Sentiment Analysis on Twitter Data

Sanjeev Venkatesan
Department of Computer Science
University of California at Los Angeles
sanjeev2@ucla.edu

Jakob Didio
Department of Computer Science
University of California at Los Angeles
jakobdidio@cs.ucla.edu

Srinath Naik
Department of Computer Science
University of California at Los Angeles
srinath@cs.ucla.edu

Yiyou Chen
Department of Computer Science
University of California at Los Angeles
yiyouchen@cs.ucla.edu

1 Introduction

Sentiment analysis, the task of identifying the sentiment underlying a selection of text, is a classically challenging natural language processing (NLP) task. This task is difficult for both machines and humans in many cases. For example, in the case of written text, it is challenging to determine an author's sentiment given that intonation is not represented well in written text. Sentiment analysis therefore serves as a suitably rigorous benchmark for determining a system's capability to reason about ideas conveyed through text that are not readily evident from only the vocabulary used. This task also has numerous application areas and is especially useful for downstream tasks such as determining potential consumers' overall attitudes towards and intentions regarding a particular product or service. In this report, we undertake the task of defining a model architecture for sentiment analysis, training it on a domain-specific corpus, and analyzing its ability to identify sentiment in the training domain and in several others.

1.1 Goal

The goal for this project is to build an end-to-end sentiment analysis model, trained on Twitter posts ("tweets"). We will be using the Sentiment140 annotated dataset from [1] for training and testing. We aim to develop a model that can receive a short text piece as input and, accurately and efficiently, return a sentiment classification for that text. We will test our trained model on data collected from Amazon food reviews [2] and IMDB movie reviews [3]. Ideally, the trained model could be used for a wide variety of tasks, like detecting toxicity in reported tweets or determining aggregate consensus of a group of tweets towards a particular topic, like a news event.

1.2 Motivation

We aim to address two primary inquiries with this project. The first is whether a lightweight architecture like the one proposed in this report is sufficient to produce a model that can effectively analyze the sentiment of Twitter posts. The second is whether the trained model then generalizes well to analyzing sentiment of corpora from other domains. Another goal of this project is for the team to gain experience in granularly constructing a sentiment analysis architecture, coupled with training and evaluating on multiple varied data sets.

2 Related Work

[4] presents the traditional Naïve Bayes algorithms that classifies sentiment by applying the Baye's rule to calculate the probability of each class (postive, negative, neutral). [5, 6, 7] present

Convolutional Neural Networks(CNN) approach to sentiment classification of individual sentences. [8, 9] present Long short-term memory(LSTM) approach to sentiment classification of individual sentences. [10] uses a two layered network for document-level sentiment classification.

[11] provides a Bayesian approach for ensemble learning after classification of individual parts (i.e. sentences). [12] provides a stacked classifier technique for ensemble learning.[13] surveys ways to incorporate user feedback into the learning. One reference for word embedding is [14], but we don't rule out the possibility of using third-party implementations.

[15] performed sentiment analysis using CNN and LSTM networks to achieve state of the art results. They use large unlabelled data to train word embeddings and a subset of that data to further fine tune using distance supervision so that dissimilar word embeddings are distant apart. They also presented an ensemble of multiple CNN's and LSTM's to boost up performance. [16] explores the idea of using cosine similarity instead of dot product to generate document embeddings and demonstrate further usage of these embeddings to achieve better performance on classification tasks. [17] leverages on BERT[18] model and incorporates contextualized representation with binary constituency parse tree to capture semantic composition by adding additional attention modules. It also describes transfer usage of sentiment composition learned from SST to other related tasks.

3 Proposed Method

We first propose to preprocess text data from the Sentiment140 data set using NLTK's TweetTokenizer [19]. Then, the tokenized data will be used as input to a Word2Vec model [20] to generate word embeddings which will be fed as input to our model. The produced embeddings will then be used as input to our deep learning model to train on the sentiment analysis task. The trained model will then be evaluated on a testing subset of the Sentiment140 data, along with data from Amazon food reviews and IMDB movie reviews.

