Text Classification with Misspelling

Tian Xia
ACM Honour Class
Shanghai Jiao Tong University
xia_tian@sjtu.edu.cn

Abstract

Informal language, such as spelling or grammarly mistake, often appears in our daily life. For huamns, by context, we are able to detect these errors and auto-correct it to get the intended meaning. However, in Natural Language Processing area, some state-of-the-art model might be fooled by some simple mistake. This paper, inspired by the pretrain idea widely used in Deep Learning area, proposed a way to decrease the impact brought by spelling mistakes (or typos).

1 Introduction

Language in daily use often comes with some kinds of informality, such as grammarly mistake and spelling errors (or typos). As humans, we have the ability to auto-correct these small mistake and understand the original meaning of the sentence. However, dealing with these informality in daily used language is still challenging for Deep Learning, in specific, Natural Language Processing (NLP) area. A small mistake of text could greatly impact the result of a state-of-the-art natural language processing model. For example, Sentiment Analysis is one classical task of NLP area, which assign a sentence or document an appropriate category (sentiment positive or negative). In one of the widely-used model Transformers' [6] demo we get the following result:

```
>>> from transformers import pipeline
>>> classifier = pipeline('sentiment-analysis')
>>> classifier('I am satisfied')
[{'label': 'POSITIVE', 'score': 0.9996809959411621}]
>>> classifier('I am astisfied')
[{'label': 'NEGATIVE', 'score': 0.9479582905769348}]
```

Figure 1: A sample of misspelling

In this example, a small spelling mistake from "satisfied" to "astisfied" makes the result completely different. However, NLP model for daily use will inevitably encounter informal sentences, which will decreace accuracy and precision of result.

In NLP area, pretrain is a common method to save training resources. If different model shares same structure, by using previous trained model's weight directly, a decent accuracy could be reached within limited training resources (e.g. GPU and time). This paper will focuses on spelling mistakes (or typos) mainly, since this is one of the simplest informality in language and is often occur when typing in our daily life. The rest of paper researches pretrain model in different dataset, and offers a solution to the informal sentence input.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

2 Datasets

In this paper we use a subset of IMDB and AG news, which originally contains 127,600 entry in AG news and 50,000 entry in IMDB. AG news provides a map from raw news to 4 category: World, Sports, Business and Sci/Tech; IMDB map movie review to 2 category: positive or negative. 40,000 in each dataset for train and 7,600 for test is taken in one experiment. Beside from that, We made some typos in original dataset by the following method:

```
def typo(word):            # core typo function
  if len(word) <= 2:
    return word[0] + word
  type = random.randint(0, 5)
  pos = random.randint(0, len(word) - 2)
  if type == 0: # duplicate
    return word[:pos] + word[pos] + word[pos:]
  elif type == 1: # delete
    return word[:pos] + word[pos + 1:]
  else: # swap
    return word[:pos] + word[pos + 1] + word[pos] + word[pos + 2:]
```

In which we introduce a typo for each original text. Randomly choose a word, and 3 common types of typos is possible (all example is from raw data used in experiment):

(i) 1/6 possibility of duplicate a random letter, for example, *has* $\rightarrow$ *haas*.

(ii) 1/6 possibility of delete a random letter, for example, *kept* $\rightarrow$ *kep*.

(iii) 2/3 possibility of swap two neighbour letter, for example, *could* $\rightarrow$ *colud*.

In different experiments we use dataset contain or not contain separately. Notice that in traning dataset we keep all original data and in test dataset we only keep the typo version. Meanwhile, we only use AG news as test dataset since IMDB have plenty of training text which is oversize than BERT Model's input threshold. In implementation, we directly truncate it to a fixed max length, however, this will affect its reliability for test dataset. Details is shown in the Experiment section.

3 Related Work

After Transformer's [6] release, NLP area step into a inflection point. Following BERT [3] Model outperform Transformer and leading the whole area in a long time. These state-of-the-art model enjoy high performance on traditional NLP tasks such as Machine Translation and Text Classification, however, might be fooled with simple informal language phenomenon like "astisfied" we have seen before.

