Comparison of Fine-tuned BERT and Pretrained Nemo Models for Punctuation Restoration

Introduction:

This project aims to compare the performance of two different models, namely Fine-tuned BERT and Pretrained Nemo, for punctuation restoration. Punctuation restoration is a crucial natural language processing task that enhances the readability and coherence of text. The comparison will be based on the Jaccard similarity metric, which measures the similarity between the original and restored sentences.

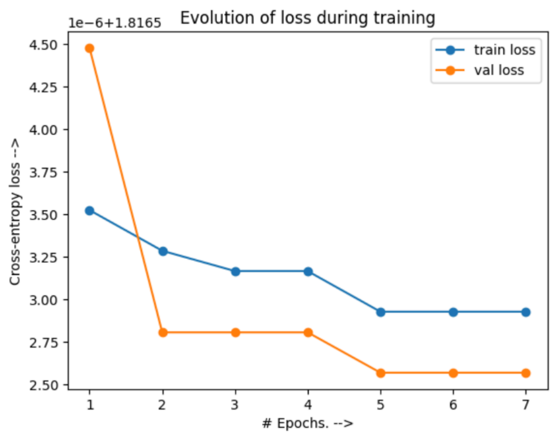
Data Pre-processing

Data cleaning is a crucial preprocessing step, ensuring the quality and reliability of the dataset. In this project, the cleaning process involves addressing NaN values, replacing special characters, and standardizing text. The removal of non-informative elements, such as extraneous whitespace and control characters, contributes to uniformity. Additionally, tokenization and lemmatization aid in creating a standardized and consistent representation of words. Special entities like URLs, numbers, and dates are replaced with placeholders, enabling a focused analysis. The meticulous handling of these aspects not only enhances the model's robustness but also facilitates meaningful insights by presenting the text in a structured and standardized format.

Fine-tuned BERT Model:

Model:

The decision to employ a BERT + LSTM model was driven by the need to leverage the contextualized embeddings provided by BERT while capturing sequential dependencies with LSTM. BERT, a pre-trained language model, excels in understanding context but lacks sequential information. By integrating an LSTM layer, the model gains the ability to capture patterns and dependencies within the text. This combination empowers the model to deliver enhanced performance in punctuation restoration tasks, where both contextual understanding and sequential coherence play pivotal roles.
The BERT + LSTM architecture strikes a balance, harnessing the strengths of both components to achieve better punctuation predictions.



Model Architecture:

The model architecture combines a BERT model, an LSTM layer, and a fully connected layer. The BERT model provides contextualised word embeddings, while the LSTM layer captures sequential information.

The model is trained using the CrossEntropyLoss function and the Adam optimizer.

Training and Evaluation:
The model is trained for seven epochs, with monitoring of both training and validation losses.
After training, the model is evaluated on a test set using a custom function to restore punctuations.
The Jaccard similarity metric is computed to measure the accuracy of the model.

Pretrained Nemo Model:

Model Implementation:
The Nemo model is based on the PunctuationCapitalizationModel, a pre-trained BERT-based model. The model is loaded from the Nemo library, and punctuation is added to the sentences using the add_punctuation_capitalization function.

Evaluation:
The restored sentences from the Nemo model are compared to the original sentences using the Jaccard similarity metric.

Limitations:

Limited Computational Power:
The primary limitation of this project is the lack of sufficient computational power. Due to these constraints, only a subset of the dataset was used, potentially limiting the model's ability to generalize to a broader range of examples.

Time Constraints for Training:
Training the Fine-tuned BERT model required more time due to the subset size and computational limitations. This constraint may impact the exploration of different hyperparameters and limit the model's overall training potential.

Hyperparameter Tuning:
The project acknowledges that additional hyperparameter tuning may be required, especially after changes in dataset size and model architecture. Due to constraints, an exhaustive search for optimal hyperparameters was not performed.

Comparison:

The final scores for both models, based on Jaccard similarity, are calculated.

BERT + LSTM	**Nemo**
0.18189	0.95370

Conclusion:

The pre-trained Nemo architecture performs better on the dataset. Further analysis and experimentation may be required to fine-tune hyperparameters and explore additional pre-trained models for improved performance.