4 Data

Our proposed evaluation strategy involves training our model to classify sentiments using the Sentiment140 Twitter dataset, then evaluating on two related datasets curated for sentiment analysis. More details on these three datasets are provided below.

4.1 Sentiment140

Our sentiment analysis model was trained on the Sentiment140 dataset [1]. 140,000 Twitter posts were collected by the Sentiment140 team using the Twitter Search API, and were automatically classified according to sentiment. The automatic classification was performed by analyzing the presence of emoticons in a given tweet, where a "positive" emoticon resulted in the tweet being assigned a positive sentiment, a negative emoticon resulted in a negative sentiment, and the absence of emoticons resulted in a neutral classification. The tweets also were scrubbed of any emoticon data besides simple text emoticons, e.g. :). Once we acquired the data, we tokenized every tweet using NLTK's [19] TweetTokenizer software. The TweetTokenizer was ideal for tokenizing the collected tweets, as it intelligently removed usernames (Twitter handles), truncated long punctuation strings, and correctly identified emoticon tokens. The tokenized tweets were then used to train a Word2Vec model [14], and the produced embeddings were used as input to our model.

4.2 Amazon Fine Food Reviews

After training on the Sentiment140 Dataset, one of the test datasets used to evaluate the quality of the model was the Amazon Fine Food Reviews dataset from [2]. The data contained 568,454 reviews from Amazon's collection of food items. The data was labeled according to the number of "stars" that an author associated with their review, which is a numerical value ranging from 1 to 5, with 1 indicating extreme displeasure with the product and 5 indicating great satisfaction. To utilize this data for testing with our model, we classified the sentiment of each review according to the following scheme: 1-2 star reviews received a negative sentiment, 3 star reviews received a neutral sentiment,

and 4-5 star reviews received a positive sentiment. After adjusting labels, 2,001 reviews were selected for use with testing, with an even split of 667 reviews for each sentiment category.

4.3 IMDB Movie Reviews

Similarly, we tested our model on the IMDB Movie Review dataset [3] as well. This dataset contains 50000 movie reviews equally divided into 'positive' and 'negative' sentiment. For our purpose we pre-processed and tokenized this dataset and labeled 'positive' as 4, 'negative' as 0 to be consistent with our twitter training and testing pipeline.

5 Model

We adopt the transformer model from *Attention is All you Need* [21] as the training network structure . The original paper focused on the task of sequence to sequence translation, where an encoder-decoder model is involved. The sentiment analysis is a classification task where only an encoder is required. Therefore, we could remove the decoder part, while keep the other parts of the structure. Figure 1 presents a brief overview of the model.

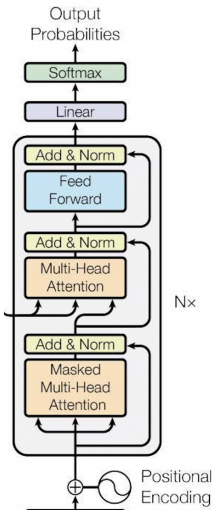

Figure 1: Transformers model adopted from [21]

5.1 Positional encoding

Given a word embedding prepared as in section 3, we first apply a positional encoding to it. The positional encoding uses the traditional method involving a coordinate wave function, $p(t)$, defined as

$$p(t) = \begin{pmatrix} \sin(\omega_1 t) \\ \cos(\omega_1 t) \\ \sin(\omega_2 t) \\ \cos(\omega_2 t) \\ \dots \\ \sin(\omega_{d/2} t) \\ \cos(\omega_{d/2} t) \end{pmatrix}$$

where $\omega_i = \frac{1}{10000^{2i/d}}$ and d is the embedding dimension.

The positional encoding takes into account the positions of words appeared in the text. The changing frequencies serves like the bit flips in the coordinate representation.

5.2 The Transformers

Each transformers layer consists of two sub-layers: masked multi-head attention and feedforward network.

The masked multi-head attention layer allows to "interpret" a word in the entire text, which is a non-local layers. In particular, after attentions are computed for each head, the results are concatenated to produce the final encoding of each word. After this step, the encoding takes words' original meanings, positions, and relative meanings into consideration.

