Punctuation Prediction

SeyedMostafa Ahmadi
ahmadism@yorku.ca

Armin Gholampoor
arminir@yorku.ca

Abstract

Punctuation prediction aims at assigning proper punctuation in an unpunctuated text. In this paper, we propose three models and evaluate them based on their performance on this task. We use Wikipedia dataset Wikitext103V1 (Merity et al. [2016]) for both training and testing phases. For two of our models, we took advantage of a pre-trained BERT model for the NER task which showed promising results in a shorter training time in comparison with a RNN-based method. We report the accuracy of our models and discuss a detailed conclusion about them. We also compare our work with a previous work that used RNN similar to our RNN-based method, although their punctuations were limited (only 3) and the dataset was different and closed-source.

Keywords: Data Mining, Classification, Word Labeling, Deep Learning, BERT, RNN, LSTM, Natural Language Processing

1 Introduction

Text mining is the process of extracting previously unknown, understandable, potential, and practical patterns or knowledge from the collection of text data. Because of large amounts of unstructured text data generated on the Internet, text mining is believed to have high commercial value (Zhang et al. [2015]). This is because these texts usually contain valuable information hidden inside them. For example, the rapid data made in social media is a great source of information because it has a large community. However, it has become common practice not to use the correct grammar, spelling, and punctuation. This will lead to ambiguities like lexical, semantic, and syntactic. Consequently, we have to find meaningful information patterns from this data and provide a well-structured form of the data (Salloum et al. [2017a]).

Machine learning provides many powerful tools to analyze text-based datasets, such as hidden Markov models (HMM) [1], conditional random field (CRF) [2], maximum entropy models (MaxEnt) [3], support vector machines (SVM) [4], Naïve Bays [5], and deep learning (DL) [6] (Khan et al. [2016]). However, with the emergence of attention-based models in deep learning, a new era was begun in NLP. One of the cornerstones in these models is BERT (Devlin et al. [2018]).

BERT stands for Bidirectional Encoder Representations from Transformers. This model is better than previous methods in many ways, such as being pre-trainable from unlabeled data, fine-tunable, parallel learning, and context-aware.

Punctuation prediction is the task of assigning punctuations to an unpunctuated text. Punctuations are vital for understanding the meaning of a text. A text can be meaningless without punctuation or have a thoroughly different meaning. This can lead to destructive results in many fields, such as medical prescriptions. Punctuations are also highly crucial in Automated Speech Recognition (ASR) tasks as they give structure and meaning to texts. For example, voice assistants such as Siri or Alexa need to predict punctuations to distinguish between multiple user commands. Also, Speech-to-Text agents have to assign punctuations in the correct place, or they will only produce text that is not informative.

In this article, we have two primary goals. Firstly, we want to propose three models for the aforementioned task. Our models contain a combination of BLSTM and BERT. BLSTM stands for Bi-directional Long Short-Term Memories. It consists of two LSTMs stacked up on each other to analyze the text in two directions (start to end, end to start) with each layer. LSTMs [7] are deep learning models designed to overcome the vanishing gradient problem in RNNs. These models have a parameter-based memory in themselves that decreases the effect of long-term dependency issues.

Our second objective is to train our models with only a

[1] Awad and Khanna [2015]

[2] Lafferty et al. [2001]

[3] Berger et al. [1996]

[4] Evgeniou and Pontil [2001]

[5] Vikramkumar et al. [2014]

[6] Schmidhuber [2015]

[7] Hochreiter and Schmidhuber [1997]

small number of epochs instead of training a big network for a couple of weeks. This is because, as students, we do not have access to powerful and large-scale computing power. As a result, we have to tune our parameters such that our models can generalize their knowledge of the dataset faster.

Our idea of labeling each word comes from the methodology used in Named Entity Recognition(NER). As Yadav and Bethard [2018] explains, Named Entity Recognition is the task of identifying named entities like person, location, organization, drug, time, clinical procedure, biological protein, etc. in text. That is they assign a label to each word and classify them as the corresponding categories. Our strategy comes from the same origin. We want to process a text and assign each word a label indicating whether it is followed by punctuation.

In this article, we first review a couple of related works in this area in Section 2. Then, in Section 3, we discuss our methodology and explain the design of our models. We also describe the training procedure for each model. In Section 4, we state the result of our experiments. We compare the outcome of our models with each other and with another paper. Finally, we conclude with a discussion on challenges that need to be addressed in Section 5.

2 Related Works

In this section, we will briefly discuss five of the recent papers on the topic. Additionally, since we have used pretrained models, we will also mention transfer learning approach.

2.1 Robust Prediction of Punctuation and Truecasing for Medical ASR

Sunkara et al. [2020] proposed a conditional modeling framework. They used pretrained BERT-based models (BERT, BioBERT [8], and RoBERTa [9]). They also present techniques for fine-tuning models on specific domains to build task-specific language models. Finally, in order to reduce error rate of model, they use data augmentation.

2.2 NMT-based Segmentation and Punctuation Insertion for Real-time Spoken Language Translation

Cho et al. [2017] studied punctuation prediction in a real-time application in spoken language translation. They stated that NMT translators (Neural Machine Translation) achieve better results than PBMT (Phrase-Based Machine Translation). Moreover, this is the first work in which authors adapted an Encoder-Decoder model with attention for this purpose.

[8]A domain-specific language representation model pre-trained on large-scale biomedical corpora. See Lee et al. [2019].

[9]An optimized version of BERT that its pretraining phase is improved. See Liu et al. [2019].

