Introduction

In speech recognition and transcription, the resulting text often lacks punctuation, leading to blocks of text that are difficult to read and understand. This poses a significant challenge in scenarios where accurate and comprehensible transcriptions are needed, such as in legal documentation, medical records, and educational materials. The task at hand is to develop a system that can accurately insert punctuation into such unpunctuated text, enhancing readability and maintaining the original spoken content's integrity.

Literature Review / Related work

Punctuation restoration is a widely studied topic in the field of NLP. Transforming a segment of speech into written-text using automatic speech recognition (ASR) systems then restoring punctuation in the text can significantly enhance the performance of downstream tasks (such as real-time translation, sentiment analysis, etc.). One of the current SOTA techniques employs a network layer combination of Bert-BLSTM-CRF (Yi et al. 2020), surpassing other similar or previous language model architectures. This paper focuses on various factors that may affect the experiment results when fine-tuning, differing from other research that emphasizes major adjustments (training methods, model architectures, etc.). The goal is to find the optimal combination of different fine-tuning parameters.

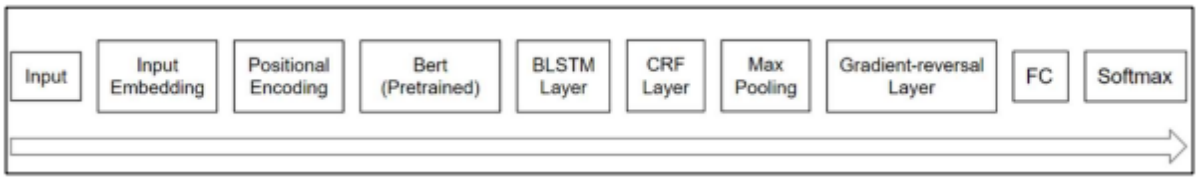


Figure 1: BERT-BLSTM-CRF Language Model Architecture

Detecting sentence boundaries is another challenging issue when performing punctuation restoration on long texts. For single-sentence punctuation restoration, the appropriate punctuation can be determined by the keywords or the context of the sentence. However, when a text consists of multiple sentences, the ability to identify the boundaries of the sentence then individually restoring each sentence becomes the core problem. Studies have shown that the CRF network layer achieves the best results in determining sentence boundaries (Sanchez 2019).

Previous research often uses the "The International Conference on Spoken Language Translation" (IWSLT) dataset for both training and evaluating model performance. Using the same dataset ensures a fair reflection of the model's performance without worrying that the results may be biased due to the differences between the training data and evaluation data. The F1-score is not only one of the common methods used to evaluate the results of punctuation restoration of language

models but is also an indicator of performance in most grammar error correction tasks. Compared to precision and recall, the F1-score more comprehensively reflects the model's performance.

Besides BERT, roBERTa and T5 are also commonly used pre-trained models. RoBERTa improves upon BERT by using a dynamic masking technique, which has been shown to be more effective than the static masking used in BERT. It also employs longer training times, larger training batches, longer training sequences, and larger training datasets to enhance BERT.

T5, on the other hand, uses a completely different training logic. It converts all natural language tasks into text-to-text tasks and uses the C4 dataset as its basis, which is significantly larger than other training datasets. The T5 pre-trained model is also categorized based on the number of parameters, with the T5-11B (which uses 11 billion parameters) being the best overall performer. However, the computational resources required for training it are far beyond normal ranges, making its time cost less favorable compared to BERT-derived models (roBERTa, ALBERT, etc.).

However, the advantage of the T5 model lies in its vast training data volume, allowing it to be fine-tuned with specific data to handle most natural language tasks. For the punctuation restoration task, for example, by fine-tuning it with a certain amount of unpunctuated sentences and their punctuated counterparts, the T5 pre-trained model can be specialized to solve punctuation restoration tasks. Our project employs this method and investigates other possible factors that may affect the experiment results when fine-tuning, aiming to find the best overall combination.

Dataset

For our project, we decided to use the British National Corpus (BNC) as our dataset. As mentioned above, punctuation restoration can serve as a down-stream task of ASR, while BNC is a collection of English text containing approximately 100 million words, with 10% of which is derived from spoken language while the other comes from written text. BNC also covers a wide variety of genres, including academic, commerce, science, etc., which can avoid potential biases in punctuation usage related to specific topics.
To make our model to adapt better under some abnormal conditions, we only removed words that are not readable under utf-8, which will result in “[UNK]”. In the “Data” folder of our project, the original data of the BNC is stored in the following format with two columns “input” and “target”, the format of input is “grammar: {unpunctuated_sentence}”, and the format of target is “{original_sentence}”. This format is chosen to fit our package that we used to implement the fine-tuned T5 models (happy transformer). On top of that, in the “Length” folder of our project, we grouped the corpus in the dataset by its length, stored in “Length_{length}.csv”, for further experiment and discussion.

