NLP Winter 2021/2
Final Project

Menachem Brief

Abstract

The aim of this project is to deal with the problem of converting raw English text, e.g., lower case words with no punctuation, into punctuated text with structured paragraphs. This problem arises in multiple domains including text generation, speech recognition, writing assistance, and document recovery. Recent works have shown good results in the task of punctuation prediction/generation, including capitalization and correct usage of common punctuation characters (‘,’,’.’,’?’,’!’), but to the best of my research no works have been published dealing specifically with paragraph reconstruction. An additional contribution of this work is an extension of automatic punctuation to more types. The main component of this project is the creation of a supervised model, based on a pretrained encoder, that predicts what punctuation comes before/after each word, if the word should be capitalized, and whether it’s the last word in a paragraph. The dataset for this project contains 85 fiction books obtained from Project Gutenberg[1], all in English. The model achieves an F1 score of 0.8828 for capitalization, 0.543 for predicting paragraphs, and varying results per type of punctuation, similar to recent results on other datasets.

1 Introduction

Punctuation, capitalization, and structure play a major role in our ability to comprehend text documents (Moore, 2016). Therefore, the ability to correctly format raw text is an important task. Punctuation allows for clarification of meaning, conveyance of emotions, and is also necessary for adherence to grammar conventions. Specifically, the types of punctuation that this project tries to predict are the following:
`!`, `"`, `'`, `(`, `)`, `,`, `-`, `.`, `:`, `;`, and **`?`.**
Capitalization, also known as true-casing, plays a key role both in our ability to identify entities and in our inference of sentence structure in many languages. The meaning of structure in this context is the splitting of text into paragraphs, mainly used to point out were one idea ends and another begins. Capitalization is the easiest task of the three, due to the many hints a model can get about this aspect, especially repeating proper nouns that are always capitalized. The ease of creating punctuation varies greatly, with weak correlation to frequency. Paragraph endings are the most difficult of the three since they involve understanding where one idea ends and another begins, and additionally are infrequently used.

The task of automatic punctuation, which is a special form of machine translation, can be formulated as one of two problems: a prediction task per token or a sequence generation task. The approach in this work is like the former formulation, namely, to predict the relevant tags per sequence using each token’s context within the sequence, and then to use that prediction deterministically to construct the predicted text.
While most previous works focus on shorter texts, such as those created by ASR (automatic speech recognition) or news articles, this work attempts to reconstruct text from books. The dataset contains numerous authors and writing styles and evaluates a test set containing many whole chapters.

[1] https://www.gutenberg.org/

2 Related Work

Predicting punctuation has been the subject of many works in computational linguistics and works in the field have evolved in similar ways to other NLP tasks.

2.1 Earlier Methods

Earlier works (e.g., Beeferman et al., 1998) in the field used classic methods such as Hidden Markov Models using n-grams to create language models that can be used to construct punctuated text. Slightly later (e.g., Lu and Ng, 2010) works utilized conditional random fields, and were considered SOTA until the rise in popularity of recurrent neural networks (RNNs). For a few years, works that utilized word vectors trained on large corpora and combined them with RNNs (e.g., Tilk and Alumäe, 2015) yielded the best results; until the introduction of transformer architectures.

2.2 Recent Methods

Transformers have transformed many domains, but arguably none more than NLP. These models tend to be very large and demand long and intensive training, making it difficult to train a specific model per task. This has led to a shift towards training large general-purpose models, such as BERT (Devlin et al., 2019), trained using gigantic corpora.

By using models such as BERT it became possible to achieve SOTA results with fewer resources by only finetuning a pretrained model.

Virtually all the best results in recent years are based on finetuning much larger models. Multiple comparisons have been conducted between this approach and previous approaches, including an extensive comparison by Courtland et al. (2020), demonstrating that by using any of the commonly used pretrained models they could achieve better results than previous methods. By using RoBERTabase (Liu et al., 2019), Courtland et al. were able to achieve the best results at the time, combining fine tuning with multiple predictions per token, and selecting the best of the predictions. Others, such as Nguyen et al. (2019), approached the problem using seq2seq models, generating entire sentences with their punctuation. In their work they also combined the tasks of true-casing and punctuation, using a single multitarget model. Although this method achieved very good results, in generating the sequences and incorporating bidirectionality, a large computational overhead is required.

$$
\begin{array}{ccc}
(bef_1^1, aft_1^1, cap_1^1, br_1^1) & \dots & (bef_T^1, aft_T^1, cap_T^1, br_T^1) \\
\dots & & \dots \\
\dots & & \dots \\
(bef_1^N, aft_1^N, cap_1^N, br_1^N) & \dots & (bef_T^N, aft_T^N, cap_T^N, br_T^N)
\end{array}
$$

Figure 1. Model output shape. $\dim(bef_t^n), \dim(aft_t^n)$ is 12 and $\dim(cap_t^n), \dim(br_t^n)$ is 2 . Altogether, with $N = 16, T = 150$, it outputs four tensors: two of shape $16 \times 150 \times 12$, and two of shape $16 \times 150 \times 2$.

