Sentiment Analysis and Text Classification on Twitter Dataset

Chung Him, Tony KWOK

April 23, 2023

1 Introduction

Sentiment Analysis has been a challenging task in Natural Language Processing (NLP) as it involves information extraction based on the model understanding of certain texts. In this project, we attempt to tackle a text classification problem in this field with several different approaches. In particular, we work on Task 4, Subtask A of Semeval challenge 2017 [Rosenthal et al., 2019], which is classifying the sentiment of a particular tweet as 'Positive', 'Neutral', or 'Negative'. We evaluate our results on three different test datasets based on the macro-averaged F1 score on the 'Positive' and 'Negative' classes.

2 Non-neural Classification Models

In this section, we focus on non-neural models. In particular, we train i) K-nearest neighbors Classifiers, ii) Multinomial Naive Bayes Classifiers (not applicable for Sentence Embedding), and iii) Multinomial Logistic Regression models, based on two different word vectorization approaches, namely Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer [Ramos, 1999], and Sentence Embedding [Reimers and Gurevych, 2019].

2.1 Bag-of-Words Approach: TF-IDF Vectorization

Unlike images, which can be vectorized as pixels in a straightforward manner, there are many different methods to convert texts into vector inputs for model training. Traditional Bag-Of-Words (BoW) approaches, like TF-IDF Vectorizer, used to be de facto before the advent of word embedding methods. We first build our first baseline models with TF-IDF vectorizer. The main principle of TF-IDF is to encode the words from the texts based on term frequency and the inverse of document frequency and create a large, sparse matrix.

2.1.1 Word Preprocessing

For word preprocessing, since TF-IDF only concerns term frequency in documents, we lowercase all letters, and remove URLs, user mentions, hashtags, special characters, and punctuations. We attempted both Stemming and Lemmatization with POS tagging for contextualization and discovered that Stemming gives a slightly better performance in our validation performance, so we stick with the Stemming approach. We conjecture that it is because stemming reduces the dictionary size and creates more overlapping of words in their stemmed form.

2.1.2 Models

After fitting a TF-IDF vectorizer on our training dataset, we fit i) a K-nearest neighbors Classifier, ii) a Multinomial Naive Bayes Classifiers, and iii) a Multinomial Logistic Regression model, on the training set.

2.1.3 Validation Performance

We then evaluate them on the validation dataset and we discovered that Multinomial Logistic Regression performs the best, with an evaluation score of 0.61, followed by the Naive Bayes Classifier with around 0.58. The k-NN classifier possibly suffers from the curse of dimensionality and performs not much better than a random classifier with a score of 0.32. (Note: a random classifier gives evaluation score of 0.3)

2.2 Neural Vectorization: Sentence Embedding

We attempt to implement the same classical machine learning models (e.g. k-NN, Logistic Regression) but with a more advanced word vectorization method.

2.2.1 Pre-trained Word and Sentence Embeddings

Word2Vec [Goldberg and Levy, 2014] is the first neural model trained to embed words into vectors and happens to be a natural choice to start with, but its 3-dimensional output for the whole dataset makes it difficult to feed into classical machine learning models. Instead, we adopt a pre-trained Sentence BERT [Reimers and Gurevych, 2019] available on HuggingFace that encodes each sentence into a 384-dimensional vector.

2.2.2 Simplified Preprocessing

Since SBERT can take a whole sentence (including punctuations) as input, for text preprocessing, we only remove URLs, user mentions, and hashtags, in order to preserve as much sentence information as possible.

2.2.3 Models

We then train two models i) Multinomial Logistic Regression, and ii) k-NN Classifier. Note that Multinomial Naive Bayes requires non-negative input values so it does not apply here.

2.2.4 Validation Performance

We evaluate the two models on the validation set and observed improvements on both our Logistic Regression and k-NN classifier produce significantly better results. In particular, our multinomial Logistic Regression obtains a new benchmark on evaluation score at 0.66, while the k-NN classifier gives 0.55, which shows a huge improvement from the BoW approaches.

3 Neural Models

In this section, we attempt to improve the above benchmark results above by implementing neural models. In particular, we start with a Long Short Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] RNN model with GloVe word embedding [Pennington et al., 2014] to train our baseline for neural models, followed by finetuning a BERT [Devlin et al., 2018] as an attempt to improve our results.

3.1 GloVe Embedding and Long Short Term Memory (LSTM)

3.1.1 GloVe Embedding

For word vectorization, we adopt the pre-trained Glove word embeddings with 100 dimensions, with added UNK and PAD tokens for out-of-vocabulary tokens and padding tokens respectively. We preprocess the data more of less in the same way as we did for TF-IDF, except that we do not stem or lemmatize our tokens as GloVe is able to handle different forms of vocabularies. Yet, we still lowercase all the vocabs as we noticed that the GloVe embedding contains mostly only lowercase tokens.

3.1.2 Vocab Lookup Table

We then build a vocab dictionary of 5000 words (chosen based on frequency in training data) and build an embedding matrix based on the embeddings, which essentially serves as a lookup table.