Some effort were put into traditional NLP Model previous than Transformer area. A context-based noisy channel model [2] were promoted by Church and Gale. N-gram Model [1] were used in error correction too.

Researchers also focused on model based on neural network. A Sequence to Sequence (seq2seq) model [4] is trained to correct grammatical error, and a special encode of error inputs were used in [5].

4 Method

4.1 Text Classification

For Text Classification task, we use a BERT [3] tokenizer to tokenize input words to tensors, pad and truncate all input to same fixed size and generate attention masks.

Then a BERT base pretrained model, which consist of 12 transformer layers, is used to identify a input sequence. In detail, the model take in a list of token embeddings and produce the same number of embeddings with the same hidden size on the output, and we take the output of final transformer layer of the [CLS] token as the features of the whole input sequence.

After that we use a 3-hidden-layer feed-forward neural network as classifier.

A visualization of the structure of our model looks like:

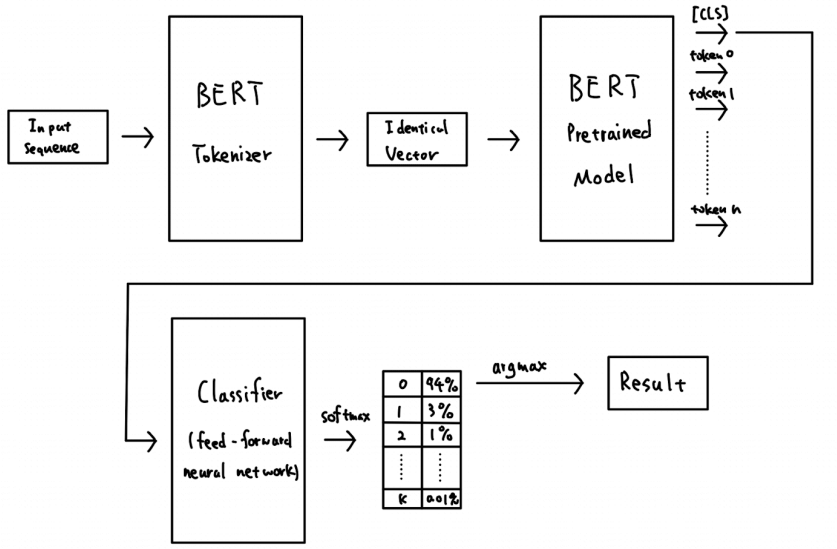

Figure 2: A visualization of our model

4.2 Typo correct

The main idea is train model on two different dataset. To implement this, when traning on two different dataset, we only change the shape of the last output layer, and all hidden layer as well as pretrained BERT model is same. Therefore we could load previous trained weight as initialize value of neural network, and train as usual. In our source code of classifier the structure is clearly explained (only keep important part):

```
class BertClassifier(nn.Module):
  def __init__(self, dataset):
    din, in1, in2, h, dout = 768, 300, 100, 50, 2 if dataset == 'imdb' else 4
    self.bert = BertModel.from_pretrained('bert-base-uncased')
    self.inner0 = nn.Sequential(nn.Linear(din, in1), nn.ReLU())
    self.inner1 = nn.Sequential(nn.Linear(in1, in2), nn.ReLU())
    self.inner2 = nn.Sequential(nn.Linear(in2, h), nn.ReLU())
    self.classifier = nn.Linear(h, dout)
```

In Experiment section, we will load different number of layer from pretrained and figure out how many layers from pretrained will output the best result.

5 Experiment and Result

5.1 Experiment Argument

Our experiment is taken in 5.1

5.2 Training on single dataset

First we train on AG news for 4 epoch. Result is shown in figure 5, in which blue line represent training dataset contains typos 1 and green line contains no typos. We can see after 3 epoch the accuracy is non-increasing either with or without typos, therefore simply scale up train only contribute to limited performance increment. Therefore, a new way of training is needed if we want to reach higher accuracy.