2.3 Deep Learning for Punctuation Restoration in Medical Reports

Salloum et al. [2017b] introduced a novel feature extraction method in order to restore punctuation to clinical reports. They used a vocabulary reduction strategy by collapsing medical words into particular roots (morphemes) in the pre-processing step. They modeled the problem as a tagging problem. The goal is to assign each word a tag between (NONE, COLON, COMMA, PERIOD) to indicate the next letter coming after them. They used BRNN [10] with attention and a late fusion strategy on top of it. The reason they used BRNN was that this model performs well in learning long-range dependencies on the left and right of the current input word.

2.4 Punctuation Prediction Model for Conversational Speech

Żelasko et al. [2018] analyzed the problem of ASR in the context of telephone conversations. They commented that there is a lack of domain-specific datasets for conversational speech in this task and that they have to train the models on general-purpose texts. They also used BLSTM and CNN [11] models in their solutions and compared the performance of the two models.

2.5 Punctuation Prediction using a Bidirectional Recurrent Neural Network with Part-of-Speech Tagging

Juin et al. [2017] mapped the problem to a translation task. That is where unpunctuated sentences are translated into punctuated sentences considering the punctuation symbols as any other word. They used Part-of-Speech (POS) tagging method. Additionally, they used two models consisting of BRNN with attention in an Encoder-Decoder network and LSTM. They used LSTM because it learns the long-term dependencies well. Moreover, they used Wikitext103V1 dataset for their research.

[10]A combination of two RNNs training the network in opposite directions. See Schuster and Paliwal [1997].

[11]A Convolutional Neural Network is a class of Artificial Neural Network, most commonly applied to analyze visual imagery. See O'Shea and Nash [2015].

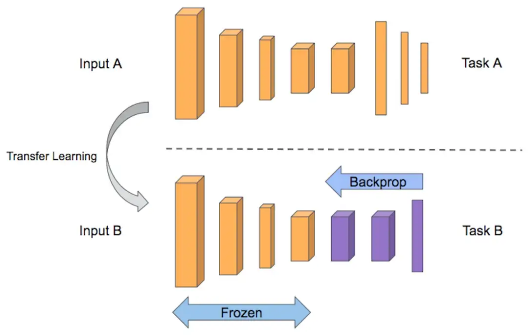


Figure 1: Transfer Learning Malgonde [2018]

2.6 A Comprehensive Survey on Transfer Learning

Zhuang et al. [2019] summarizes different strategies used in transfer learning. As they said, "Transfer learning aims at improving the performance of target learners on target domains by transferring the knowledge contained in different but related source domains." In fact, when we are using a pretrained BERT transformer, we are utilizing this method. In figure 1 (Malgonde [2018]) we can see an illustration of transfer learning.

3 Methodology

In this section, we will propose three approaches for doing the task. Moreover, we will illustrate models' designs in the according figures.

3.1 BLSTM

In this model, we only used a bidirectional LSTM and no parts of a BERT model were used. These models are known for taking a long time to train because of being sequential. In Salloum et al. [2017b], they also used a similar approach. The only difference between the model we implemented and the model presented in Salloum et al. [2017b] is that they have an attention node in their model. We, instead, use a weighted loss to reach a better training approach; weights are based on the frequency of the labels and they have an opposite relationship. For example, label *O* is the most frequent label and the least interesting to predict. If the model predicts *DOT* or *COMMA* correctly it would be more interesting.

We used fasttext embedding [12] to convert each word to a 300-dimension vector. We use these embeddings as input to the model and expect the model to assign the correct labels for each word. Figure 2 shows the structure of this model.

[12]A method for word representation using vectors. See Bojanowski et al. [2016].

3.2 BERT

For our main model, we chose BERT as it is the most known transformer model these days, although it is not state-of-the-art. However, thanks to the python library we used for training the model, the replacement of any other state-of-the-art model in our approach is as easy as changing a line of code.

We used a BERT model which is pre-trained on NER task. Our theory is based on the fact that both of these tasks look similar. For example, Nelson Mandela is a person's name, but it can be a street name as well.

- Sentence 1: Nelson Mandela (Person) was born on 18 July 1918.
- Sentence 2: Nelson Mandela (Location) is an affluent and upper-class district in northern Tehran.

So, a fine-tuned model on the NER task has extracted the relationship between words so that it can understand the differences of a Named Entity based on the neighboring words. A similar case happens in the punctuation prediction task, too.

- Sentence 3: First, we have to understand the problem.
- Sentence 4: He was the first runner to finish the tour.

The word "First" is used in both of these sentences, but in Sentence 3, it has a comma after it, and in Sentence 4, we do not have any punctuation mark after it.

A network that is pre-trained on a NER task knows to extract the relationship between the words. We can use this beneficial information through parameters in the model as our base network to converge to the correct answer fast.

To the best of our knowledge, this is the first time that anyone is looking at the punctuation prediction task in this way and trying to use a pre-trained model on NER as their base work. We think this will result in stronger answers in a smaller amount of time.

3.3 BLSTM on BERT

For our last model, we used BERT output as a representation of words and used the according word representations for the input of BLSTM. In other words, we input the sentences into BERT and get a 768-dimension output for each word from the latent layer of BERT. After that, we use these embeddings as input for the BLSTM model. The main reason for using BERT before BLSTM is that in this way we can use information from the relation between the neighboring words more. Consequently, we will reduce the error rate of the model because the model is context-aware. Figure 4 shows the design of this model. Using this model was in the "Future Work" of our presentation; however, we decided to implement it in order to have more models to compare the results. And of course, it was fun! To the best of our knowledge, this is the first work that uses a RNN on top of a transformer model for the punctuation prediction task.