3.1.3 Maximum Sequence Length, Padding, and Truncation

We observed that the longest tweet in our training dataset consists of 44 tokens after preprocessing, but most of the tweets actually only contain around 10-20 words, so we set the maximum sequence to be 20. We pad sequences shorter than 20 words and truncate sentences that are longer than 20 words, which we see as the optimal compromise between information loss and excessive padding.

3.1.4 Model Training

We train the model by passing the embedded text data into a single-layer Bi-directional LSTM cell and a linear layer with hidden units = 256. We use Cross Entropy Loss as our criterion, a batch size of 128, and Adam optimizer with a learning rate = 0.003 to train the model for 10 epochs.

3.1.5 Validation Performance

The evaluation score we obtained on the valiation set with this simple LSTM model is around 0.63, which outperforms all the previous approaches, except for the Logistic Regression on pretrained SBERT. This showcases the power of pre-trained models and suggests the approach of finetuning pre-trained Transformer model in the next section.

3.1.6 Overfitting

Our model also suffers from overfitting during the training, as even if we are able to fit the LSTM up to a training accuracy of 0.9, the validation loss keeps increasing. We tried adding dropout, tuning the hidden unit size, changing the batch sizes, and even different maximum sequence lengths. Yet we are still unable to produce a model that outperforms the SBERT Logistic Regression at this stage.

3.2 BERT Finetuning

We attempt to improve our benchmarks by finetuning a pre-trained Bidirectional Encoder Representations of Transformers (BERT) [Devlin et al., 2018]. BERT is essentially a language encoder that captures deep understanding and knowledge of the language. We wish to harness its language understanding for our downstream task of sentiment analysis.

3.2.1 Word Preprocessing

We again make minimum word preprocessing on the original texts by only removing URLs, user mentions and hashtags from the tweets. Since BERT has its own tokenization and vocabulary lookup table, we use the pre-trained version of the model available on Huggingface.

3.2.2 Maximum Sequence Length

As BERT tokenization breaks down words into 'subwords', we use a maximum sequence length of 100 to ensure that whole sentences are included. Sentences that are too short or too long will be padded and truncated to have lengths of 100 tokens respectively.

3.2.3 Model Structure

We adopt a full BERT model with 12 layers and attach a linear layer at the end to produce the outputs. We attempted to freeze and unfreeze the BERT layers parameters and decided that we freeze the first 8 layers of parameters. We use a batch size of 32, the Cross Entropy Loss as our criterion, and Adam Optimizer with a learning rate of 0.0002 to train our model with 25 epochs. A small learning rate is chosen as we are fine-tuning the last layer of the model instead of training the whole model. The model is large and the learning might takes up to 25 minutes to run on a Google Colab GPU.

3.2.4 Validation Results and Observations

Our model (perhaps, surprisingly), does not outperform the simple LSTM model we trained above in the evaluation score on validation set. Yet, we observed that unlike the LSTM in the previous subsection which tends to overfit, our model is actually slightly underfitted. We attempted to unfreeze more BERT layers and plug in additional FC layers at the end, but they did not improve the overall model performance.

4 Results

Finally, we evaluate all of our models on the three given test sets based on the macro-f1 score of two classes. The table below summarises the results of our models.

Evaluation on Test Set based on Macro F1 Score in two classes			
Model	Test Set 1	Test Set 2	Test Set 3
Logistic Regression (BoW)	0.467	0.446	0.464
Logistic Regression (Sentence Embedding)	0.569	0.594	0.565
Naive Bayes Classifier (BoW)	0.413	0.421	0.444
K-NN Classifier (BoW)	0.237	0.268	0.238
K-NN Classifier (Sentence Embedding)	0.428	0.454	0.416
Glove + LSTM	0.589	0.588	0.556
BERT Finetuning	0.518	0.573	0.530

5 Conclusions and Future Works

In this project, we demonstrate the power of neural models and particularly, pre-trained language embeddings and encoders, in Natural Language Processing tasks like Sentiment classification. We observed that in general, the neural models and the classical methods with pre-trained neural embeddings tend to perform better than traditional Bag-of-Words approaches.

There are several aspects that our project that could be further improved. We believe that more combinations of the LSTM model configutations could be considered and tested if given more computational resources. Also, we did not test the performance of vanilla transformers [Vaswani et al., 2017] on our downstream tasks. The training time of a vanilla BERT is quite long so distilled version of it (i.e. DistilBERT) [Sanh et al., 2019] could be considered. The finetuning of BERT does not perform up to our expectation due to underfitting, which suggests that in the case where computational resources are abundant, we could also consider finetuning even larger versions of BERT, such as roBERTa [Liu et al., 2019].

References

[Devlin et al., 2018] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

[Goldberg and Levy, 2014] Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. cite arxiv:1402.3722.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. cite arxiv:1907.11692.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

[Ramos, 1999] Ramos, J. (1999). Using tf-idf to determine word relevance in document queries.

[Reimers and Gurevych, 2019] Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

[Rosenthal et al., 2019] Rosenthal, S., Farra, N., and Nakov, P. (2019). Semeval-2017 task 4: Sentiment analysis in twitter. *CoRR*, abs/1912.00741.

[Sanh et al., 2019] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.