Entry	Environment
python	3.8.10
PyTorch	1.10.0
CUDA	11.3
Transformers	4.19.2
Hardware	NVIDIA GeForce RTX 3090

Figure 3: Experiment Environment

Hyper Arguments	Value
Learning Rate	5×10^{-5}
Adam's Epsilon	1×10^{-8}
Batch Size	16
Warmup Ratio	0.1
Default Epoch	2

Figure 4: Experiment Hyper Arguments

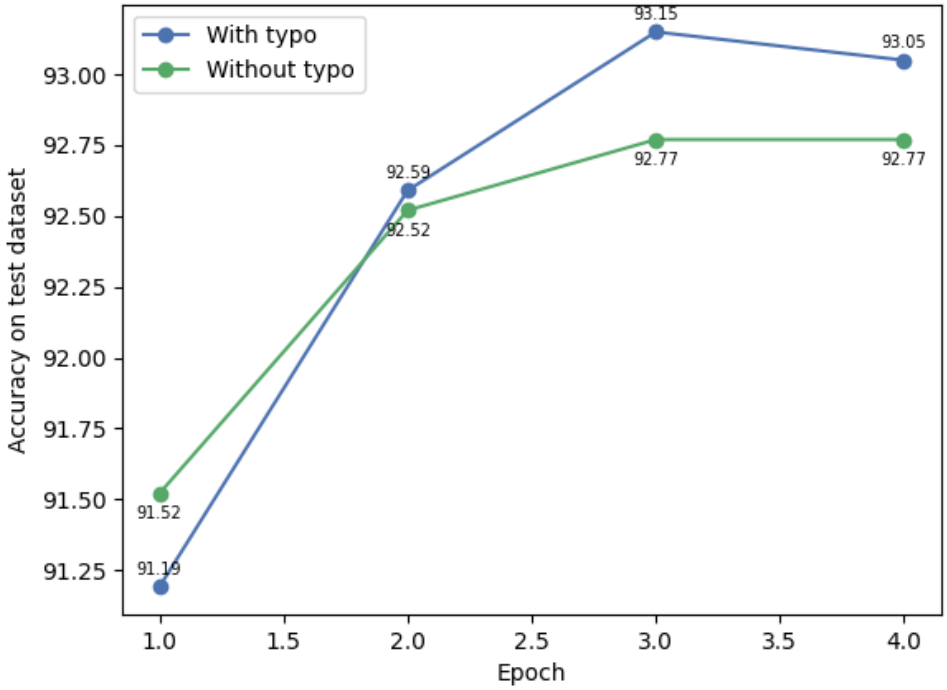

Figure 5: Result for 4 epoch in AG news

5.3 Pretrained on Different Dataset

Then we pretrain in IMDB dataset for 2 epoch, and train another 2 epoch on AG news. In the plot's legend, "From Pretrained Number" (hereinafter called FPN) stands for number of layers loaded from pretrained model, details in the following tabular 1.

Table 1: From Pretrained Number's meaning

FPN	Load from Pretrained
≥ 1	BERT Model
≥ 2	Hidden Layer 0
≥ 3	Hidden Layer 1
≥ 4	Hidden Layer 2

For example, FPN $= 2$ means load BERT Model and Hidden layer 0 from pretrained, left Hidden Layer 1 & 2 and classifier layer to randomly initialized.

The result is shown in the following graph 6, in which both of them reached a higher accuracy than 4 epoch on AG news dataset. However, the peak of performance is not unified. Pretrain with typo

reach its peak when only inherit the BERT Model, and pretrain without typo reach its peak when inherit BERT Model and hidden layer 0 & 1.

