# Exploring Sentiment Analysis by Fine-tuning Pre-trained Transformers

**Shicheng Wen**
wenshich@usc.edu

## Abstract

In this study, we focused on fine-tuning various pre-trained language models and assessing their performance across multiple sentiment analysis datasets. Additionally, we explored the impact of dataset augmentation and conducted evaluations on transformed datasets. Our findings suggest that the bidirectional Transformer structured pre-trained language model performed excellent on sentiment analysis tasks. Furthermore, dataset augmentation enhances the model's robustness and effectiveness, particularly in handling out-of-distribution scenarios encountered during inference. This improvement underscores the value of incorporating diverse and challenging data variations in the training process to better prepare models for real-world applications.

## 1 Introduction

In recent times, the evolution of pre-trained large language models has markedly advanced the benchmarks in natural language understanding tasks. Fine-tuning technology is playing a crucial role in this progress. These pre-trained language models, upon fine-tuning, have demonstrated superior capabilities in a spectrum of downstream tasks, notably including sentiment analysis.

In our study, we apply fine-tuning to various pre-trained language models such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), GPT2 (Radford et al., 2019), XLNet (Yang et al., 2019), and ELECTRA (Clark et al., 2020) through sentiment analysis datasets like IMDB (Maas et al., 2011)[1], Amazon Polarity (McAuley and Leskovec, 2013, Zhang et al., 2015a)[2], Yelp Polarity (Zhang et al., 2015b)[3], and Rotten Tomatoes (Pang and Lee, 2005)[4] and evaluated them. We aim to show the effectiveness of these pre-trained models in sentiment analysis after fine-tuning.

Furthermore, our work extends to assessing the performance of the language models that are fine-tuned using augmented datasets in facing challenges during inference. These challenges incorporate modifications such as synonym substitutions, typographical errors, capitalization mistakes, adjacent word exchanges, and word deletions at the inference stage.

Our findings reveal that pre-trained large language models not only excel in downstream natural language understanding tasks like sentiment analysis but also exhibit amplified improvement following data augmentation. This underscores the robustness and adaptability of pre-trained models and fine-tuning methodologies in handling complex linguistic tasks.

## 2 Methodology

### 2.1 Pre-trained Language Models

In our work, we utilized five transformer-based pre-trained language models as foundational baselines.

**BERT** (Devlin et al., 2019) is an encoder-only model for natural language understanding, rooted in the Transformer architecture. It is pre-trained on a large corpus of English data using masked language modeling (MLM) and next sentence prediction (NSP) objectives in a self-supervised fashion.

**RoBERTa** (Liu et al., 2019) is an optimized version of BERT. RoBERTa enhances performance by refining the pre-training process, which involves using larger datasets and longer training, removing the NSP objective, and using dynamic token masking settings.

**GPT-2** (Radford et al., 2019) is a decoder-only model based on the Transformer framework. It

[1]`https://huggingface.co/datasets/imdb`
[2]`https://huggingface.co/datasets/amazon_polarity`
[3]`https://huggingface.co/datasets/yelp_polarity`
[4]`https://huggingface.co/datasets/rotten_tomatoes`

leverages a causal language modeling (CLM) pre-training objective, which involves predicting the subsequent word in a text sequence.

**XLNet** (Yang et al., 2019) uses a permutation-based language model that enables the learning of bidirectional context without the limitations of masked tokens in MLM. Combining the strengths of both autoregressive and autoencoding techniques, the unique approach allows XLNet to capture a more nuanced understanding of language context and sequence relationships.

**ELECTRA** (Clark et al., 2020) introduces a novel pre-training approach named Replaced Token Detection (RTD). This strategy involves a smaller generator network that substitutes some words in the text. A larger discriminator network is then trained to identify which words have been replaced. This method enhances the efficiency of the training process.

### 2.2 Data Transformation and Augmentation

We transformed the test set to simulate out-of-distribution scenarios that might arise during inference. Similarly, we sampled a subset of the training set with 5000 samples for data augmentation. The transformation or augmentation includes operations such as synonym substitutions, transformation to typographical errors, conversion to all lowercase, adjacent word exchanges, and deletion of random words.

