Data Augmentation for Natural Language Inference

Tianyuan Qiu
520030910137
frank_qiu@sjtu.edu.cn

Abstract

This paper proposes enhancing the performance of the BERT base model for Natural Language Inference (NLI) through data augmentation techniques. The study focuses on generating additional training samples by modifying the original dataset using strategies like synonym and antonym substitution. Experiments on a benchmark NLI dataset demonstrate that data augmentation significantly improves the accuracy of the BERT base model. The findings highlight the potential of data augmentation in enhancing NLI models and contribute to research on NLP and data augmentation. The problems reflected in the results are also discussed and directions for further improvement are proposed. Code related to the study is publicly available at https://github.com/PaperL/Toy_NLI

1 Introduction

1.1 Natural Language Inference

The semantic concepts of entailment and contradiction are central to all aspects of natural language meaning, from the lexicon to the content of entire texts. Natural Language Inference (NLI) is a task that aims to determine the logical relationship between two given sentences, which can be categorized as entailment, contradiction, or neutrality. This task is of significant importance in various natural language processing applications, including question answering, summarization, and sentiment analysis. To successfully perform NLI, a comprehensive understanding of semantic and syntactic structures is necessary in order to make accurate inferences about the relationship between the provided sentences.(Bowman et al., 2015)

1.2 Data Augmentation

Data augmentation is a technique used to expand the training dataset by generating additional samples through various modifications or transformations applied to the original data. (Feng et al., 2021) In the context of NLI, data augmentation can involve techniques such as synonym substitution, paraphrasing, or back-translation. By introducing diverse examples, data augmentation aims to enhance the model's generalization ability, improve robustness, and mitigate overfitting, ultimately boosting the performance of NLI models.

1.3 BERT Model

The BERT (Bidirectional Encoder Representations from Transformers) model is a state-of-the-art pretrained language model that utilizes Transformer architecture to capture contextual information from text.(Devlin et al., 2018) BERT has achieved remarkable success in various NLP tasks due to its ability to learn bidirectional representations of words. It has been widely adopted as a baseline model for many NLP applications, including the NLI task.

2 Data Augmentation for NLP Task

Data plays a crucial role in achieving high performance in machine learning models, and the more data we have, the better results we can obtain. However, manually annotating a large amount of training data can be a luxury in terms of time and resources. To overcome this challenge, data augmentation techniques come into play, offering a way to enhance model performance by generating synthetic training data. While data augmentation is widely popular in the field of computer vision, applying it to natural language processing (NLP) tasks poses unique challenges due to the complexity of language.

In computer vision, augmenting images is relatively straightforward. Techniques such as flipping, adding noise, or cropping can be easily applied using libraries like imgaug, leading to improved performance in vision models. Augmentation has been proven to be a key factor in the success of computer vision models.

However, augmenting text in NLP is more intricate. Not every word can be easily replaced with alternatives like "a," "an," or "the," and not every word has a direct synonym. Moreover, even a slight change in a word can drastically alter the context and meaning of a sentence. Unlike images, where introducing noise or cropping may still allow the model to classify the image correctly, the same level of simplicity does not apply to text augmentation.

Given the limitation of manually creating an unlimited amount of training data, researchers have explored various methods to generate more data for model training.

2.1 Thesaurus

Zhang et al. found that one of the useful ways to do text augmentation is by replacing words or phrases with their synonyms.(Zhang et al., 2015) Leveraging existing thesaurus help to generate lots of data in a short time. Zhang et al. select a word and replace it with synonyms according to the geometric distribution.

2.2 Word Embeddings

Wang and Yang introduced a data augmentation approach for automatic categorization of annoying behaviors.(Wang and Yang, 2015) They proposed two methods for finding similar words for replacement. The first method involved utilizing k-nearest-neighbor (KNN) and cosine similarity to identify similar words. As an alternative, they also explored the use of pre-trained classic word embeddings such as word2vec, GloVe, and fasttext to perform similarity searches. These techniques aimed to enhance the data augmentation process and improve the accuracy of the automatic categorization model.

2.3 Back Translation

In the field of machine translation, back-translation has been employed as a data augmentation technique to address the issue of limited training data for certain language pairs. Sennrich et al. utilized back-translation to augment the training data for low-resource translation models.(Sennrich et al., 2015) By translating the target language sentences back to the source language and combining them with the original source sentences, the number of training examples from the source language to the target language could be significantly increased, thereby improving the performance of the translation model.

