COE379L - Project 04
Martin Infante, mbi244
Version History:

- Predator-Prey Danger Detector deployed on Multi-Model Inference Server (oldest)
- Danger Proximity with Nearest Civilian Validation in response to Armed Threats

---

Fine-tuning a BERT Transformer for Better Punctuation Accuracy on Auto-Generated Captions

-> From Project Ideas section on Project 04 Assignment

> Perform sentiment analysis, text summarization or other classical NLP tasks on commonly available datasets such as social media postings, product reviews, articles or papers, etc. What model(s)/techniques will you use? You might consider using LLMs/transformers as part of this project. How will you evaluate the model?

INITIAL PROPOSAL

Introduction

My proposal focuses on improving YouTube's auto-generated captions by fine-tuning a BERT transformer model to correct punctuation errors using Hugging Face's YouTube Caption Corrections dataset. An evaluation script will quantify an accuracy score–rooted on the subset of punctuation-based errors found in the original auto-generated captions–by comparing model outputs against human-corrected captions.

Datasets

Hugging Face's YouTube Caption Corrections dataset

- *default_seq,* lists the auto generated captions
- *correction_seq,* lists the manually corrected word at index if *diff_type* at said index list a value > 0
- *diff_type,* lists a number between 0-8 based on error produced by auto generated captions.

For the context of this project, *diff_type = 2* is the only case to be considered, as this describes punctuation differences between auto-generated and corrected captions. A BERT model will then be trained to find a better punctuation to use, rather than the incorrect one. So for our use case, items in *default_seq* will be masked at indexes where *diff_type* has a value equal to 2. This modified string list will be fed to the BERT model to notify it that the original punctuation at index was incorrect, and should make an inference on the correct punctuation to use.

Technologies

BERT base model

- Pretrained via MLM

Products to be Delivered

The final deliverables include a fine-tuned BERT model specialized in punctuation correction for YouTube captions, and an evaluation report detailing performance improvements of the BERT model for caption generation.

FINAL REPORT

Introduction and project statement

The project focuses on improving the quality of YouTube's auto-generated captions by fine-tuning a BERT transformer model specifically to correct punctuation errors. Auto-generated captions often lack proper punctuation, reducing reading comprehension when accompanied by audio output. To address this, the project utilizes the YouTube Caption Corrections dataset from Hugging Face, which contains auto-generated captions with manual human corrected annotations. The punctuation types that could be considered tokens include the following set of symbols: ['.', ',', ';', ':', '-', '?', '!']. Underlying logic behind model creation stems from the idea that the transformer model would be trained to detect patterns in day-to-day speech, and by training the model on cases where it knows there is a punctuation error, the fine-tuned model would be able to extrapolate patterns unique to the type of diction present in Youtube videos.

Data sources and technologies used

The dataset comprises sequences of tokens from auto-generated captions (default_seq), corresponding human-corrected tokens (correction_seq), and error type annotations (diff_type). For this project, only tokens marked with punctuation errors (diff_type = 2) are masked and used as targets during training. The pretrained BERT base model is then fine-tuned on this processed dataset using Hugging Face's Transformers and Datasets libraries. Pipelines were used to make inferences on cleaned data. Tokenizers, collators, and trainers were also used in model instantiation. Lastly, the base for the BERT model originated from the AutoModelForMaskedLM pretrained checkpoint: "google-bert/bert-base-uncased".

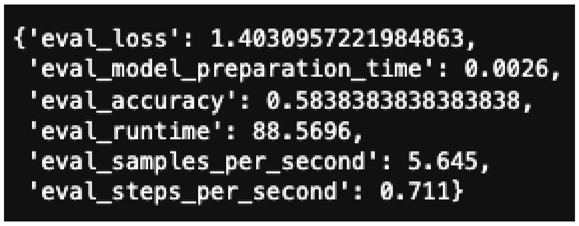
Methods employed

Raw dataset needed considerable preprocessing before it could be used for training. Each single entry in default_seq (which we will refer to as the auto generated caption) had multiple punctuation errors, and the project goal was to conduct MLM fine-tuning with a single mask per sequence. So the methodology that was adopted in order to create a usable dataset involved a mapping function *find_punctuation_errors()* that would splice the auto generated string from the beginning index, up until the second punctuation error was detected. The rationale behind this decision was that BERT is a bidirectional model, so we wanted to preserve the accompanying data on both sides of the mask token. Once the dataset was cleaned, it was split into 60% train, 20% validation, and 20% train. Afterwards our model checkpoint was pipelined, using the "fill-mask" NLP task to check the base model's capabilities.

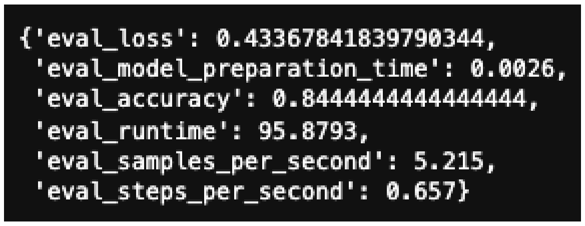
As stated earlier, we used the pretrained checkpoint: "google-bert/bert-base-uncased", which we then used to establish our tokenizer and data collator (with padding). Our evaluation metric was loaded with ‘accuracy’. The cleaned dataset from earlier was then tokenized with another mapping function *preprocess()* that would call our established tokenizer on the *masked_default_seq* feature in order to create the necessary *input_ids*, *token_type_ids*, and *attention masks* of the input strings. This mapping function also created the *labels* feature in the dataset which is the same length as the *input_ids* feature. The *labels* feature contains an id value of -100 at all indexes, excluding the location of the masked token (i.e. the index where the corrected punctuation should be located). The rationale behind this was for the model to compute loss only at the masked position. The tokenized datasets were then stripped of unnecessary features, leaving only the following columns: *input_ids, attention_mask*, and *labels*. Lastly, the training arguments were set up and the model was finally instantiated with the tokenized dataset attributes.

Results

Due to issues with the kernel force restarting in the jupyter notebook, the model was only trained with one epoch, and with no validation set. Now, the image on the left showcases the results from the base model, while the image on the right showcases the results from the fine tuned model. There was a little over a 25% increase in evaluation accuracy, indicating that the fine tuned model was able to generate the correct punctuation mark more consistently than before.



```
{'eval_loss': 1.4030957221984863,
 'eval_model_preparation_time': 0.0026,
 'eval_accuracy': 0.5838383838383838,
 'eval_runtime': 88.5696,
 'eval_samples_per_second': 5.645,
 'eval_steps_per_second': 0.711}
```



```
{'eval_loss': 0.43367841839790344,
 'eval_model_preparation_time': 0.0026,
 'eval_accuracy': 0.8444444444444444,
 'eval_runtime': 95.8793,
 'eval_samples_per_second': 5.215,
 'eval_steps_per_second': 0.657}
```

Other Deliverables

The fine tuned model is public on HuggingFace and I have attached an accessible link here:
https://huggingface.co/Martin8156/finetuned-BERT-punctuation-restoration

References

[1] Google. “BERT Base Model (Uncased).” Hugging Face,
https://huggingface.co/google-bert/bert-base-uncased. Accessed 5 May 2025.

[2] Community Datasets. “YouTube Caption Corrections.” Hugging Face,
https://huggingface.co/datasets/community-datasets/youtube_caption_corrections/viewer/default/train.
Accessed 5 May 2025.