Summarization Model with GPT-2

June 11, 2024

Contents

Brief introduction to the project

- **Purpose**: To compare the performance of a fine-tuned GPT-2 model with baseline methods for text summarization.
- **Motivation**:
  - Understanding the implementation of a Large Language Model (LLM) from scratch.
  - Exploring the effectiveness of GPT-2 in generating text summaries.
- **Methods**: Implementation of summarization, evaluation, and comparison using multiple metrics.

1 Related work

We found that the work of Radford et al. (2019) on GPT-2 was a significant inspiration for our project. Their research demonstrated the potential of large-scale language models in generating coherent text. Additionally, the work of Liu et al. (2019) on Text Summarization provided valuable insights into the summarization process. We built upon these studies to develop our project.

- **GPT-2**: A large-scale unsupervised language model that can generate coherent text.
- **Text Summarization**: The process of distilling the most important information from a source text.
- **Baseline Methods**: Simple summarization techniques like truncation and keyword extraction.

2 dataset

The dataset used in this project consists of news articles from various sources. Each article is paired with a human-generated summary. The

dataset is preprocessed to remove irrelevant information and ensure the quality of the text data.

- **Source**: Multiple news websites and blogs.
- **Format**: JSON files containing articles and summaries.
- **Preprocessing**: Cleaning, tokenization, and normalization of text data.
- **Size**: 10,000 articles with corresponding summaries.

3 Overwiew of structure

We implemented two baseline methods for text summarization: truncation and keyword extraction. These methods serve as simple benchmarks for evaluating the performance of the GPT-2 model.

- **Truncation**: Selecting the first few words of the article as the summary.
- **Keyword Extraction**: Identifying the most important words in the article as the summary.
- **Evaluation**: Comparing the baseline summaries with human-generated summaries.
- **Metrics**: ROUGE scores, BLEU scores, and semantic similarity.

Project Structure Overview

- **Overview**: This project aims to compare the performance of a fine-tuned GPT-2 model with baseline methods for text summarization.
- **Components**:
  - final.ipynb: Main notebook for running the training, testing, and comparison scripts.

- baseline_methods.py: Implements baseline methods for text summarization.
- preprocess.py: Handles text preprocessing tasks.
- utils.py: Utility functions for model saving and loading.
- unit_test.py: Contains unit tests for the summarization functions.
- training.py: Script for fine-tuning the GPT-2 model.
- comparison.py: Script for comparing the GPT-2 model summaries with baseline summaries.

3.1 final.ipynb

The final.ipynb notebook is the orchestrator for the entire project. It installs necessary packages, runs the training script, executes unit tests, and performs comparisons between the model and baseline methods. Below is a detailed explanation of each section in the notebook:

3.2 1. Installing Required Packages

The first step is to ensure all required packages are installed. This includes packages for natural language processing, machine learning, and evaluation metrics.

3.3 2. Running the Training Script

The next step is to run the training.py script to fine-tune the GPT-2 model on the provided dataset.

3.4 3. Running Unit Tests

After training the model, it's important to verify its functionality using unit tests.

3.5 4. Performing Comparisons

Finally, the notebook runs the comparison.py script to compare the summaries generated by the GPT-2 model against baseline methods.

3.6 baseline_methods.py

The baseline_methods.py script implements simple baseline methods for text summarization. These methods serve as a point of comparison for evaluating the performance of the fine-tuned GPT-2 model. Below is a detailed explanation of the functions and their purposes:

3.7 1. Importing Libraries

The script starts by importing necessary libraries for data manipulation and text processing.

3.8 2. Summary Truncation Function

This function creates a truncated version of the summary by selecting the first few words.

```
summary_truncation: Takes an input text and returns the
first num_words words. This simple method provides a naive
baseline for summarization by truncating the text.
```

3.9 3. Extract Keywords Function

```
extract_keywords: Uses TF-IDF to identify the most important
words in the text. The TfidfVectorizer transforms the text
into a matrix of TF-IDF features, and the function selects
the top num_keywords words. This method provides another
simple baseline by summarizing the text based on its key terms.
```

3.10 4. Process Articles Function

```
process_articles:
- Samples num articles from the provided dataset.
- For each article, it generates a truncated summary and a
  keyword-based summary.
- Appends the original article, original summary, truncated
  summary, and keyword summary to the results list.
- If show is True, prints out the details of each processed
  article.
```

3.11 preprocess.py

The preprocess.py script is based on HW2's preprocess file and is designed to clean and prepare text data for further processing, such as training a machine learning model. This script primarily focuses on text normalization, stop word removal, and lemmatization. Below is a detailed explanation of the functions and their purposes:

3.12 1. Importing Libraries

The script begins by importing necessary libraries for text processing.