2.4 Contextualized Word Embeddings

Fadaee et al. introduced the use of contextualized word embeddings as a form of text augmentation in the context of low-resource neural machine translation.(Fadaee et al., 2017) Instead of relying on static word embeddings, they leveraged contextualized word embeddings to replace target words in the translation process. This approach, known as Translation Data Augmentation (TDA), demonstrated improvements in the machine translation model's performance by leveraging the power of contextualized embeddings and is further confirmed in other work.(Kobayashi, 2018)

2.5 Text Generation

Kafle et al. proposed an alternative approach to text augmentation for visual question answering by generating augmented data through text generation techniques.(Kafle et al., 2017) They presented two approaches: template augmentation and LSTM-based question generation. Template augmentation involved using pre-defined question templates that could be paired with rule-based answers, while the LSTM-based approach utilized image features to generate questions. By generating complete sentences instead of replacing individual words, Kafle et al. aimed to enhance the diversity and quality of the augmented data, contributing to improved performance in visual question answering tasks.

3 Experiment

The experimental environment is *WSL2 (Ubuntu 20.04) under Windows 10*, using *python3* and *pyTorch*. Specific code and package requirements is attached.

3.1 Data Preprocessing

In summary, the data preprocessing pipeline reads the input data, extracts relevant tokens, augments the texts by replacing adjectives and verbs with synonyms and antonyms, generates new augmented examples. Then determine the proper token length according to the dataset. And finally saves the preprocessed data for follow-up use in text classification tasks using deep learning models.

3.1.1 Word Augmentation

The objective of preprocessing is to augment the existing data by replacing adjectives and verbs in the input texts and generate new data samples for improved training.

First, data are read from a JSONL file and the `spacy` library is utilized to extract adjectives and verbs from a given sentence.[1] These extracted adjectives and verbs serve as the targets for augmentation. To modify these words while preserving the correctness of the example, the `nlpaug` library is employed.[2] The augmenters in `nlpaug` are based on the `nltk` toolchain. The tense is maintained using the `en_core_web_sm` model provided by `spacy`. The tense of words is indicated by the tags generated by the `spacy` model, and any undesirable augmented data is discarded.

The `SynonymAug` augmenter replaces words with similar meanings while keeping the label unchanged. On the other hand, the `AntonymAug` augmenter replaces words with opposite meanings. Several situations can arise: if the label of the example is *neutral*, reversing word meanings does not have a significant impact; if the label is *entailment*, reversing word meanings should result in a contradiction; if the label is *contradiction*, reversing a random number of words introduces uncertainty to the label, as the reversal may lead to either less conflict (entailment), more conflict (contradiction), or a broken relationship (neutral).

After the replacement process, the program checks if the augmented texts are different from the originals and constructs new examples accordingly, because augmenter sometimes returns the same words. If a synonym-replaced text is different, it creates an augmented example and appends it to the augmented data list. Antonym-replaced text is checked similarly. In addition, the program utilizes parallel processing with a `Pool` to efficiently process the data examples in parallel.

3.1.2 Tokenization

Tokenization is a fundamental process that involves splitting a sequence of text into individual tokens or words. In our study, we analyzed and determined the length of tokens in the given dataset to facilitate the tokenization process.

To ensure compatibility with the PyTorch model's input data requirement of having the same length, we needed to establish a maximum token length. This decision was based on an analysis of the distribution of sequence lengths within our dataset. We employed the `AutoTokenizer` from the `transformers` library to examine the token lengths of the text pairs. By calculating the sequence lengths for each text pair, we determined the maximum token length that would encompass a greater number of examples, particularly those with shorter lengths, thereby reducing computational complexity.

For the tokenization process, we utilized the `BertTokenizer` provided by the `transformers` library. This tokenizer converts the input text into a suitable format for the BERT model. We performed preprocessing on both the training and test datasets using the `preprocess_dataset` function. The sequences were padded to the maximum token length, and the resulting tokenized inputs included the `input_ids`, `attention_mask`, and `token_type_ids`.

To conform to the data format required by the deep learning model, we quantized the label information. In this quantization scheme, the label "0" represents *entailment*, "1" represents *contradiction*, and "2" represents *neutral*. Prior research has demonstrated the effectiveness of this quantification approach, and the order of the labels does not impact the results (Chen, 2015).

Practically, tokenization results in a data size increase of less than 10 times. Since the text data itself occupies relatively less space, preprocessing the data beforehand rather than during model execution can significantly improve training efficiency.

3.2 Model Training

Experimental Platform: *Ubuntu 20.04, Cuda 11.3, PyTorch 1.11.0, Python 3.8*. Experimental device: one *A100-PCIE-40GB*. Each epoch performs training using all the data. Batch size is set to 32, and the *AdamW* optimizer is used with a learning rate of 2e-5. The loss function used is *CrossEntropyLoss*. No additional components were added except for Bert Base uncased.

