Sentiment analysis using Transformer with attention mechanism

Shivaleela Ishwarappa Hubli
Department of Computer Science and Engineering
State University of New York at Buffalo
shivalee@buffalo.edu

Abstract

The task of sentiment analysis is to provide the sentiment corresponding to the highest probability for any given text. With the advancement in neural networks, it is possible to have a look into the network's decision making process. We implement multi-head attention mechanism that answers the question of what part of the input should be focused on, which can be used to diagnose the network's interpretation of the data. The attention mechanism also allows the network to focus more on relevant areas of the input, thus improving the classification performance. The proposed model performs sentiment analysis task which is a binary classification problem and achieves 89% accuracy on IMDB dataset and 93% accuracy on Yelp dataset. Furthermore, the attention mechanism is visualized by extracting attention weights and incorporating color maps to them.

1. Introduction

Sentiment analysis refers to "the use of natural language processing, text analysis, computational linguistics, and bio-metrics to systematically identify, extract, quantify, and study effective states and subjective information", by the definition of Wikipedia. In the digital era, especially with the incorporation of social media as an inseparable part of daily life, the opinion of an individual is now reachable to a wider network and this in turn has a domino effect influencing the decisions of other individuals. For example, when we plan to buy a product on an e-commerce website we pay much attention to the customer reviews, that influence our decision of buying the product.

Sentiment analysis has wide applications that range from customer service to marketing, applicable for survey responses, reviews, etc. With the advancement in natural language processing and deep learning, we have many advanced networks that can be used for sentiment analysis of text such as RNN, LSTM, GRU, BERT, transformer etc. The following sections provides an overview of few networks with their advantages and disadvantages which can be used for this problem.

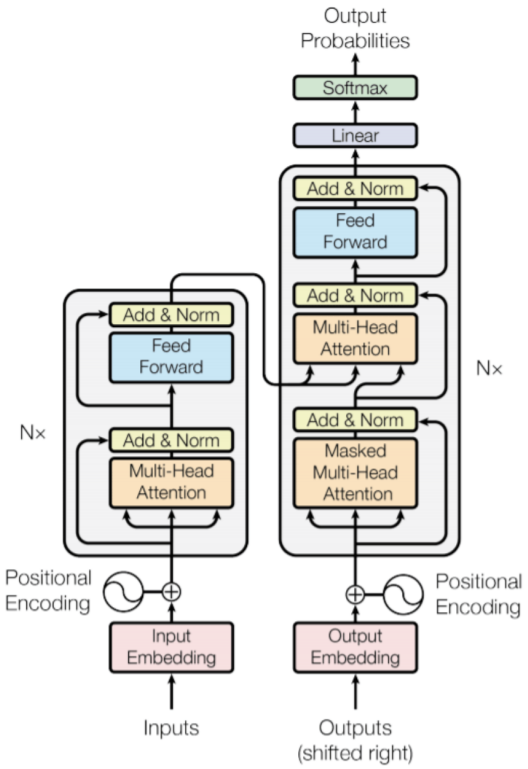

Figure 1. Transformer model architecture [4]

Recurrent Neural Networks are feed forward neural networks rolled out over time. They deal with sequence data, where the input has some defined ordering. In sentiment analysis, the movie review is the input and a fixed sized vector is the output indicating how good or bad that person thought the movie was. However, RNNs have some disadvantages. RNNs are slow to train. Though we use a truncated version of Back Propagation Through Time (BPTT) to train it, even then it's very intense. RNNs cannot deal with long sequences very well. If the network is

too long, we may face the issue of vanishing gradients or exploding gradients.

Long Short-Term Memory Neural Network was introduced in 1991 with Long Short-Term Memory cells in place of neurons. This cell has a branch that allows the past information to skip a lot of the processing of the current cell and move on to the next. This allows the memory to be retained for longer sequences. LSTM can deal with longer sequences of words probably to the order of 100's of words. LSTMs are slower than RNNs since they are more complex. For both RNNs and LSTMs, the input data has to be passed sequentially hence, does not exploit the current computational GPUs that are designed for parallel computations. The recent advancement in the incorporation of attention mechanism into LSTM models have proven to perform better than the LSTM models without them [2]. Though, those models are out of scope for this project.

