Noisy Text Data: Achilles' Heel of BERT

Ankit Kumar, Piyush Makhija and Anuj Gupta

Vahan Inc

Abstract

- We focus on how BERT performs when fine-tuned on noisy text.
- We show that presence of noise in text data leads to significant degradation in BERT's performance.
- We identify shortcomings in the existing BERT pipeline that are responsible for this drop in performance.

Introduction

Given BERT's phenomenal success on various NLP tasks and their benchmark datasets, machine learning practitioners in industry are actively experimenting with fine-tuning BERT to build NLP applications for solving industry use cases. For most datasets used by practitioners to build industrial NLP applications, it is hard to guarantee absence of any noise in the data. While BERT has performed exceedingly well for transferring the learnings from one use case to another, it remains unclear how BERT performs when fine-tuned on noisy text. In this work, we explore the sensitivity of BERT to noise in the data.

We work with most common noise (spelling mistakes, typos) and show that it results in significant degradation in BERT's performance. We present experimental results to show that BERT's performance on fundamental NLP tasks like sentiment analysis and textual similarity drops significantly in the presence of (simulated) noise on benchmark datasets viz. IMDB Movie Review, STS-B, SST-2.

Further, we identify shortcomings in the existing BERT pipeline that are responsible for this drop in performance. Our findings suggest that machine learning teams in industrial settings need to be vary of presence of noise in their datasets while fine-tuning BERT in order to solve industry use cases.

Experimental setup

From the original dataset D_0 we create new datasets $D_{2.5}$, D_5, ... $D_{22.5}$. Here, D_k is a variant of D_0 with k% noise in each datapoint in D_0. To create $x_{i,k}{}^{noise}$ from x_i, we randomly choose k% characters from the text of x_i and replace them with nearby characters in a qwerty keyboard. We inject noise in the complete dataset. Later we split D_i into *train* and *test* chunks.

Results

	SA		TS
% error	**IMDB**	**SST-2**	**STS-B**
0.0	0.93	0.89	0.89
2.5	0.85	0.86	0.84
5.0	0.79	0.80	0.75
7.5	0.67	0.76	0.65
10.0	0.62	0.70	0.65
12.5	0.53	0.67	0.49
15.0	0.51	0.60	0.40
17.5	0.46	0.59	0.39
20.0	0.44	0.54	0.29
22.5	0.41	0.49	0.31

Table 1:Results of experiments on both clean and noisy data. SA = Sentiment Analysis; TS = Textual Similarity

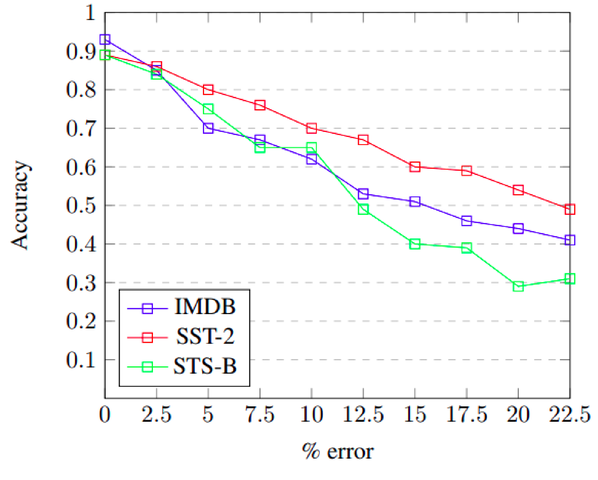

Figure 1:Accuracy vs Noise

Analysis

To understand the reason behind the drop in BERT's performance in presence of noise, we need to understand how BERT processes input text data. A key component of BERT's pipeline is the tokenization of input text. When tokenizing noisy text data, we see an interesting behaviour from BERT's pipeline.

Example 1 (from IMDB):
(a) that loves its characters and communicates something rather beautiful about human nature. (0% error)
(b) that loves 8ts characters abd communicates something rathee beautiful about human natuee. (5% error)

Corresponding output of tokenization:
(a) 'that', 'loves', 'its', 'characters', 'and', 'communicate', '##s', 'something', 'rather', 'beautiful', 'about', 'human', 'nature'
(b) 'that', 'loves', '8', '##ts', 'characters', 'abd', 'communicate', '##s', 'something', 'rat', '##hee', 'beautiful', 'about', 'human', 'nat', '##ue', '##e'

Example 2 (from STS-2):
(a) poor ben bratt could n't find stardom if mapquest emailed him point-to-point driving directions. (0% error)
(b) poor ben bratt could n't find stardom if mapquest emailed him point-to-point drivibg dirsctioje. (5% error)

Corresponding Output of tokenization:
(a) 'poor', 'ben', 'brat', '##t', 'could', 'n', '", 't', 'find', 'star', '##dom', 'if', 'map', '##quest', 'email', '##ed', 'him', 'point', '-', 'to', '-', 'point', 'driving', 'directions', '.'
(b) 'poor', 'ben', 'brat', '##t', 'could', 'n', '", 't', 'find', 'star', '##dom', 'if', 'map', '##quest', 'email', '##ed', 'him', 'point', '-', 'to', '-', 'point', 'dr', '##iv', '##ib', '##g', 'dir', '##sc', '##ti', '##oge', '.'

Conclusion

We studied the effect of synthetic noise (spelling mistakes) on the performance of BERT. We show that as the amount of noise increases, its performance drops drastically. The reason for this drop is how BERT's tokenizer (WordPiece) handles the misspelled words.

Future Work

- Our results suggest that one needs to conduct a large number of experiments to see if the findings hold across other popular NLP tasks such as information extraction, text summarization, machine translation, question answering, etc.
- One also needs to investigate how other models such as ELMo, RoBERTa, and XLNet which use character-based, byte-level BPE, and SentencePiece tokenizers respectively.
- It will also be interesting to see how BERT performs in presence of other types of noise.

Contact Information

Email & LinkedIn:

- ankit@vahan.co /ankit-ahlawat
- piyush@vahan.co, /piyushmakhija
- anuj@vahan.co, /anujgupta-pnlp

vahan

Y Combinator