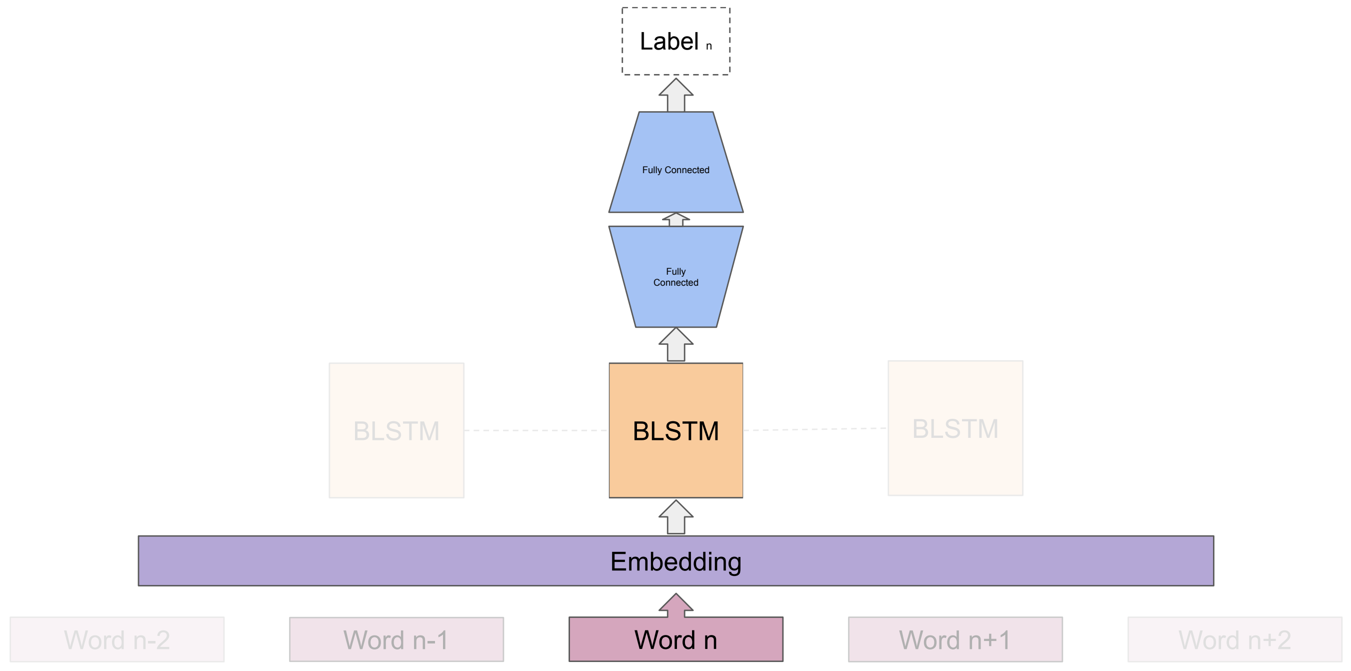


Figure 2: BLSTM Model Architecture. In each training step, 200 words are added as we define our batch size to be 200. For each word in the batch, embedding transforms the word into a vector in a 300d vector space. Then it goes to a bi-directional LSTM layer with an output size of 32 * 2 (since it is bi-directional). Then it enters a fully connected layer with an output size of 96. After that, another fully connected layer reduces the vector to a 7d vector. Each index in the vector represents the probability of each label for the input word that has been normalized using a Softmax function. The highest probability will be chosen. The training loss is a weighted negative log likelihood loss (NLLLoss). Weights are based on the frequency of each label (the less frequent, the more important). All the details are available in model's jupyter notebook file.

4 Experiments

In this section, we include the results of our experiments and how we did them. We also discuss the dataset we used and preprocessing steps in addition to the running environment.

4.1 Dataset

We used WikiText103V1 (Merity et al. [2016]) dataset. Before this, the Penn TreeBank (PTB) (Marcus et al. [1993]) was the commonly used dataset. However, it had some limitations. The first problem, is that PTB tokens are all lower case. The second, is the lack of punctuation in the dataset. Finally, the vocabulary size is only 10k words which makes it unrealistic for real language. These were the motivations behind creating the WikiText103V1 dataset.

WikiText103V1 is built using only *Good* or *Featured* articles in Wikipedia. These article were reviewed by humans which resulted in 23,805 Good and 4,790 Featured articles. After gathering the articles, a few processing steps were implemented to create a usefull dataset. This resulted in a dataset with over 103 million words which is a hundred times larger than PTB.

The size information of each subset of data is shown in table 1.

Subset	Size
Train	1801350
Validation	3760
Test	4358

Table 1: Size of each subset of dataset

4.2 Data Preparation

Our goal in the preprocessing step is to label each word in every sentence such that each label declares whether the current token is followed by a punctuation. After labeling the word we can train a model over the dataset to predict where, which punctuation is needed.