For synonym substitutions, we utilized the NLTK's (Bird et al., 2009) averaged perception tagger[5] to identify the part-of-speech (POS) tag for a targeted token. Subsequently, we selected the first synonym sharing the same POS tag from WordNet[6]. This method is likely to ensure that the substituted synonym maintains a similar contextual meaning.

In creating typographical errors, we randomly chose either one character or a pair of adjacent characters in a word and replaced them with either a neighboring character on the QWERTY keyboard or with characters that have a phonetic resemblance, thereby creating a 'synophone'. For example, the word "movie" might be altered to "mocie", reflecting a common keyboard misstroke between 'v' and 'c'. Similarly, "phase" could be changed to "fase", given the phonetic similarity between 'ph' and 'f'. Detailed rules for these substitutions are elaborated in the appendix A.

For each text sample for transformation or augmentation, 60% of non-punctuation and non-stop-word tokens are masked for transformation. Among these tokens, 25% of them will be replaced by their synonyms, 25% of them will be transformed to lower-cased with typos, 25% of them will be swapped with an adjacent word, and 25% of them will be deleted. These operations are done sequentially, but the probabilities are independent and identically distributed.

## 3 Experiment

### 3.1 Datasets

We fine-tuned and evaluated the pre-trained models on 5 different sentiment analysis datasets.

**IMDB** (Maas et al., 2011) is a large movie review dataset for binary sentiment classification, with a 25K training split and a 25K test split.

**Amazon polarity** (McAuley and Leskovec, 2013, Zhang et al., 2015a) is a large reviews dataset consisting of reviews from Amazon. The binary labels are constructed by taking review scores 1 and 2 as negative, and 4 and 5 as positive. Samples of score 3 are ignored. The training split includes 3.6M samples, and the test split includes 400K samples. Due to the limitation of computational resources, we sampled 25K from each split in balance in this work.

**Yelp polarity** (Zhang et al., 2015b) is a Yelp review dataset extracted from the Yelp Dataset Challenge 2015 data. The labels are constructed by considering stars 1 and 2 negative, and 3 and 4 positive. The original dataset includes 560K training samples and 38K test samples. We sampled 25K from each split in balance in this work.

**Rotten Tomatoes** (Pang and Lee, 2005) is a movie review dataset for binary sentiment classification containing 5.3K positive and 5.3K negative processed sentences from Rotten Tomatoes movie reviews. There are 8.53K training samples and 1.07K samples each for validation set and test set. We merged the validation set and test set to a larger test set in this work.

### 3.2 Implementation Details

We deployed the models using PyTorch (Paszke et al., 2019)[7] and the pre-trained weights are loaded from Transformers by HuggingFace (Wolf et al.,

[5]https://www.nltk.org/_modules/nltk/tag/perceptron.html

[6]https://www.nltk.org/_modules/nltk/tag/perceptron.html

[7]https://pytorch.org/

2019). Each model is trained on an NVIDIA V100 Tensor Core GPU. The maximum sequence length for all models is set to 512. We used AdamW as the optimizer. The initial learning rate is set to $5 \times 10^{-5}$ and will decrease over the training process by a linear learning rate scheduler. The mini-batch size for each update is set as 8 for BERT, GPT2, ELECTRA, and 16 for RoBERTa and XLNet. The model is trained for up to 3 epochs. We used the EOS token as the padding token for GPT2 and left padding with respect to GPT2's autoregressive nature. The pre-trained weights we used, imported from HuggingFace Transformers, are BERT-base-uncased (110M), RoBERTa-base (125M), GPT2 (124M), XLNet-base-cased (110M), and ELECTRA-base-discriminator (110M).

Fine-tuning on all the original datasets is deployed for all models. Evaluation of the transformed datasets and training on augmented datasets are used on the BERT model.

## 4 Results

### 4.1 Main Results

Table 1 shows the results of the model performance trained and evaluated on the original train or test set. Notably, RoBERTa and ELECTRA emerged as the top performers in all datasets, while GPT2 consistently exhibited the lowest accuracy.