Baseline

We decided to implement this task by considering it as a text-to-text task. In this case, for training we simply define the input as a non-punctuated version of a given corpus, and the output is the original version of the corpus. Although this task can be also treated as a classification task, where we consider every gap between two words then classify which punctuation should be used or simply not use any of them according to the given context (which can be implemented using BERT). Since we mainly focus on fine-tuning on this project, we consider the not fine-tuned t5-base model as our baseline.

Main Approach

For our own implementation of a punctuation restoration model, we used a python package called “Happy Transformer”, which allows us to fine-tune a chosen pre-trained model (t5-small, t5-base, BART, etc.) using our own chosen dataset. Happy Transformer is basically a modularized version of the pytorch and transformer package of python, where we can make a concise approach to setup the parameters for fine-tuning a pretrained-model.

Our main approach is to fine-tune a t5-base pre-trained model using a train dataset of 3500 entries, an evaluation dataset of 1500 entries, and a test dataset of 4000 entries for evaluating the performance of our model.

Evaluation Metric

As mentioned in the literature review, f1-score is commonly used when it comes to grammar error correction tasks. In order to calculate the f1-score, we fed the unpunctuated sentence in our test dataset into our fine-tuned model, and obtained our result dataset. Then we compare our result one by one with the original version of the corpus stored in our test dataset, and evaluate the whether the appearance of each kind of punctuation in each position is appropriate, which allow us to obtain the confusion matrix for every kind of punctuation (including period, comma, exclamation mark, question mark, hyphen, colon, and semicolon). Then we can calculate the accuracy, precision, recall and f1-score for each of the punctuation.

Results & Analysis

Our baseline, which is the not fine-tuned version of t5-base, cannot produce a result that is close to the actual answer, therefore we decided to neglect its result and consider it as not being capable of performing punctuation restoration. As for the result for our main approach is the following:

```
period
acc: 0.9353389531068598, pre: 0.7651006711409396, rec: 0.5752845910104448, f1: 0.6567524115755627
comma
acc: 0.9975266266215739, pre: 0.8248175182481752, rec: 0.39649122807017545, f1: 0.5355450236966824
exclamation mark
acc: 0.9971480490636515, pre: 0.11363636363636363, rec: 0.06329113924050633, f1: 0.08130081300813008
question mark
acc: 0.9977790116601888, pre: 0.38461538461538464, rec: 0.028901734104046242, f1: 0.05376344086021506
```

Since the fine-tune process of T5 requires a lot of computing resources, increasing the dataset size and the maximum input and output token when fine-tuning will cause the required computing resources to increase exponentially, therefore we trained it on a rather very small dataset but still get a reasonable result. We made 3 additional experiments to take a deeper look at the task.

Experiment 1:

We experimented on the size of the dataset for fine-tuning on t5-base pretrained model, where model#1, model#2, model#3, model#4 is fine-tuned with a dataset of size 1000, 3000, 5000, 10000 respectively.

Period restoration:

Model	Accuracy	Precision	Recall	F1-score
model#1	0.94	0.54	0.05	0.09
model#2	0.97	0.81	0.66	0.73
model#3	0.98	0.85	0.70	0.77
model#4	0.98	0.89	0.73	0.80

Comma restoration:

Model	Accuracy	Precision	Recall	F1-score
model#1	0.95	0	0	*
model#2	0.95	0.38	0.01	0.03
model#3	0.95	0.51	0.07	0.12
model#4	0.95	0.54	0.15	0.23

We can see that there is a positive correlation between the size of the dataset and the f1-score of the restoration of period and comma.

Experiment 2:

We experimented with using different pre-trained models but fine-tuning with the same dataset (size 10000).

Period restoration:

Model	Accuracy	Precision	Recall	F1-score
T5-small	0.99	0.82	0.88	0.85
T5-base	0.99	0.89	0.93	0.91
BART	0.99	0.90	0.93	0.91

Comma restoration:

Model	Accuracy	Precision	Recall	F1-score
T5-small	0.95	0.55	0.09	0.16
T5-base	0.96	0.68	0.50	0.58
BART	0.96	0.73	0.53	0.61

We can see that T5-base and BART performs a lot better compared to T5-small, so the size of the pre-trained model also has a positive correlation with the performance of the model.