At first, we tokenize the paragraphs and remove the ones with less than 5 or more than 510 tokens. Following, we assign

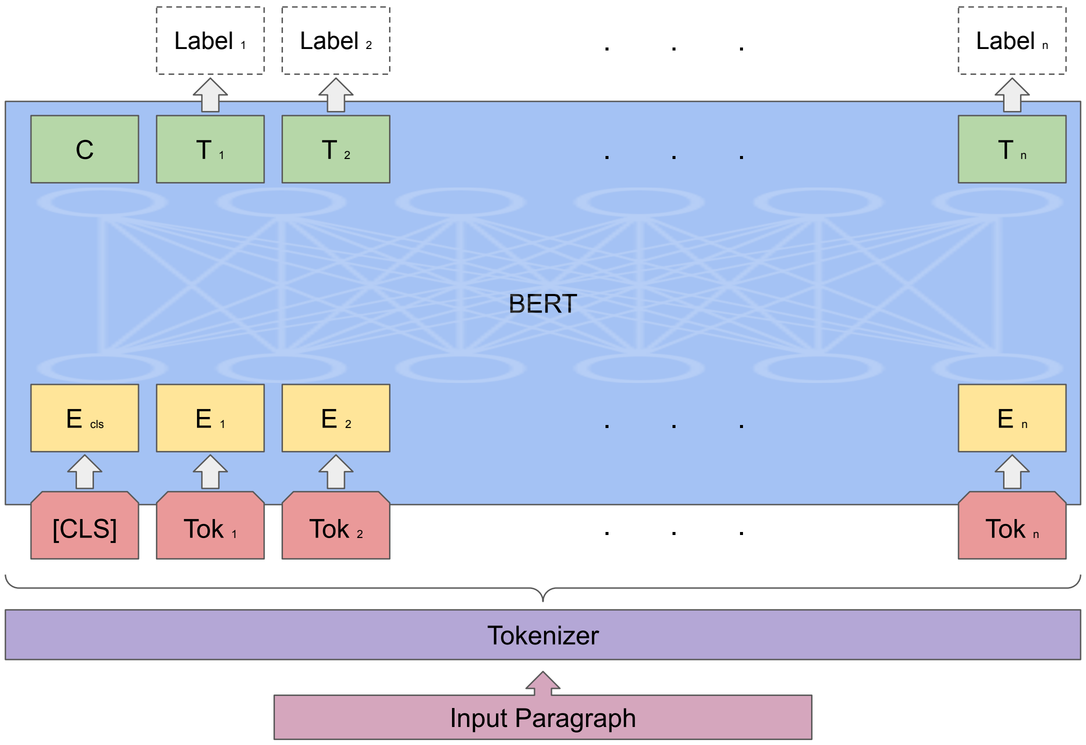


Figure 3: BERT Model architecture. In each training step, the tokenizer passes 512 tokens to the pre-trained BERT model. The pre-training had done fine-tuning on the NER task before. Each paragraph may not meet 512 tokens; the tokenizer will increase the tokens using padding but set the attention input to the tokens that really contain words. Then tokens are input into the model. After the vectors pass through the encoders and decoders, the output will be ready. All the details are available in model's jupyter notebook file.

the labels for each word. Our labels consist of 7 different tags which are demonstrated in table 2. For example, in the sentence "I am a student.", the according labels for each word are "O O O DOT". In another word, after the word "student" we should have a dot.

Punctuation	Label
,	COMMA
.	DOT
?	QMARK
!	EMARK
:	COLON
;	SEMICOLON
Other tokens	O

Table 2: Punctuation and their according label

For real examples from the dataset we can see the figure 5.

After checking the subsets of the dataset, we found that the validation and test set do not have the same distribution of punctuations as train set. This is shown in table 3. Hence, we decided to divide the dataset into 10 bins and choose one bin as test set and the rest as train set. The distribution of labels in each of the bins is demonstrated in table 4.

4.3 Running

First, we used Google Colab Pro. However, it took so long for each epoch to finish, and it got restarted once in a while, which caused the running environment to get deleted, so we had to change our platform. After that, we used a machine with NVIDIA GeForce 3090 with CUDA 11.04.

We run the jupyter notebooks in the order described in Section A. We trained all the models on the first 9 bins of the dataset and tested them on the 10th bin. We trained the BERT and BLSTM on BERT models only one epoch, and the BLSTM model 4 epochs.

4.4 Results

The final results are presented in 6, 7, 8. The diagrams have been created using the jupyter notebooks attached to this report. We have normalized the confusion matrices so that the color in the diagram feels the same in each row, regardless of the support count of each label. For example, label *O* has the