The BERT model and its derivatives, including RoBERTa, XLNet, and ELECTRA, are distinguished by their ability to comprehend bidirectional context, which is integral in capturing the comprehensive content information of sentences. This capability renders them highly effective in natural language understanding tasks like sentiment analysis. The bidirectional nature of these models allows them to incorporate context from both sides of a token in the input sequence, enhancing their understanding and interpretation of sentence structures and meanings.

GPT-2, in contrast, is limited by its unidirectional, autoregressive nature. This design constrains the model to only use previous tokens for context, hindering its ability to fully grasp the complete information of a sentence. This characteristic can be a drawback in tasks requiring a holistic understanding of sentence content.

RoBERTa, in particular, excels in semantic understanding due to its extensive pre-training on larger datasets and a more refined pre-training strategy. This results in improved model performance, especially in tasks involving complex semantic interpretations.

ELECTRA, with its discriminator model, achieves higher efficiency in training and enhanced performance in binary classification tasks. Its unique approach of distinguishing replaced tokens in the input sequence allows for more refined model tuning and contributes to its effectiveness in understanding nuances in language.

Moreover, the dynamic masking in RoBERTa and the token discrimination task in ELECTRA's pre-training further augment these models' capabilities in semantic comprehension. These features enable the models to better understand and interpret the intricacies of language, leading to superior performance in tasks requiring deep linguistic insights.

### 4.2 Data Augmentation

| **Training Set** | **Dataset** | | | |
|---|---|---|---|---|
| | **IMDB** | **Amazon** | **Yelp** | **Rotten** |
| Original | 0.85 | 0.5 | 0.503 | 0.488 |
| Augmented | 0.877 | 0.87 | 0.877 | 0.759 |

Table 2: Fine-tuned **BERT** model performance in accuracy on transformed datasets. **Amazon** stands for Amazon Polarity dataset, **Yelp** stands for Yelp Polarity dataset, and **Rotten** stands for Rotten Tomatoes dataset.

Table 2 shows the results of the BERT performance evaluated on the transformed test set, which trained on the original training set or the augmented training set. The findings from the study indicate a notable reduction in the performance of the BERT model when fine-tuned on the original training set and tested on the transformed dataset. Specifically, the accuracy of BERT, post fine-tuning on the original dataset, decreased by 7.9% on the IMDB dataset. On other datasets like Amazon, Yelp, and Rotten Tomatoes, the performance dropped to around 50%, suggesting a significant diminution in learning efficacy. However, the introduction of data augmentation markedly improved the model's performance.

A key factor contributing to this improvement is BERT's utilization of the WordPiece (Wu et al., 2016) tokenization method. This approach allows the model to break down words into smaller segments, or even individual characters, which proves particularly useful when encountering out-of-distribution (OOD) words or typographical errors in the enhanced dataset. The typographical er-

| Model | Dataset | | | |
|---|---|---|---|---|
| | **IMDB** | **Amazon Polarity** | **Yelp Polarity** | **Rotten Tomatoes** |
| **BERT** | 0.929 | 0.951 | 0.928 | 0.837 |
| **RoBERTa** | **0.953** | **0.973** | 0.951 | 0.879 |
| **GPT2** | 0.887 | 0.923 | 0.883 | 0.746 |
| **XLNet** | 0.932 | 0.969 | 0.944 | 0.859 |
| **ELECTRA** | 0.952 | 0.97 | **0.953** | **0.893** |

Table 1: Model performance in accuracy on IMDB, Amazon Polarity, Yelp Polarity, and Rotten Tomatoes

ror substitution rule employed in our study involves replacing small fragments of words, aligning well with BERT's tokenization strategy.

As BERT can learn contextual information from these small fragments, it becomes better equipped to handle and predict accurately during inference time, even in datasets with transformed data. This ability for granular understanding and adaptation enables BERT to perform effectively in datasets that have undergone such data enhancements, showcasing the model's resilience and adaptability to diverse linguistic challenges.