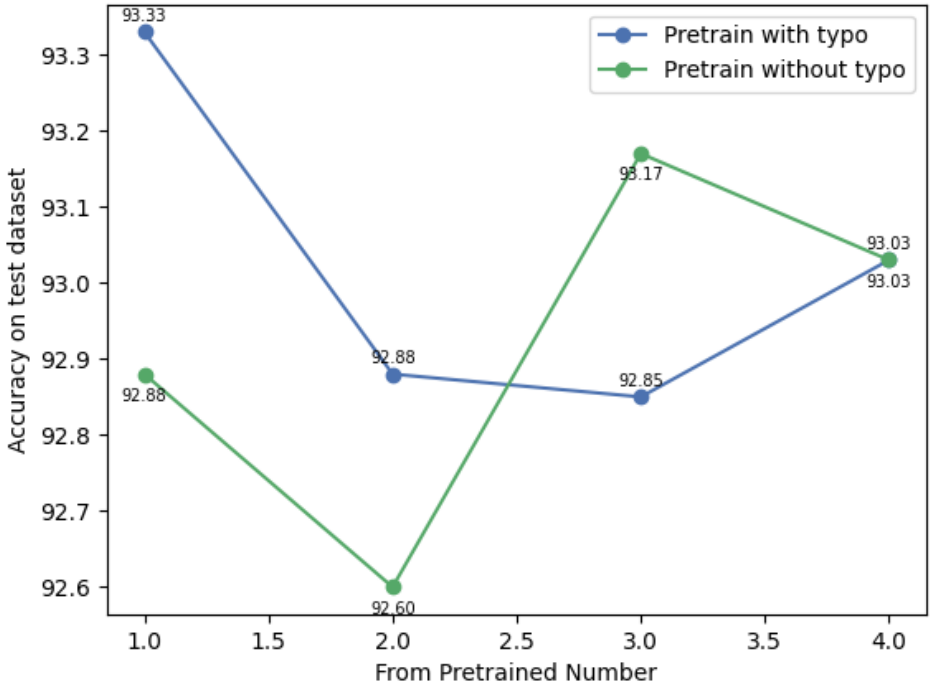

Figure 6: Result for AG news pretrained on IMDB dataset

6 Conclusion

6.1 Overfitting

From figure 5, we can see after 3 epoch, the accuracy is non-increasing in both cases. This might due to our Model overfitting on train dataset, and therefore perform worse on test dataset.

6.2 Pretrain on Different Dataset

Figure 6 displays the peak performance of pretrain on different dataset, in both case the performance is better than simply train 4 epoch on single dataset. This might own to the overfitting problem, which training on different dataset effectively avoid this problem; or our model studied on more inputs, learn more information about language. This is similar to expand the size of dataset.

6.3 Peak Performance

In pretrain with typo, the peak performance appear when only inherit the BERT Model, which might due to understanding typos is mainly done by the BERT Model; and in pretrain without typo, its peak appear when inherit BERT Model and hidden layer 0 & 1, maybe because sentiment analysis share some common classify standard with news classifixation, for example, news about Sports is more likely to be positive.

7 Future Work

There are still some aspect could be explore in future work. This paper only focus on how many layer inherit from pretrained model, however, other hyper parameters also have influence on the result. Also, the method current model dealing with oversize input is to direct truncate and abandon the rest part, which might have some impact on traning and evaluate our performance.

References

[1] Andrew Carlson and Ian Fette. Memory-based context-sensitive spelling correction at web scale. In *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pages 166–171. IEEE, 2007.

[2] Kenneth W Church and William A Gale. Probability scoring for spelling correction. *Statistics and Computing*, 1(2):93–103, 1991.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Tao Ge, Furu Wei, and Ming Zhou. Reaching human-level performance in automatic grammatical error correction: An empirical study. *arXiv preprint arXiv:1807.01270*, 2018.

[5] Yifei Hu, Xiaonan Jing, Youlim Ko, and Julia Taylor Rayz. Misspelling correction with pre-trained contextual language model. In *2020 IEEE 19th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pages 144–149. IEEE, 2020.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.