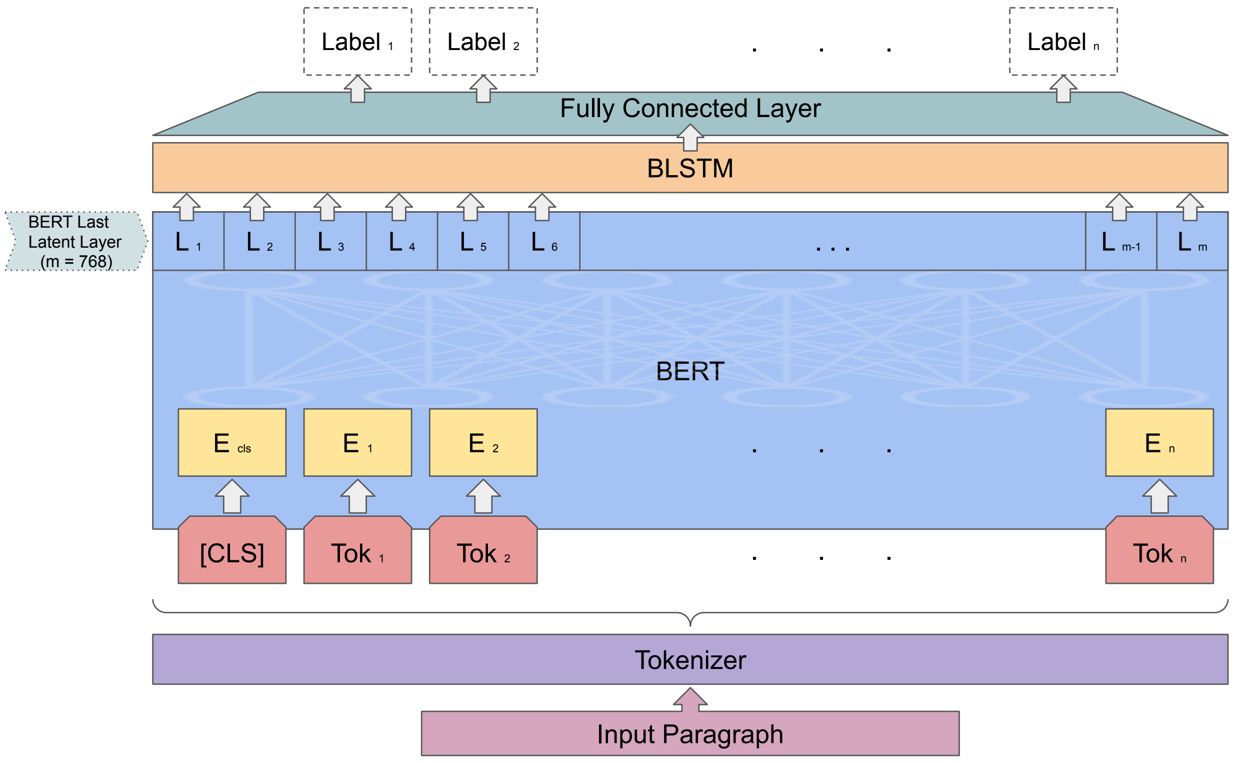


Figure 4: BLSTM on BERT Model architecture. The first section of this model is just like the BERT model in figure 3. But, before going to the output layer, the last hidden layer with a size of 768 will be input into a BLSTM. BLSTM output size is 512*2 because it is bi-directional. Then, it inputs to a fully connected layer to output a vector with size 7 for each label. Each index in the output vector represents the probability for each label, and the highest probability will be chosen as the prediction. To the best of our knowledge, this is the first work that uses a RNN on top of a transformer model for the punctuation prediction task. All the details are available in model's jupyter notebook file.

Subset	'O'	'COMMA'	'DOT'	'SEMICOLON'	'COLON'	'EMARK'	'QMARK'
Train	81100448	4930180	3593348	188808	172093	14566	10922
Test	194821	11091	8891	547	310	20	12
Validation	173456	10068	7759	348	272	26	17

Table 3: Distribution of labels in each subset of dataset.

most support in the dataset and can easily dominate the color bar in the diagram.

By looking at the figures 6 and 7, the BERT model shows an obvious superiority since the diagram is more diagonal. Accordingly, BERT has overall better precision, recall, and f1-score. Of course, BERT also has flaws, for example, in the exclamation mark or *EMARK* where the model predicts no punctuation (*O*) or *DOT* in most cases.

Comparing figures 7 and 8, again, BERT shows a slightly better performance than BLSTM on BERT model. We take it as a good sign since adding a BLSTM to a pre-trained BERT model does not degrade the performance that much. With this model, we put extra attention on the relation of the neighboring words using BLSTM. Also, it can prove that sometimes this extra attention can be useful. For example, predicting the question mark label (*QMARK*) gets better with this model. These close results made us think about designing further analysis, and if we had enough time we would have gone through the experiments. For instance, we could try text with more specialized texts like medical applications. In these kinds of texts where the contexts are close in each sentence, there are lots of repetitive patterns that can help the model easily predict the punctuation based on similar cases. But still, this is a theory and needs further investigation and proof.

Tables 6, 7, 8, and 5 show the comparison of our models with the model presented in Salloum et al. [2017b]. Our BLSTM model is very similar to that model, and we created that because we wanted to have a fair comparison since we do not have access to the closed-source medical dataset they used in Salloum et al. [2017b]. Also, they only used three

Bin #	'O'	'COMMA'	'DOT'	'SEMICOLON'	'COLON'	'EMARK'	'QMARK'
bin 1	8029053	484574	355423	18788	16875	1400	1078
bin 2	8101003	492828	358899	19375	17563	1565	1075
bin 3	8144042	498427	357869	19356	17082	1579	1132
bin 4	8041108	490299	357069	17637	16794	1421	1071
bin 5	8222611	499524	365307	19440	17389	1516	1184
bin 6	8052719	488164	357975	18996	17364	1599	1083
bin 7	8162914	495072	362724	19265	17398	1376	1046
bin 8	8130355	495434	360526	19014	17404	1362	1154
bin 9	8075951	489160	359203	18203	17173	1245	1022
bin 10	8140692	496698	358353	18734	17051	1503	1077

Table 4: Distribution of labels in each bin.

punctuation marks to predict them with their model (our DOT equals their PERIOD). Another point is that we have to keep in mind that when a text is specialized, like medical text, the sentence structures get similar, and this helps the model to make better decisions. So, probably, BERT will perform better in specialized texts rather than texts in Wikipedia, where there are lots of sentences with a broad range of topics. We can see that although their model performs better in their dataset (that their support count is not available), our BERT model outperforms the BLSTM model, which is similar to their model in our dataset.

Here we will compare the results of our models with the work of Salloum et al. [2017b]. We can see the comparison in tables 5 6 7 8.