## 5 Conclusion

In this work, we demonstrated the superior performance of fine-tuned pre-trained large language models in various downstream tasks. A key finding was the comparative advantage of the bidirectional Transformer structure, as seen in models like BERT, RoBERTa, XLNet, and ELECTRA, over the unidirectional autoregressive structure of the Transformer, exemplified by GPT-2, in natural language understanding tasks. This distinction highlights the importance of the bidirectional context in effectively processing and interpreting language.

Moreover, we highlighted the significant impact of dataset enhancement on model performance, particularly in scenarios involving out-of-distribution (OOD) challenges, such as typographical errors and synonym replacements. The augmentation of datasets with these transformations proved to be a critical factor in improving the models' robustness and adaptability. By incorporating such variations in the training data, the models were better equipped to handle and accurately respond to unconventional or altered inputs, which are common in real-world applications.

## 6 Limitations and Future Work

One limitation of our current study was the restricted application of data augmentation testing due to computing resource limitations. In the future, extending this testing to all models discussed in our research is imperative to gain a comprehensive understanding of their performance under augmented conditions.

A critical aspect to consider is that, while our study demonstrated the effectiveness of data augmentation in addressing out-of-distribution (OOD) scenarios during inference, our approach was based on heuristic rules that may not encompass the full spectrum of real-world linguistic variations. For instance, typographical errors can be more varied than those covered in our study, and synonyms may not always be perfectly interchangeable in meaning. Moreover, the consistency between our test set transformation rules and dataset augmentation strategies raises concerns about potential data leakage, a scenario unlikely to occur in real-world applications. To address these issues, future research should aim to diversify dataset augmentation rules, incorporating a broader range of linguistic variations and typographical anomalies. Furthermore, introducing more randomness into the transformation of the test set could help in better simulating the unpredictability of real-world data, thus enhancing the robustness and applicability of our findings. These steps will contribute to the development of language models that are more resilient and accurate in handling the complexities of natural language processing.

## References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on*

*Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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. *CoRR*, abs/1907.11692.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, page 165–172, New York, NY, USA. Association for Computing Machinery.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015a. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015b. Character-level Convolutional Networks for Text Classification. *arXiv:1509.01626 [cs]*.

## A Typographical Error Substitution Rules

Table 3 and 4 are heuristic typo and synophone substitution rules for single characters and double characters.

| Original | Substitution |
|---|---|
| a | s, q, w, z, x, u, ei |
| b | v, g, h, n |
| c | x, d, f, v, s, ck, sc |
| d | e, r, s, f, x, c, t |
| e | w, r, s, d, ee, ea, i, ei |
| f | r, t, d, g, c, v, ph, gh |
| g | t, y, f, h, v, b, j |
| h | g, j, b, n, y, u |
| i | u, o, j, k, e, ee, ea, ai, ay, ie, y |
| j | u, i, h, k, n, m, g |
| k | i, o, j, l, m, ch, ck |
| l | o, p, k |
| m | n, j, k |
| n | b, g, h, j, m |
| o | i, p, k, l, oa, ou, oh, ow, au, ao |
| p | o, l |
| q | w, a |
| r | e, t, d, f |
| s | a, d, w, x, e, z, th, c, sc |
| t | r, y, f, g, d |
| u | y, i, h, j, a, oo |
| v | c, f, g, b |
| w | q, e, a, s |
| x | z, c, s, d |
| y | t, u, g, h, i, ie, ey |
| z | a, s, x, th |

Table 3: Single Character Substitution

| Original | Substitution |
|---|---|
| ai | ay, i |
| ao | o, au, oa, ou, ow |
| au | o, ao, oa, ou, ow |
| ay | ai |
| ch | sh, tr, tch, c |
| ck | c, k, ch |
| ea | i, e, ee, ie |
| ee | i, e, ea, ie |
| ei | a, e, ie |
| gh | f |
| ie | i, y, ea, ee, ei |
| ph | f |
| mb | m |
| oa | o, ou, ow, ao, au |
| ou | o, oa, ow, ao, au |
| ow | o, oa, ou, ao, au |
| oo | u, o |
| sc | s |
| sh | ch, s |
| ti | sh |
| th | s |
| tt | t |
| tr | ch |
| wr | r |
| wh | w |

Table 4: Double Character Substitution