3 Model

In an approach inspired by both Nguyen et al. and Courtland et al., this work uses a single multitarget network to predict all three desired outputs, as well as aggregate predictions for each class of punctuation. Due to the tokenization mechanism BERT uses, greedily breaking up unknown words into sub-words, BERT was selected over RoBERTA to gain another simple heuristic for aggregating predictions.

3.1 Architecture

The architecture of the model is composed of a pretrained encoder, BERT, fed by sequences that are 150 tokens long, with the output passed to four different fully connected (FC) heads. A log-softmax activation is used on the output of each head. The architecture of each head is:

```
Linear(embedding_dim,100) → ReLU → LayerNorm →
Linear(100, num_classes)→LogSoftmax.
```

The full output of the model per batch is shown in Figure 1.

The first two FC's output log-probabilities across 12 classes, the number of possible punctuations in this task, with the first corresponding to the punctuation before a token and the latter to the punctuation following it. The third head outputs log-probabilities for whether the word is capitalized, and the fourth for whether the token is the last in the paragraph. Negative log likelihood was used as the loss function for all heads.

Label	!	"	'	(	)	,	-	.	:	;	?	Capital	Break
Number(10^3)	26.2	20.5	69.5	4.4	4.4	46.7	40.7	294.3	12.2	48.9	28.9	686.5	125.5
Relative(%)	0.35	2.76	0.93	0.06	0.06	6.28	0.54	3.95	0.16	0.64	0.38	9.22	1.68

Table 1. Number of occurrences of each label, and its relative percent in the entire dataset.

3.2 Aggregation Boosting

The need for separate classifiers for before and after stems from two reasons: 1.Not all punctuation is mutually exclusive, meaning that ` . ` may follow a word, but the next word can be preceded by ` ” `. 2.To gain a boost in performance, inspired by Courtland et al.

The first form of aggregation was done by combining the different predictions per sub-word. Since BERT breaks up unknown words into multiple words, the word `homeschooling` might be broken into: `home`,`##school`, and `##ing`. Thus, the seemingly redundant predictions can be combined heuristically to create a single better prediction. Specifically, I chose to boost recall over precision, meaning the rarest of each class was selected from the sub-word predictions, and if any were tagged as capitalized or paragraph endings, then the token always got a positive label.

The second form of aggregation was done by using another heuristic, this time taking advantage of the before/after predictions. If both agreed on the punctuation then that value was returned, otherwise, if it was possible that both were valid (e.g., `he asked, “do you know?” `) both were predicted, otherwise, the same logic as before was applied.

4 Data

The dataset used for this project is part of a much larger dataset shared via Kaggle[2]. The dataset contains the meta data necessary for downloading books from Project Gutenberg using the `gutenberg` module. The main advantage of the books available on Project Gutenberg is that their copyrights have expired, allowing anyone to use them freely. Of the 15,000 books published on Kaggle, only those that were from fiction genres were selected to avoid dealing with non-textual features, and of those, only books identified as being in English were included. The filtered dataset contains 85 books with a total of 6,115,035 words. A summary of the data is shown in table 1.

A simple preprocessing was performed to remove punctuation that was not part of the task and replaced multiple line breaks with a marker noting the paragraph ended. The annotation process was entirely straightforward, except that words that were fully capitalized, e.g., “THE END”, were treated the same as standard capitalization. To remove the publication information, table of contents, etc. the first 50 paragraphs of each book were removed, a specific heuristic for this dataset. Each book was then divided into 150 token long sequences. The words were tokenized using the BERT tokenizer from the `hugging face` module (with the annotations adjusted to the sub-word division), and then the sequences were divided into three sections: train(75%), validation(15%), test(10%). Finally, the train, validation, and test from all the books were concatenated together.

5 Experiments and Results

All experiments used the architecture described in section 3, only in some replacing BERT with DistilBERT (Sanh et al., 2019). DistilBERT is a smaller version of BERT, which has been shown to achieve similar results with significantly less resource use. Since its average epoch time was close to half that of BERT’s (18 minutes vs. 33), the initial hyperparameter search was conducted with DistilBERT as the backbone.

Finetuning the pretrained model proved to be significantly better than simply training the FC layers. After some experimentation, the best training regime I found was using two separate optimizers - one for the pretrained model, and one for the heads, with learning rates of $3^{-5}, 10^{-3}$ respectively, and a scheduler that decreases the learning rate every 15 steps.

[2] https://www.kaggle.com/kerneler/starter-15000-gutenberg-books-5bec0363-3/data/

Label	!	"	'	(	)	,	-	.	:	;	?	Capital	Break
DistilBERT	0.36	0.7	0.94	**0.55**	0.5	0.67	0.75	0.73	0.4	0.12	0.64	0.86	0.48
BERT	**0.4**	**0.7**	**0.95**	0.48	**0.53**	**0.71**	**0.78**	**0.77**	**0.41**	**0.18**	**0.69**	**0.88**	**0.54**

Table 2. F1 scores per label for the best version of each model, with aggregated predictions.