Punctuation	Precision	Recall	F-Score
COLON	98.6%	98.6%	98.6%
COMMA	84.0%	82.2%	83.1%
PERIOD	96.1%	96.4%	96.3%
Overall	94.2%	94.0%	94.1%

Table 5: Metrics of Salloum et al. [2017b]

Punctuation	Precision	Recall	F-Score	Support
COMMA	0.25%	0.51%	0.34%	496702
DOT	0.64%	0.73%	0.68%	358354
QMARK	0.0%	0.0%	0.0%	1077
EMARK	0.0%	0.0%	0.0%	1503
COLON	0.0%	0.0%	0.0%	17051
SEMICOLON	0.0%	0.0%	0.0%	18734
O	0.96%	0.90%	0.92%	8140717
Macro Avg	0.26%	0.30%	0.28%	9034138
Weight Avg	0.90%	0.86%	0.88%	9034138

Table 6: Metrics of BLSTM model

Speaking of training time, BERT wins the competition here again. Each bin of our dataset took about 10 minutes for BERT to train with our GPU setting. This was about 30 minutes for BLSTM and BLSTM on BERT models. With

Punctuation	Precision	Recall	F-Score	Support
COMMA	0.80%	0.77%	0.79%	496702
DOT	0.93%	0.94%	0.93%	358354
QMARK	0.71%	0.48%	0.57%	1077
EMARK	0.52%	0.05%	0.10%	1503
COLON	0.73%	0.48%	0.58%	17051
SEMICOLON	0.71%	0.51%	0.59%	18734
O	0.98%	0.98%	0.98%	8140717
Macro Avg	0.77%	0.60%	0.65%	9034138
Weight Avg	0.97%	0.97%	0.97%	9034138

Table 7: Metrics of BERT model

Punctuation	Precision	Recall	F-Score	Support
COMMA	0.81%	0.75%	0.78%	496698
DOT	0.94%	0.93%	0.94%	358353
QMARK	0.72%	0.51%	0.60%	1077
EMARK	0.64%	0.03%	0.07%	1503
COLON	0.75%	0.40%	0.53%	17051
SEMICOLON	0.75%	0.50%	0.60%	18734
O	0.98%	0.98%	0.98%	8140692
Macro Avg	0.80%	0.59%	0.64%	9034108
Weight Avg	0.97%	0.97%	0.97%	9034108

Table 8: Metrics of BLSTM on BERT model

the difference that we trained BLSTM model for 4 epochs, while the other models were trained only for 1 epoch. A quick trainable model is extremely advantageous in research like this, where the researchers are new to the field and have to deal with lots of problems all at once and in a limited time.

5 Conclusion

In this paper, we have designed three models and trained them on the dataset. For a fair test phase, we divided train set into ten bins and chose the new training set to be the first nine bins and the new test to be the last bin. Because the word count in the test and validation split of the Wikitext103V1 (Merity et al. [2016]) is not enough for a fair comparison. We wanted to investigate the transfer learning in punctuation prediction and check if we can use it for the punctuation prediction task using the NER task and successfully pass the test.

We used BERT model to capture the context of the text. Additionally, we could use state-of-the-art models to improve the results. However, due to the way we implemented the code, the model can be easily substituted without much change. On the other hand, BERT can only accept 512 input tokens; this problem is also present for BLSTM models based on their batch size. Another problem is that we should input the whole sentence at once into this model. This is not practical in real-time speech-to-text applications since they get the input incrementally.

