:** Use a suitable NLP model (e.g., GPT-2 or Llama) to learn the style and narrative structure of the TV show.
- **Script Generation:** Generate a script for a new episode that maintains the characters' distinct voices and the show's thematic elements.

Technical Approach

Data Preparation

- **Script Scraping and Compilation:** Collect scripts from available episodes, focusing on extracting dialogue, stage directions, and character actions.
- **Text Cleaning:** Standardize the format, remove extraneous metadata, and ensure consistency across the dataset.

Model Training

- **Choice of Model:** Depending on the complexity of the show and the available computational resources, choose an appropriate pre-trained model like GPT-2 or Llama for fine-tuning.
- **Fine-Tuning Parameters:** Adjust training parameters to balance between learning the specific style of the show and retaining the model's general language capabilities.

Project 3: Text Style Transfer

Project Overview

The Text Style Transfer project aims to explore and implement techniques in Natural Language Processing (NLP) that allow for the transformation of text from one style to another while retaining the original content's meaning. Examples of style transfer include changing the sentiment of a sentence, transforming modern English text into Shakespearean English, or converting formal text into informal text and vice versa.

Project Objectives

- **Understanding Text Styles:** Analyze and understand the characteristics that define different text styles, including linguistic features, tone, and stylistic elements.
- **Data Collection and Preparation:** Gather and preprocess datasets suitable for training models capable of understanding and generating text in the target styles.
- **Model Development and Training:** Employ state-of-the-art NLP models for the task of style transfer, focusing on models that are capable of separating content from style.
- **Evaluation and Refinement:** Use appropriate metrics for evaluating the effectiveness of the style transfer and refine the model based on feedback and performance.

Technical Approach

Data Preparation

- **Dataset Identification:** Identify and compile datasets that exemplify the source and target styles. This may include collecting text from different domains or authors known for their distinctive styles.
- **Preprocessing:** Clean and preprocess the text to facilitate model training, including tokenization, normalization, and the creation of parallel corpora if available.

Model Selection and Training

- **Model Architecture:** Explore and select appropriate model architectures for style transfer, such as Seq2Seq models with attention mechanisms, variational autoencoders (VAEs), or transformer-based models like Llama.
- **Style Disentanglement:** Implement techniques to disentangle content and style in text, enabling the model to retain content while altering style.

Evaluation

- **Quality of Transfer:** Assess the quality of the style transfer, ensuring the transformed text accurately reflects the target style without losing the original content's meaning.
- **Coherence and Readability:** Evaluate the coherence and readability of the generated text, ensuring the style transfer does not result in nonsensical or disjointed text.

Project 4: Knowledge Distillation for NLP Tasks with Teacher-Student Models

Project Overview

In this project, students will engage in the development of a knowledge distillation framework to compress large, complex NLP models into smaller, efficient counterparts without significant loss in performance. The focus will be on replicating the hidden state outputs of a teacher model within a student model for specific NLP tasks such as sentiment analysis, named entity recognition (NER), or machine translation. This project provides an opportunity to explore the efficiency and effectiveness of neural network models in processing natural language.

Project Objectives

- **Explore Knowledge Distillation:** Gain a deep understanding of knowledge distillation techniques and their role in reducing model size and computational demands.
- **Implement Teacher-Student Model Architecture:** Develop and train a smaller student model to mimic the hidden state outputs of a larger teacher model.
- **Task-Specific Learning:** Apply the knowledge distillation process to specific NLP tasks, utilizing relevant datasets to train and evaluate the models.
- **Model Evaluation:** Assess the distilled student model's performance in terms of accuracy, efficiency, and its ability to replicate the teacher model's behavior.

Technical Approach

Model Architecture

- **Teacher Models:** Select from pre-trained models like BERT (for tasks like sentiment analysis and NER) or Transformer models like OpenAI's GPT-2 (for generative tasks like machine translation). These models are known for their high performance but are computationally intensive.
- **Student Models:** Design smaller models, such as distilled versions of BERT (DistilBERT), TinyBERT, or custom lightweight transformer models, aiming for a balance between performance and efficiency.

Training Strategy

- **Distillation Objective:** Design a loss function that effectively combines the traditional task-specific loss and a distillation loss that measures the discrepancy between the teacher's and student's hidden states.
- **Dataset Selection:** Utilize task-specific datasets like the Stanford Sentiment Treebank (SST) for sentiment analysis, CoNLL-2003 for NER, and WMT datasets for machine translation.
- **Training and Distillation:** Fine-tune the student model on the selected dataset while also learning from the teacher model's hidden state outputs to achieve similar performance.

Evaluation Metrics

- **Performance Metrics:** Use accuracy for classification tasks (sentiment analysis, NER) and BLEU scores for generative tasks (machine translation) to measure the student model's effectiveness.
- **Efficiency Metrics:** Evaluate model size reduction and inference speed improvements in the student model compared to the teacher model.
- **Fidelity Metrics:** Measure the similarity between the student and teacher model outputs using techniques such as cosine similarity for hidden states.

General Evaluation Criteria and Submission Requirements

Evaluation Criteria

The evaluation of the projects will be based on the following criteria:

1. **Accuracy and Performance:** The model's performance on the task, measured against a predefined metric (e.g., BLEU for generation, F1 for classification tasks, perplexity for language models).

2. **Quality of Code:** Clarity, organization, and documentation of the code. The code should be well-structured, readable, and accompanied by comments explaining the functionality.

3. **Data Preprocessing and Handling:** The effectiveness of data cleaning, preprocessing, and handling techniques. This includes dealing with missing values, data augmentation, and the use of effective data loaders.

4. **Model Optimization and Training:** The effectiveness of model optimization techniques, including hyperparameter tuning, regularization methods, and training procedures.

5. **Analysis and Interpretation:** Depth of analysis on the model's performance, including error analysis, limitations of the model, and interpretation of the results. Insightful observations and potential improvements should be discussed.

6. **Project Report:** The quality of the project report, including clarity of writing, organization, and thoroughness in covering the project's scope, methodology, results, and conclusions. The report should also include a clear abstract and introduction to the problem.

7. **Innovation and Creativity:** The uniqueness of the approach and creativity in solving the problem. Innovative use of NLP techniques or development of new methods will be highly regarded.

Submission Requirements

1. **Code:** Submit all code files used for the project in a single compressed folder (.zip or .tar.gz). Include a README file with instructions on how to run the code and any dependencies required.

2. **Data:** If the dataset used is publicly available, provide the link to the dataset. If the dataset is created or modified for the project, include it in the submission with a description in the README file.

3. **Report:** A project report in PDF format, not exceeding 10 pages (excluding appendices and references), formatted in a clear and readable style. The report should include sections on introduction, methodology, results, analysis, and conclusions. Include any relevant figures, tables, and references.

4. **Presentation:** Prepare a short presentation (10-15 slides) summarizing the project's objective, methodology, key results, and conclusions.

5. **Additional Materials:** Any additional materials used or created for the project, such as scripts for data preprocessing, trained models, or Jupyter notebooks, should be included in the submission.

Submissions failing to meet these requirements may be penalized. Please ensure that all submitted materials are well-organized and clearly labeled for easy navigation.