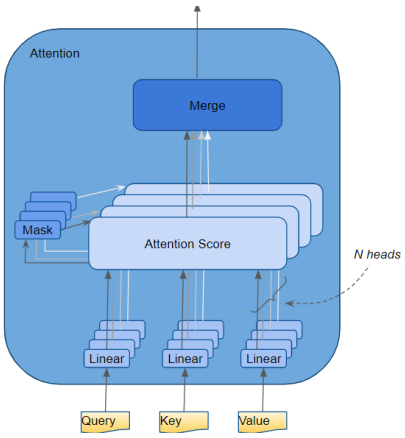

Figure 2: Figure from [22]

After each attention layer comes the feedforward network, which consists of linear models that learns from the attention encoding.

The entire transformers layer can be repeated multiple times. And there can be a dropout rate defined throughout the entire layer.

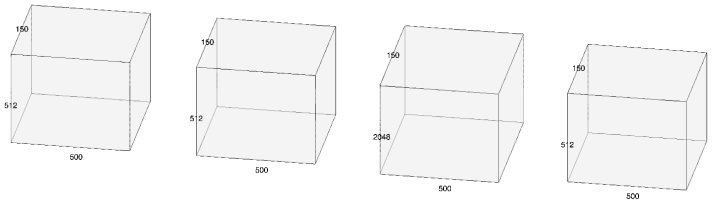

Figure 3: Transformers layer dimensions: one to two: multi-head attention; two to three: feedforward layer; three to four: feedforward layer.

5.3 Linear Classification

The output from the transformers layer is then fed into a linear classification model, which consists of multiple fully connected layers (MLP). For the purpose of this paper, we fix the dimension of the labels to 3 (negative, neutral, and positive). In our model, instead of feeding all all encoded words in a sequence to a linear classifier, we first shrink the dimension of the encoding for each word in the sequence using a MLP, and then apply another MLP to all the words in the sequence.

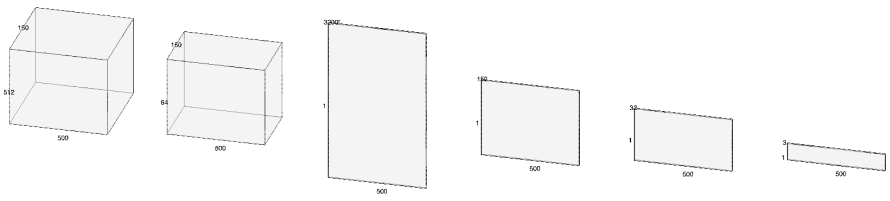

Figure 4: MLP for linear classifier

6 Training

We used Pytorch for training the model. We have performed various experiments by changing number of transformer layers, heads and layers in final MLP layer after the transformer encoder. We were limited by the computation power needed to fully train this network on the twitter dataset. Each epoch of training took nearly 3 hours of GPU time.

6.1 Optimizer

Adam Optimizer with weight decay 0 is used.

6.2 Loss Function

The cross entropy loss function is used as the loss function for the classification purpose since probability distributions are .

6.3 Hyperparameters

- Sequence length: 150.
- Batch size: 500.
- Embedding dimension: 512.
- Number of batches per epoch: 300.
- Number of class labels: 3.
- Initial learning rate: 10^{-2}.
- Scheduler learning rate decay per 200 batches: 0.8.
- Lower bound of learning rate: 10^{-6}.
- Transformers number of heads: 4.
- Transformers feedforward network hidden layer dimension: 2048.
- Number of feedforward network in each transformer layer: 2.
- Number of transformers layers: 1.
- Dropout rate: 0.1.

7 Results

Due to limitations of computational power, we were able to run our model on a maximum of 5 epochs(equivalent to 15 hours of training time). The model achieved a training accuracy of 50%, which clearly shows that it has underfit the training data. With this model, we found the test accuracy to be around 37% on twitter data. Our experiments of inference on Amazon reviews dataset resulted in a 33% accuracy where as inference on IMDB data resulted in 50% accuracy. The Test accuracy for each dataset is summarized in below table.