Transformer Neural Network, was introduced in 2017 [4]. The network employs an encoder-decoder architecture much like RNNs. The difference is the input sequence can be passed in parallel. For example, the language translation from English to French. With a RNN encoder, we pass an English input sentence with one word after the other. The word embedding is generated one-time step at a time. With a Transformer encoder, there is no concept of time step for the input. We pass all the words in the sentence simultaneously and determine the word embedding simultaneously. Figure 1 shows the general architecture of a transformer model. Transformers have many advantages, such as:

1. Non sequential: sentences are processed as a whole rather than word by word.

2. Self-Attention: this is the newly introduced 'unit' used to compute similarity scores between words in a sentence.

3. Positional embedding: another innovation introduced to replace recurrence. The idea is to use fixed or learned weights which encode information related to a specific position of a token in a sentence.

In order to understand the network's decision making process, we lean towards the attention mechanism found in the transformer models. It is found that that models that connects encoder and decoder through an attention mechanism performs the best [4].

1.1. Problem statement

The objective of the project is to perform sentiment analysis on the given text using a transformer neural network with attention mechanism.

Task definition: (1) Implement transformer model with attention mechanism to classify the sentiment of reviews in IMDB and Yelp datasets with the following configurations: number of layers from 1 to 3 and number of heads = 1, 2, 4 and 8. (2) Implement the visualization of attention mechanism in terms of color maps for transformer models. (3) Implement baseline models like RNN, LSTM, GRU to compare the performance with respect to transformers. (4) Calculate the accuracy, loss, F1 score, precision and recall for every experiment.

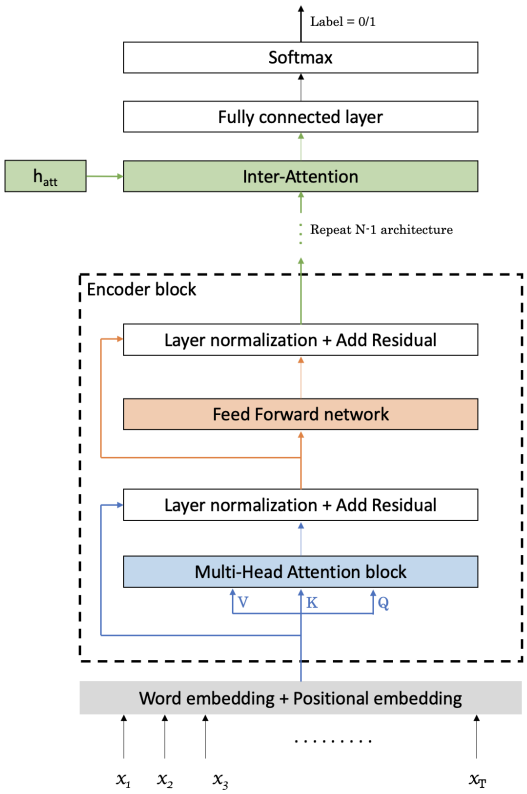

Figure 2. Proposed model architecture

2. Transformer model

We introduce transformer and its detailed implementation in this section. The major step is the pre-training. During pre-training, the model is trained on labeled data for the classification task. A distinctive feature here is that, greedy decoding is used, where the model produces the outputs one at a time and the model is selecting the output with the highest probability from the probability distribution and throwing away the rest. This is called greedy decoding.

