Report on Approach and Model Evaluation

1. Problem Statement

My task was to build a category prediction model for news articles based on their summaries.

Approach

Data Pre processing

- **Data Reading and Cleaning:** The initial step is reading the data set using Pandas. Duplicate rows and missing values are removed to ensure data quality.

- **Data Sampling:** To balance the dataset, a portion of the data is sampled for each unique category. In my code, I chose to sample 25% of the data for each category, ensuring that no single category dominates the dataset.

- **Label Encoding**: The categorical labels in the 'Category' column are transformed into numerical values using LabelEncoder from Scikit-Learn. This step is essential because machine learning models require numerical inputs.

Model Building

- **Tokenization and Input Encoding:** I used the BERT tokenizer (BertTokenizer) to preprocess the news article summaries. These tokenized summaries are then converted into input tensors with necessary padding and truncation.

- **Model Selection**: I chose the BERT-based model for sequence classification, specifically BertForSequenceClassification. This model is pretrained on a large corpus of text and fine-tuned for my specific classification task.

- **Training Loop:** The model is trained using a custom training loop. I iterate through the data in mini-batches, calculate the loss, and perform backpropagation to update the model's parameters. The AdamW optimizer is used with a learning rate of 2e-5, and the cross-entropy loss is chosen as the loss function.

Model Evaluation

- **Evaluation Function**: An evaluation function is defined to assess the model's performance. In this function, the model's predictions are compared to the true labels to calculate accuracy.

- **Loading Pretrained Model:** I also provided the functionality to load a pretrained model from a file.

- **Prediction:** For each news summary in the dataset, the model is used to predict the category. The predictions are then compared to the true labels to calculate accuracy. The calculated accuracy score is logged with a timestamp for reference.

Deployment

- **Flask API:** I deployed the model as a Flask API. The API accepts a JSON request containing a news summary and returns the predicted category.

Metrics for Model Evaluation

I chose to use accuracy as the primary metric to measure the performance of my model. Accuracy is a suitable metric when the classes are roughly balanced, as is the case after your dataset sampling step. It measures the ratio of correctly predicted categories to the total number of predictions, providing a good overall view of model performance.

2. Problem Statement

My task is to build a model for generating headlines given a news description or summary.

Approach

Data Preprocessing

- **Data Reading and Cleaning:** I start by reading the dataset using Pandas and remove any duplicate rows or rows with missing values to ensure data quality.

- **Data Sampling:** Similar to my previous task, I sample a portion of the data for each unique category to balance the dataset. This step ensures that the model learns from a diverse set of categories.

- **Tokenization and Preprocessing:** I use the GPT-2 tokenizer (GPT2Tokenizer) to tokenize and preprocess the news article summaries and headlines. These tokenized sequences are then padded to a maximum length to create input sequences.

Model Building

- **Model Selection:** I use the GPT-2 model (GPT2LMHeadModel) as the base model for headline generation. This model is pretrained on a large corpus of text and fine-tuned for my specific task.

- **Custom Headline Generator Model:** I define a custom headline generation model that includes the base GPT-2 model and an additional linear layer (nn.Linear) for generating headlines. This custom model is trained for headline generation.

- **Training Loop:** The model is trained using a custom training loop. I use the AdamW optimizer with a learning rate of 1e-4 and a learning rate scheduler (get_linear_schedule_with_warmup) to adjust the learning rate during training.

- **Loading Pretrained Model:** I provided the ability to load a pretrained model checkpoint for headline generation.

Model Evaluation

Cosine Similarity Score: To evaluate the generated headlines, I calculate the cosine similarity between the reference headlines and the headlines generated by the model. This metric measures the similarity between two text vectors and can give me an idea of how close the generated headlines are to the reference headlines.

Logging and Reporting: I log the cosine similarity scores along with time stamps to keep a record of the model's performance.

Deployment

Flask API: I deploy the model as a Flask API. The API accepts a JSON request containing a news summary and returns the generated headline.

Metrics for Model Evaluation

I chose to use cosine similarity scores as the primary metric for evaluating the performance of my headline generation model.
Cosine similarity measures the similarity between two text vectors and can provide a sense of how well the generated headlines match the reference headlines. A higher cosine similarity indicates greater similarity between the generated and reference headlines.

Report Prepared by:-
Swetanshu Pandey
Contact No.:- 7007575977