3.13 2. Remove Stopwords Function

```
remove_stopwords:
- Takes a string text as input.
- Tokenizes the text into individual words.
- Removes any word that is present in the NLTK stop words list.
- Returns the cleaned text as a single string.
```

3.14 3. Preprocessing Function

This function performs several text preprocessing steps including lowercasing, removing special characters, stop word removal, and lemmatization.

```
preprocessing_function:
- Converts the text to lowercase.
- Removes HTML tags (e.g., <br />).
- Uses a regular expression to remove any character that is not
  a letter or whitespace.
- Calls remove_stopwords to eliminate common stop words.
- Lemmatizes the remaining words to their base forms using the
  WordNetLemmatizer.
- Returns the fully processed text as a single string.
```

3.15 utils.py

The utils.py script is designed to provide utility functions for saving and loading machine learning models. These functions are essential for managing the trained models, enabling you to save a model's state after training and reload it for inference or further training. Below is a detailed explanation of the functions within this script:

3.16 1. Importing Libraries

The script begins by importing the necessary library for handling the machine learning model.

```python
import torch
from torch.utils.data import DataLoader, Dataset
from transformers import GPT2Tokenizer, GPT2LMHeadModel
from torch.nn.utils.rnn import pad_sequence
import pandas as pd
from preprocess import preprocessing_function
from utils import save_model
from torch.nn import CrossEntropyLoss
```

Figure 1: imported libraries

3.17 2. Save Model Function

This function saves the state of a given model to a specified file path.

```
save_model:
- Parameters:
  - model: The PyTorch model instance to be saved.
  - path: The file path where the model state dictionary will
    be saved.
- Functionality:
  - Uses torch.save to serialize and save the model's state
    dictionary (i.e., the model's parameters) to the specified
    file path.
- Output:
  - Prints a message indicating the model has been successfully
    saved.
```

3.18 3. Load Model Function

This function loads the state of a given model from a specified file path.

```
load_model:
- Parameters:
```

```
  - model: The PyTorch model instance to which the state
    dictionary will be loaded.
  - path: The file path from where the model state dictionary
    will be loaded.
  - device: The device (CPU or GPU) on which the model will be
    loaded.
- Functionality:
  - Uses torch.load to deserialize and load the model's state
    dictionary from the specified file path.
  - Maps the loaded model to the specified device using
    map_location.
  - Moves the model to the specified device using
    model.to(device).
- Output:
  - Prints a message indicating the model has been successfully
    loaded.
  - Returns the loaded model instance.
```

3.19 unit_test.py

The unit_test.py script is designed to perform unit tests on the summarization model. Unit testing is essential to ensure that individual components of the model function correctly. This script specifically tests the summarization capability of the model by generating a summary for a sample article and comparing it with an expected summary. Below is a detailed explanation of the functions and the main execution flow within this script:

3.20 1. Importing Libraries

The script starts by importing the necessary libraries for tokenization, model handling, and preprocessing.

3.21 2. Generate Summary Function

```
generate_summary:
- Parameters:
```

- `article: The input article text to be summarized.`
- `tokenizer: The tokenizer for encoding the text.`
- `model: The summarization model for generating summaries.`

4 Evaluation

The evaluation of the summarization model is crucial to understand its performance and identify areas for improvement. We use several metrics to evaluate the model's performance, including ROUGE scores, BLEU scores, and semantic similarity. These metrics provide insights into the quality of the generated summaries and help compare them with human-generated summaries.

5 result

```
--------------------------------------------------------------------------------
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Article 30: Angel di Maria's old mansion is up for sale - but Manchester United fans need
Baseline Summary: Denise 'Wewe' Ross, 43, was
AI Summary: angel di maria old mansion sale manchester united fan need worry mean way old
Di Maria moved to Manchester United from Real Madrid at the end of last season. Di Maria'
The house is set to be sold for £1.5million after break-in in 2011.
Similarity Score: 0.009673518742442563
Baseline Similarity Score: 0.009472259810554804
BLEU Score: 0.040195107858683216
Baseline BLEU Score: 1.9292759057311917e-70
ROUGE Score: 0.5999999950873887
Baseline ROUGE Score: 0.005899704724114842
--------------------------------------------------------------------------------
AI generated better summaries in 80.0% of cases (Similarity).
Baseline generated better summaries in 20.0% of cases (Similarity).
AI generated better summaries in 100.0% of cases (BLEU).
Baseline generated better summaries in 0.0% of cases (BLEU).
AI generated better summaries in 100.0% of cases (ROUGE).
Baseline generated better summaries in 0.0% of cases (ROUGE).
Average AI Summary Similarity Score: 0.053642652405670475
Average Baseline Summary Similarity Score: 0.013649467662136046
Average AI Summary BLEU Score: 0.007664525231071647
Average Baseline Summary BLEU Score: 1.281723193072564e-25
Average AI Summary ROUGE Score: 0.25141013292827924
Average Baseline Summary ROUGE Score: 0.007853243926822132
```

Figure 2: Comparison of GPT-2 model with baseline methods

It shows that the GPT-2 model outperforms the baseline methods in terms of ROUGE scores, BLEU scores, and semantic similarity. The GPT-2

model generates summaries that are more similar to human-generated summaries, indicating its effectiveness in capturing the key information from the source text.

6 error analysis

The error analysis of the GPT-2 model reveals several areas for improvement:

- **Repetition**: The model tends to repeat phrases or sentences in the generated summaries.
- **Informativeness**: Some summaries lack important details present in the source text.
- **Coherence**: The flow of the generated summaries can be improved for better readability.
- **Length**: The model sometimes generates summaries that are too short or too long compared to the source text.

7 future work

we may improve our baseline methods by incorporating more advanced techniques such as extractive summarization algorithms or neural network-based models. Additionally, we could explore other large language models like GPT-3 or BERT for text summarization tasks. Finally, we could investigate domain-specific summarization models that are tailored to specific types of text data, such as scientific articles or legal documents.

8 code

9 contributions

Name	Contribution
Alice	Data preprocessing and model training
Bob	Baseline methods implementation

10 references

- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners.
- Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, Ł., & Shazeer, N. (2019). Generating Wikipedia by summarizing long sequences.
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need.