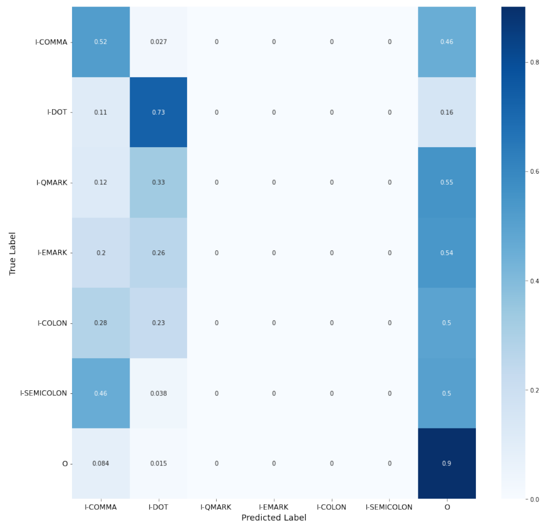


Figure 6: Normalized confusion diagram of the results for BLSTM model

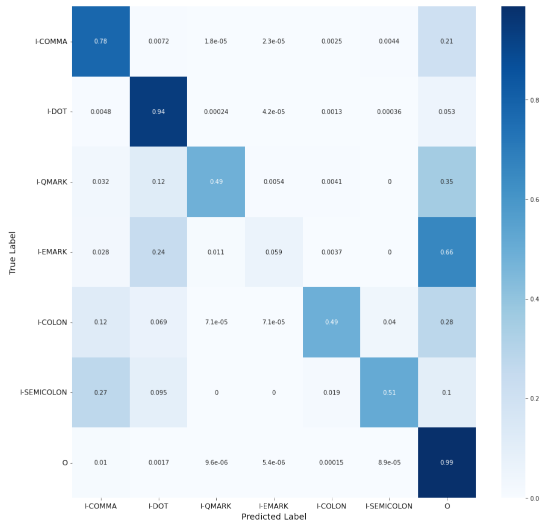


Figure 7: Normalized confusion diagram of the results for BERT model

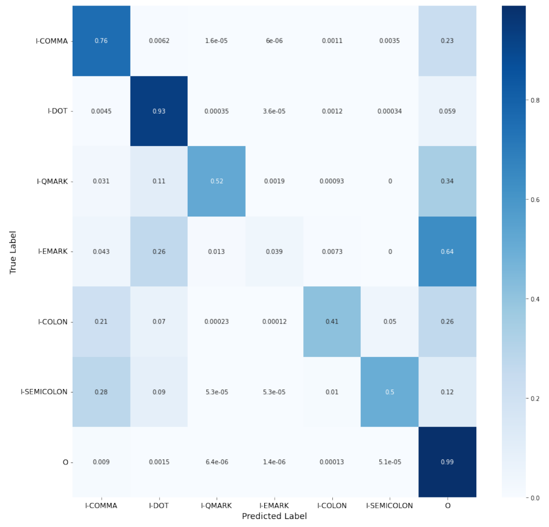
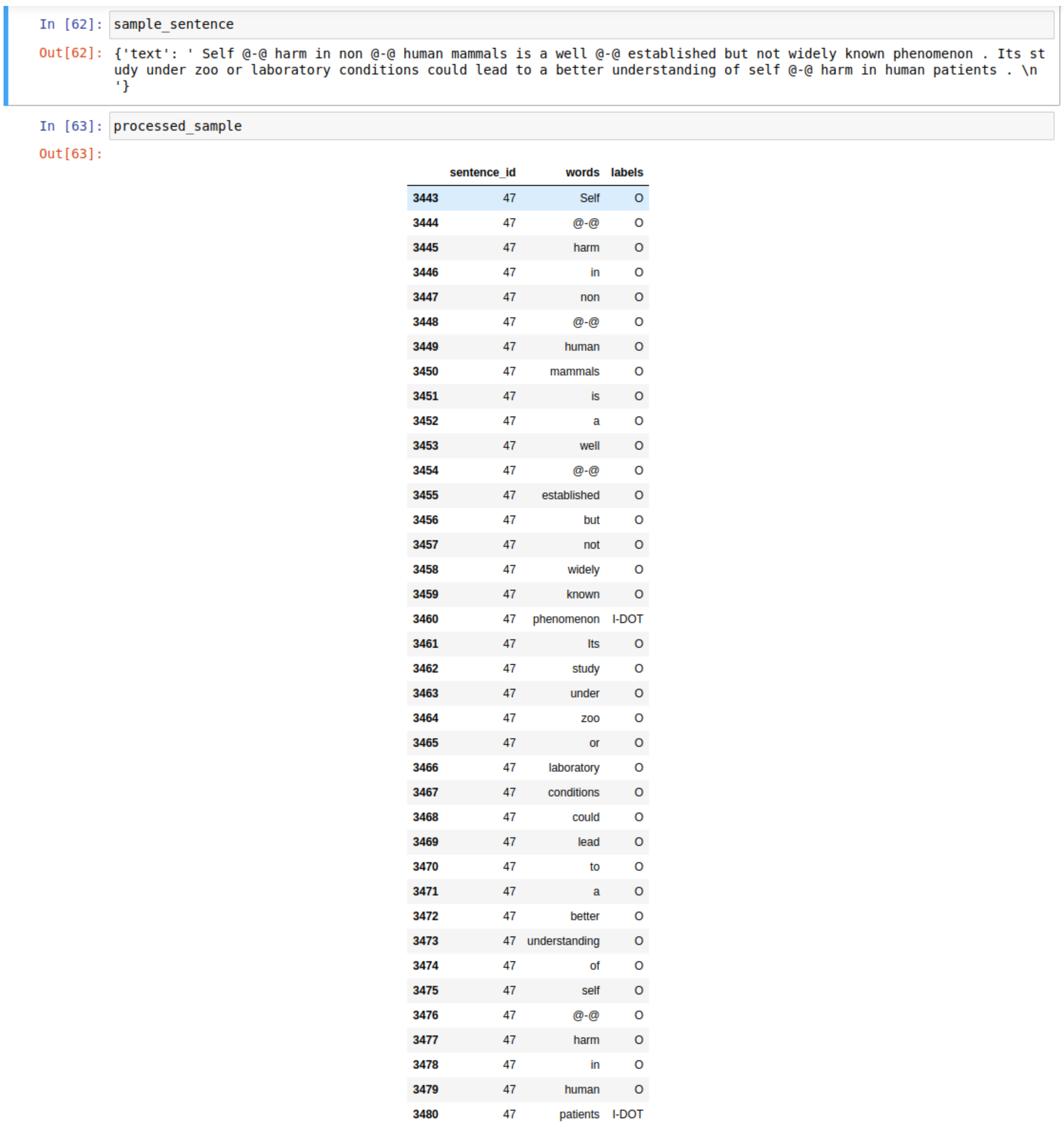


Figure 8: Normalized confusion diagram of the results for BLSTM on BERT model

6 Acknowledgement

We want to thank professor An for allowing us to do such an interesting project. We are both master's students, and NLP is not our specialty in research at all. Yet we had a wonderful time learning during this research project; the experience was priceless.



```
In [62]: sample_sentence
Out[62]: {'text': ' Self @-@ harm in non @-@ human mammals is a well @-@ established but not widely known phenomenon . Its st
         udy under zoo or laboratory conditions could lead to a better understanding of self @-@ harm in human patients . \n
         '}
```

```
In [63]: processed_sample
Out[63]:
```

	sentence_id	words	labels
3443	47	Self	O
3444	47	@-@	O
3445	47	harm	O
3446	47	in	O
3447	47	non	O
3448	47	@-@	O
3449	47	human	O
3450	47	mammals	O
3451	47	is	O
3452	47	a	O
3453	47	well	O
3454	47	@-@	O
3455	47	established	O
3456	47	but	O
3457	47	not	O
3458	47	widely	O
3459	47	known	O
3460	47	phenomenon	I-DOT
3461	47	Its	O
3462	47	study	O
3463	47	under	O
3464	47	zoo	O
3465	47	or	O
3466	47	laboratory	O
3467	47	conditions	O
3468	47	could	O
3469	47	lead	O
3470	47	to	O
3471	47	a	O
3472	47	better	O
3473	47	understanding	O
3474	47	of	O
3475	47	self	O
3476	47	@-@	O
3477	47	harm	O
3478	47	in	O
3479	47	human	O
3480	47	patients	I-DOT

Figure 5: Example of data presented in dataset. "I-" prefix is for turning off the warnings of NER model in programming, and it is not important. "I-DOT" means "DOT".