Experiment 3:

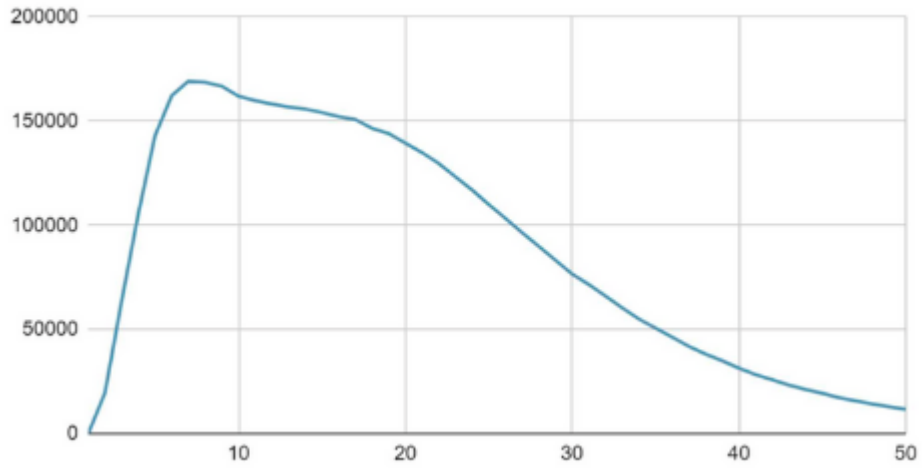
We used 6 corpus of our own choices and compared the restoration result of our own model and ChatGPT 3.5 and ChatGPT4. The full result is stored in the last part of the main.ipynb in our repository. It is clear that even ChatGPT 3.5 has completely out performed our model due to the difference in size of the dataset. For this task, though, we found out that it is kind of unfair to make comparisons like this, since the corpus that we choose might have been included in the dataset for ChatGPT, so most of the time it can be 100% correct when performing punctuation restoration.

Analysis:

We come to the conclusion that the amount of dataset used for the pretrained model or fine-tuning both has a positive correlation with the performance of the model. Unlike other text-to-text task like question answering or corpus analyzing, where there could be a lot of unpredictable features that could affect the performance of the model besides the size of the dataset, punctuation restoration is a straightforward task where the more dataset a model uses, the higher possibility the model is going to perform restoration correctly.

Error Analysis

When examining the result of our fine-tuned model, we found out that our model cannot produce a valid result when the input size is too small, sometimes it even outputs random words and punctuation that cannot form a full sentence. So we doubt that the length of our corpus that we used to fine-tune our model does not distribute uniformly.



As shown in the graph, we can see that most of our corpus has length between 8 to 20, while the corpus with length below 5 is only around 7.5%. When we randomly extracts data from our dataset, the proportion of short corpus will be low too, already having not much context to calculate the appropriate punctuation, the model does not have much experience on handling such cases, therefore can explain the reason why our model performs poorly in short corpus. To solve this problem, we might have to make our model handle such cases reflexively, since the punctuation pattern would be a lot more predictable when the length is short.

This is the same for the restoration of punctuations besides comma and periods, the proportion of other punctuations are relatively low, which leads to the lack of examples when fine-tuning on a rather low size dataset. In most of the experiments, several entries on the confusion matrix of these punctuations would be 0, which makes it impossible to analyze the performance of its restoration, which is why our project focuses mainly on only period and comma after performing several experiments of different kinds.

Future Work

In future work, we can add several enhancements to improve our punctuation restoration model, such as using advanced SOTA models like BERT, training with larger and more diverse datasets. By applying these changes, we may enhance the accuracy and the F1-Score of our model, making it a valuable tool for various text processing applications.

Code

https://github.com/chou-ting-wei/NYCU_AI-Final-Project

Contribution of each member

Name	Proportion	Detail
賴邑城	40%	Extract and preprocess BNC, Most part of main.ipynb Literature review and introduction
周廷威	30%	Most part of presentation, report video editing, data processing
蔡尚融	30%	Most part of lab03, some part of presentation, report

References

1. Păiş, Vasile, and Dan Tufiş. "Capitalization and punctuation restoration: a survey." Artificial Intelligence Review (2021): 1-42.
2. Yi, Jiangyan, et al. "Adversarial transfer learning for punctuation restoration." arXiv preprint arXiv:2004.00248 (2020).
3. Sanchez, George. "Sentence boundary detection in legal text." Proceedings of the natural legal language processing workshop 2019. 2019.
4. Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).
5. Courtland, Maury, Adam Faulkner, and Gayle McElvain. "Efficient automatic punctuation restoration using bidirectional transformers with robust inference." Proceedings of the 17th International Conference on Spoken Language Translation. 2020.