After finding the best training regime the backbone was switched to the full BERT model, and as expected, the results were better. A sample of the best model's performance on a random excerpt from the test set can be found in Appendix A. The ratio between the performance of the two is consistent with previous works findings. In table 2 is a summary of the results per label, comparing BERT and DistilBERT. In Appendix B a more extensive report of the results can be found. The full code to recreate the results can be found on GitHub at the following link: https://github.com/menibrief/NLP_Gutenberg_Punctuation

6 Conclusions

This work provides a unified approach to reconstructing raw text back into a human readable document format. It deals with many more types of punctuation than most recent works have, adding more complexity to the task it attempts to solve. It is also, to the best of my knowledge, the first attempt to use books directly, which is less useful for ASR, but adds an additional complexity of author style and allows for much larger datasets.

Very good results were achieved using a relatively naïve mechanism, demonstrating yet again the power of pretrained models.

7 Future Directions

Although the results are good, further improvement can be expected by experimenting with some options that demanded more resources than I had at my disposal: using RoBERTA as the pretrained model and adapting the labels to it, using more books from more recent years, and using multiple predictions per sequence. In addition, training also on benchmark datasets should be conducted to verify the robustness of the model. Augmenting with other data would also help avoid overfitting to specific books, a possible drawback of this work.

References

Moore, N. What's the point? The role of punctuation in realising information structure in written English. *Functional Linguist.* **3,** 6 (2016). https://doi.org/10.1186/s40554-016-0029American Psychological Association. 1983. *Publications Manual.* American Psychological Association, Washington, DC.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv, abs/1810.04805*.Association for Computing Machinery. 1983. *Computing Reviews*, 24(11):503-512.

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv, abs/1910.01108*.

Beeferman, D., Berger, A.L., & Lafferty, J.D. (1998). Cyberpunc: a lightweight punctuation annotation system for speech. *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181), 2*, 689-692 vol.2.

Lu, W., & Ng, H.T. (2010). Better Punctuation Prediction with Dynamic Conditional Random Fields. *EMNLP*.

Tilk, O., & Alumäe, T. (2015). LSTM for punctuation restoration in speech transcripts. *INTERSPEECH*.

Courtland, M., Faulkner, A., & McElvain, G. (2020). Efficient Automatic Punctuation Restoration Using Bidirectional Transformers with Robust Inference. *IWSLT*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692 [cs]. ArXiv: 1907.11692.

Nguyen, B.P., Nguyen, V.B., Nguyen, H., Phuong, P.N., Nguyen, T., Do, Q.T., & Mai, L.C. (2019). Fast and Accurate Capitalization and Punctuation for Automatic Speech Recognition Using Transformer and Chunk Merging. *2019 22nd Conference of the Oriental COCOSDA International Committee for the Co-ordination and Standardisation of Speech Databases and Assessment Techniques (O-COCOSDA)*, 1-5.

A Sample Text

Original	Model output	Raw text
Sainte-Croix was put into an unlighted room by the gaoler, and in the dark had failed to see his companion: he had abandoned himself to his rage, his imprecations had revealed his state of mind to Exili, who at once seized the occasion for gaining a devoted and powerful disciple, who once out of prison might open the doors for him, perhaps, or at least avenge his fate should he be incarcerated for life. The repugnance felt by Sainte-Croix for his fellow-prisoner did not last long, and the clever master found his pupil apt. Sainte-Croix, a strange mixture of qualities.	Sainte-Croix was put into an unlighted room by the gaoler, and in the dark had failed to see his companion. He had abandoned himself to his rage; his imprecations had revealed his state of mind to Exili, who at once seized the occasion for gaining a devoted and powerful disciple, who, once out of prison, might open the doors for him, perhaps, or at least avenge his fate, should he be incarcerated for life. The repugnance felt by Sainte-Croix for his fellow-prisoner did not last long, and the clever master found his pupil apt. Sainte-Croix, a strange mixture of qualities.	sainte croix was put into an unlighted room by the gaoler and in the dark had failed to see his companion he had abandoned himself to his rage his imprecations had revealed his state of mind to exili who at once seized the occasion for gaining a devoted and powerful disciple who once out of prison might open the doors for him perhaps or at least avenge his fate should he be incarcerated for life the repugnance felt by sainte croix for his fellow prisoner did not last long and the clever master found his pupil apt sainte croix a strange mixture of qualities

B Full Results for BERT

Before token				After token			
Label	Precision	Recall	F1	Label	Precision	Recall	F1
!	0.3	0.45	0.33	!	0.35	0.52	0.4
"	0.69	0.77	0.71	"	0.12	0.3	0.15
'	0.89	0.97	0.92	'	0.93	0.97	0.95
(	0.44	0.62	0.48	(	0.17	0.28	0.19
)	0.45	0.8	0.53	)	0.4	0.66	0.46
,	0.71	0.7	0.7	,	0.7	0.73	0.71
-	0.84	0.75	0.78	-	0.84	0.75	0.78
.	0.65	0.63	0.63	.	0.82	0.73	0.77
:	0.21	0.31	0.24	:	0.4	0.53	0.41
;	0.16	0.34	0.19	;	0.13	0.34	0.17
?	0.48	0.6	0.5	?	0.68	0.73	0.69

Capital	0.8678	0.9038	0.8828
Break	0.5237	0.6066	0.543