References

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

Yu Zhang, Mengdong Chen, and Lianzhong Liu. A review on text mining. In *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 681–685, 2015. doi: 10.1109/ICSESS.2015.7339149.

Said Salloum, Mostafa Al-Emran, Azza Monem, and Khaled Shaalan. A survey of text mining in social media: Facebook and twitter perspectives. *Advances in Science, Technology and Engineering Systems Journal*, 2: 127–133, 01 2017a. doi: 10.25046/aj020115.

Mariette Awad and Rahul Khanna. *Hidden Markov Model*, pages 81–104. 01 2015. ISBN 978-1-4302-5989-3. doi: 10.1007/978-1-4302-5990-9_5.

John Lafferty, Andrew Mccallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289, 01 2001.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71, mar 1996. ISSN 0891-2017.

Theodoros Evgeniou and Massimiliano Pontil. Support vector machines: Theory and applications. volume 2049, pages 249–257, 09 2001. ISBN 978-3-540-42490-1. doi: 10.1007/3-540-44673-7_12.

Vikramkumar, Vijaykumar B, and Trilochan. Bayes and naive bayes classifier, 2014. URL https://arxiv.org/abs/1404.0933.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, jan 2015. doi: 10.1016/j.neunet.2014.09.003. URL https://doi.org/10.1016%2Fj.neunet.2014.09.003.

Wahab Khan, Ali Daud, Jamal Nasir, and Tehmina Amjad. A survey on machine learning models for natural language processing (nlp). 43:95–113, 10 2016.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL https://arxiv.org/abs/1810.04805.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Vikas Yadav and Steven Bethard. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-1182.

Monica Sunkara, Srikanth Ronanki, Kalpit Dixit, Sravan Bodapati, and Katrin Kirchhoff. Robust prediction of punctuation and truecasing for medical asr, 2020. URL https://arxiv.org/abs/2007.02025.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, sep 2019. doi: 10.1093/bioinformatics/btz682. URL https://doi.org/10.1093%2Fbioinformatics%2Fbtz682.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Eunah Cho, Jan Niehues, and Alex Waibel. Nmt-based segmentation and punctuation insertion for real-time spoken language translation. pages 2645–2649, 08 2017. doi: 10.21437/Interspeech.2017-1320.

Wael Salloum, Greg Finley, Erik Edwards, Mark Miller, and David Suendermann-Oeft. Deep learning for punctuation restoration in medical reports. In *BioNLP 2017*, pages 159–164, Vancouver, Canada,, August 2017b. Association for Computational Linguistics. doi: 10.18653/v1/W17-2319. URL https://aclanthology.org/W17-2319.

M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. doi: 10.1109/78.650093.

Piotr Żelasko, Piotr Szymański, Jan Mizgajski, Adrian Szymczak, Yishay Carmiel, and Najim Dehak. Punctuation prediction model for conversational speech, 2018. URL https://arxiv.org/abs/1807.00543.

Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015. URL https://arxiv.org/abs/1511.08458.

Chin Char Juin, Richard Xiong Jun Wei, Luis Fernando D'Haro, and Rafael E. Banchs. Punctuation prediction using a bidirectional recurrent neural network with part-of-speech tagging. In *TENCON 2017 - 2017 IEEE Region 10 Conference*, pages 1806–1811, 2017. doi: 10.1109/TENCON.2017.8228151.

Subodh Malgonde. Transfer learning using tensorflow, 2018. URL https://medium.com/@subodh.malgonde/transfer-learning-using-tensorflow-52a4f6bcde3e.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning, 2019. URL https://arxiv.org/abs/1911.02685.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information, 2016. URL https://arxiv.org/abs/1607.04606.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, jun 1993. ISSN 0891-2017.

A Appendix

First, to run and get the same results as we did, you have to prepare the dataset. For this, please follow the jupyter notebook attached with the name: data_preprocess.ipynb

You do not need to download the dataset; the code itself will do. Then, for each model, you can follow its corresponding jupyter notebook. blstm_model.ipynb for the BLSTM model, bert_model.ipynb for the BERT model, and blstm_on_bert_model.ipynb for the BLSTM on BERT model described in this paper. A minor note to consider is that for running blstm_model.ipynb, you have to download the word vectors binary model for English language ($./cc.en.300.bin$) from this website and unzip and put it in the same directory of blstm_model.ipynb.

You also can test your punctuation-less texts with the BERT model in bert_model_test.ipynb! Please follow the jupyter notebook to see the results. You can change the texts in the "Testing on any wanted sentence" section with your own sentences.

Other than these, you can find our models in the "$models/$" directory submitted in the zip file, and you can read them to validate our results.

If you face any difficulty setting up the environment, you can contact us through email, or even if you cannot set it up, we can run it on our local computer for you.