Test Data	Accuracy
Sentiment140	37%
Amazon	33%
IMDB	50%

8 Conclusion

Our model's performance indicates that the model is essentially performing no better than just randomly guessing classifications. We believe that the model's poor performance can be attributed to a variety of factors. We generated our own embeddings and trained our model end to end on a limited dataset. Sentiment Analysis is a task that can often require extremely large amounts of training data to produce effective results. Training a competitive sentiment analysis model also generally requires extensive data and GPU hours. Due to compute limitations, we were unable to train our model on more data or for more epochs. These factors all combined to impact performance and prevented us from reaching a competitive threshold.

Although our model did not reach a relatively high accuracy, our findings instead highlight the complexity of the sentiment analysis task. We conclude that sentiment analysis requires a large amount of training data and a less lightweight model than our design to produce good results. The performance of our trained model on similar datasets also suggests that it is difficult to train a model that can generalize to related sentiment analysis tasks.

8.1 Next Steps

The evaluation results of our model on the three datasets offers some insight into next steps we can take to improve accuracy. Due to time and compute limitations, we were unable to train our model on larger datasets. Increasing the size of the training dataset could potentially improve the performance of our model on all three tasks. Additionally, the automatic annotation used by the creators of the Sentiment140 dataset to generate labels could be improved by human annotations.

Another research avenue that could lead to performance improvements involves using transfer learning to fine tune the model on a new dataset before evaluating on that dataset. This could be accomplished by appending a linear layer to our network, thus providing the additional capacity to extend to new datasets with a different number of class labels.

References

[1] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *Processing*, 150, 01 2009.

[2] Julian McAuley and Jure Leskovec. From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. *WWW*, 2013.

[3] Lakshmipathi N. Sentiment analysis of imdb movie reviews, 2020. Accessed March 15, 2022, https://www.kaggle.com/lakshmi25npathi/sentiment-analysis-of-imdb-movie-reviews/data.

[4] Hanhoon Kang, Seong Joon Yoo, and Dongil Han. Senti-lexicon and improved naïve bayes algorithms for sentiment analysis of restaurant reviews. *Expert Systems with Applications*, 39(5):6000–6010, 2012.

[5] Ping Li, Jianping Li, and Gongcheng Wang. Application of convolutional neural network in natural language processing. In *2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 120–122. IEEE, 2018.

[6] Wei Wang and Jianxun Gang. Application of convolutional neural network in natural language processing. In *2018 International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pages 64–70. IEEE, 2018.

[7] Aliaksei Severyn and Alessandro Moschitti. Unitn: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 464–469, 2015.

[8] Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. Effective lstms for target-dependent sentiment classification. *arXiv preprint arXiv:1512.01100*, 2015.

[9] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.

[10] Guozheng Rao, Weihang Huang, Zhiyong Feng, and Qiong Cong. Lstm with sentence representations for document-level sentiment classification. *Neurocomputing*, 308:49–57, 2018.

[11] E. Fersini, E. Messina, and F.A. Pozzi. Sentiment analysis: Bayesian ensemble learning. *Decision Support Systems*, 68:26–38, 2014.

[12] James Thorne, Mingjie Chen, Giorgos Myrianthous, Jiashu Pu, Xiaoxuan Wang, and Andreas Vlachos. Fake news stance detection using stacked ensemble of classifiers. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism*, pages 80–83, 2017.

[13] Zijie J Wang, Dongjin Choi, Shenyu Xu, and Diyi Yang. Putting humans in the natural language processing loop: A survey. *arXiv preprint arXiv:2103.04044*, 2021.

[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[15] Mathieu Cliche. Bb_twtr at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms. *arXiv preprint arXiv:1704.06125*, 2017.

[16] Tan Thongtan and Tanasanee Phienthrakul. Sentiment classification using document embeddings trained with cosine similarity. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 407–414, 2019.

[17] Da Yin, Tao Meng, and Kai-Wei Chang. Sentibert: A transferable transformer-based architecture for compositional sentiment semantics. In *ACL*, 2020.

[18] 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.

[19] Nltk :: Natural language toolkit, 2022. Accessed March 15, 2022, https://www.nltk.org/.

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[22] Ketan Doshi. Transformers explained visually (part 3): Multi-head attention, deep dive, Jun 2021.