Punctuation Restoration

Padakanti Srijith

Dataset Creation: This step can be ignored in my assignment as I have been assigned with a Kaggle dataset named “NLP Mental Health Conversations” which can be found at kaggle.

Dataset Understanding:
This dataset consists of 2 columns which are the question/context from the user to the psychologist and the other columns is the response from the psychologist to the user.

Each of the sentences have clearly marked out punctuation marks. There are a total of 3512 context-response pairs i.e. rows in the given data.

Out of which 995 rows have unique context/question and 2479 rows have unique responses.

Since our task is punctuation restoration, we just need all the different paragraphs which are punctuated grammatically. So taking all the unique context/question and responses would suffice as out dataset. So, we can see from the above data that we would be having 2479 paragraphs of text as the document asked us to look at the responses for the gold data.

Data Preprocessing:

1) Getting all the unique values(paragraphs in our case) in the data. (3474 paragraphs)
2) Removing the nan or null values. Removed one nan value from the data.(3473 paragraphs)
3) Splitting the paragraphs into train and test.
4) Removing all the punctuation marks except the three ‘,’, ‘.’, ’?’ from the paragraphs.
5) Now making the data into the format that can be used. By labelling and processing the data.

Now that we have the data then next step is finetuning.

At first I though of just giving the sentences without punctuations as inputs and sentences with punctuations as output and train the model but after some though I realised that it wouldn’t preform ideally.
So then I thought of using the bert embeddings of the unpunctuated text and then training a small classifier to predict what punctuation will come next given the text it has seen so far. However it also had it drawbacks as the classifier doesn’t know when to predict and when not to.
Then the idea was to predict by somehow calculating the probability of the punctuation after every token.

Then I realised that it’s a sequence labelling task and then chose BERT as we know that it excels at understanding the contextual representations within the sequence of words. Then each word, a context window of surrounding words is created, and a special token (zero) for the tokens without punctuation. I thought of doing it with 3-gram next.
Then i trained the model with the domain specific dataset I have. At the end of the training I save the best performing model after the end of the last epoch.
I used accuracy, precision, recall and F1-score to measure the model’s ability to predict.
The baseline f1-scores were:

Overall:	0.752
Comma:	0.712
Period:	0.819
Question:	0.723

This system involves four main components: data.py, model.py, train.py, and evaluate.py

1. Data.py

Purpose:
This code is responsible for loading and preprocessing the input data for punctuation prediction.

- load_file(filename): Loads text data from a file.
- encode_data(data, tokenizer, punctuation_enc): Converts words to BERT tokens and punctuation to given encoding.
- insert_target(x, segment_size): Creates segments of surrounding words for each word in x and inserts a zero token halfway the segment.
- preprocess_data(data, tokenizer, punctuation_enc, segment_size): Combines the above functions to preprocess the input data.
- create_data_loader(X, y, shuffle, batch_size): Creates a DataLoader for the preprocessed data.

2. Model.py

Purpose:
Defines the neural network model for punctuation prediction using BERT embeddings.

BertPunc: A neural network class inheriting from nn.Module, consisting of a BERT layer, batch normalization, a linear layer, and dropout.
Usage:
Initialization: BertPunc(segment_size, output_size, dropout)
Forward pass: forward(x)

3. Train.py

Purpose:
Trains the BERT-based punctuation prediction model.

validate(model, criterion, epoch, epochs, iteration, iterations, data_loader_valid, save_path, train_loss, best_val_loss, best_model_path, punctuation_enc): Validates the model on the validation set.

train(model, optimizer, criterion, epochs, data_loader_train, data_loader_valid, save_path, punctuation_enc, iterations=3, best_val_loss=1e9): Trains the model with the specified hyperparameters.

Workflow:
Load hyperparameters.
Load and preprocess training and validation data.
Initialize the BERT-based punctuation prediction model.
Train the top layer of the model.
Fine-tune all layers of the model.
Save the best-performing model.

4. Evaluate.py

Purpose:
Evaluates the BERT-based punctuation prediction model on a test set.

Workflow:
Load hyperparameters.
Load and preprocess test data.
Load the trained model.
Generate predictions on the test set.
Evaluate precision, recall, and F1 score

Even though I was able to lookup the baseline i wasn't able to run it as the computation resources i had weren't enough. As I came home because of sankranthi and wasn't able to access the college server(gpu resources). Had to run on the laptop and colab so wasn't able to fully run it.

Future Work:
Given the constraints during the project, future work involves running the model on more robust computational resources to achieve a comprehensive evaluation and further optimization and fine-tuning of hyperparameters could be done to improve model performance.