[1] https://spacy.io/

[2] https://github.com/makcedward/nlpaug

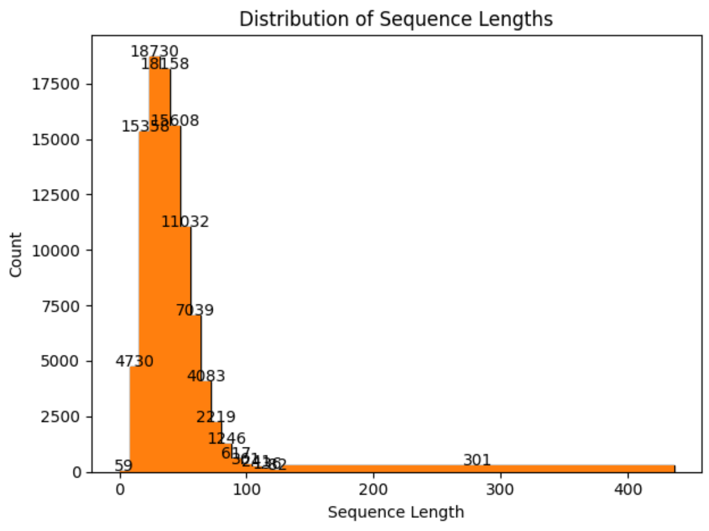

Figure 1: Distribution of sequence lengths of given dataset using tokenizer from Bert.

4 Result and Review

4.1 Dataset Preprocessing

As shown in the above figure, texts longer than 128 tokens account for less than 0.3% of all texts. Therefore, during the tokenization process, the max length is set to 128, and any text exceeding this limit is truncated. The loss of these truncated contents does not have a perceptible negative impact on the training results.

In addition, a total of 92,567 new samples were generated, effectively doubling the available dataset. Here are some examples (original text and augmented text):

```
SYNONYM:
i think we might exist capable to assist.
i think we might be able to help.
SYNONYM:
a launch to say let's preserve to a greater
extent because he never said that
a launch to say let's conserve more because
he never said that
ANTONYM:
well it's been very nasty talking to you
well it's been very nice talking to you
ANTONYM:
i think we might be unable to help.
i think we might be able to help.
```

5 Model Evaluation

The author conducted experiments using the unenhanced dataset and the Bert Base uncased model as the baseline. Compare the training process with an augmented dataset to observe and analyze the impact of data augmentation on the Natural Language Inference task.

In both the baseline and data augmentation experiments, the initial loss was above 0.8 and eventually converged to around 0.6. The training with the augmented dataset showed significantly better results compared to the original dataset. The comparison was based on the decrease in loss for the same batch size. After training on 1 million samples (approximately 10 epochs for the baseline and 5 epochs for data augmentation), the augmented model approached convergence with a loss below 0.65, while the baseline model remained around 0.7.

Furthermore, when training on 3 million samples (approximately 30 epochs for the baseline and 15 epochs for data augmentation), both models achieved relatively stable losses. Evaluation using a test dataset showed an improvement of approximately 0.6% in accuracy for the data augmentation approach.

6 Review

This study demonstrates the positive impact of data augmentation on improving model performance by comparing the training of the same pretrained model on augmented and original datasets for NLI tasks. However, the evaluation of model performance and the results of manually curated datasets suggest that the effectiveness of data augmentation in the NLP field still needs improvement when using non-massive models such as GPT-3. In this work, multiple commonly used NLP tools were employed, revealing two common deficiencies. Firstly, during the process of word replacement, particularly with verbs, it is challenging for the processing algorithm to ensure the preservation of tense and voice in the replacement results. This may be attributed to the lack of harmonious analysis at the sentence level in the replacement program. Secondly, in terms of changes involving synonyms and antonyms, the use of deep learning models results in a high error rate. This might be due to the inability of deep learning tasks' datasets to accurately reflect the lexical meanings of words, leading to a higher number of errors during word-level replacement. To address this issue, it may be necessary to strengthen the application of traditional algorithms that focus on dictionary processing.

Acknowledgements

This is the Course Project of the 2023 NLU Course taught by Professor Hai Zhao, with Yao Yao serving as the teaching assistant. I would like to express our sincere gratitude to Professor Zhao for his incredibly insightful lectures and to the teaching assistant for assigning excellent assignments and providing meticulous feedback. I also extend our thanks to our classmates for their valuable discussions.

References

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Yahui Chen. 2015. Convolutional neural network for sentence classification. Master's thesis, University of Waterloo.

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

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. *arXiv preprint arXiv:1705.00440*.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*.

Kushal Kafle, Mohammed Yousefhussien, and Christopher Kanan. 2017. Data augmentation for visual question answering. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 198–202.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. *arXiv preprint arXiv:1805.06201*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2557–2563.

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

A Appendix

Content of `https://github.com/PaperL/Toy_NLI` is attached.