Model architecture The transformer model consists of an *Encoder* block and a greedy decoding *Inter-Attention* block, followed by a linear layer and softmax layer for classification. The encoder block is made up of a *Multi-head Attention* block and a feed forward network, both of which

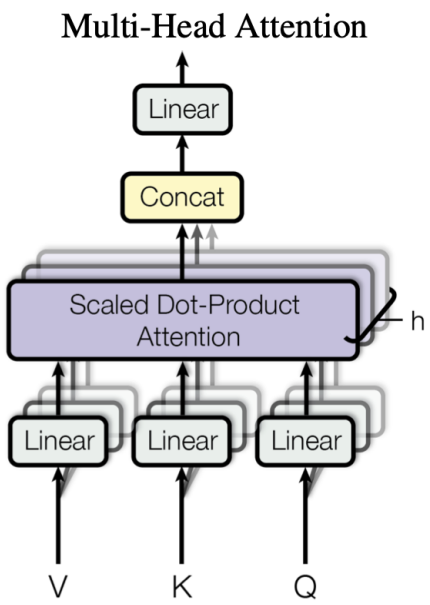

Figure 3. Multi-Head Attention unit [4]

are followed by a layer normalization with the addition of residual inputs. Figure 2 shows the proposed architecture of the transformer model with greedy decoding as per Git repository which implements an IEEE paper [5].

The *Multi-head Attention* deals with, what part of the input should be focused on? For every word, we can have an attention vector generated which captures contextual relationships between words in the sentence. The attention vector may weight its relation to itself much higher which is undesirable. So, we determine multiple attention vectors for every word and take a weighted average to compute the final attention vector using the *Scaled Dot-Product Attention* as shown in the Figure 3. It uses multiple attention vectors, hence, the name "Multi-head Attention" block. It takes 3 abstract vectors, Q, K and V that extract different components of an input word. We get Q, K and V vectors for every single word. We use these to compute attention vectors for every word using the formula [4]:

$$Z = softmax\left(\frac{Q.K^T}{\sqrt{d_K}}\right).V$$

For multi-head attention we have multiple weight matrices W^Q, W^K and W^V. So, we will have multiple attention vectors Z_h for every word. However, the feed forward neural network is expecting only 1 attention vector per word. So, we use another weighted matrix W^Z to make that the output is a single attention vector per word using the formula:

$$Z = [Z_1 Z_2 ... Z_h].W_Z$$

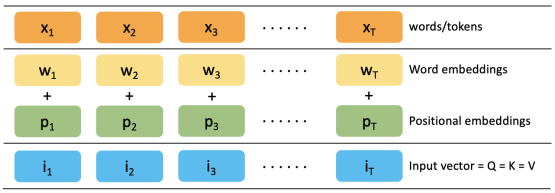

Figure 4. Word embedding + Positional embedding

A simple *feed forward neural network*, consisting of 2 fully connected linear layers. It is applied to the attention vectors one at a time to transform the attention vectors into a form that is digestible by the next block. Since the vectors are independent, we parallelize the process and the output is a set of encoded vectors for every word.

After every layer, *layer normalization* is applied along with the addition of residual input. The layer normalization performs the normalization across each feature instead of each sample, it achieves better stabilization in the normalization of every layer. This encoder block is repeated for N_x times.

The *Inter-Attention* calculates the overall outputs and attention weights taking in the outputs of the encoder block using the formula:

$$W_I = softmax\left(tanh\left(Q^T + K\right)\right)$$

$$output = V^T.W_I$$

The output from the *Inter-Attention* is passed through a fully connect layer, followed by a softmax layer that transforms the outputs into a probability distribution which is human interpretable. The final output is the sentiment corresponding to the highest probability.

Input/Output representation Every word is mapped to a point in space where words with similar meaning are physically closer to each other. The space is called an **Embedding Space**. This embedding space maps a word to a vector and the process is called **Word Embedding**. We use a pretrained embedding space called GloVe (Global Vector for Word Representation) for word embedding.

But, the same word in different sentences may have a different meaning. Hence, a Positional Encoder is used. The **Positional Embedding** is added to the word embedding to get the notion of context of the word in the sentence. It is a vector that has information on distances between the words in the sentence. You can use sine and cosine functions to generate this vector [4].

$$PE_{(pos,2i)} = sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

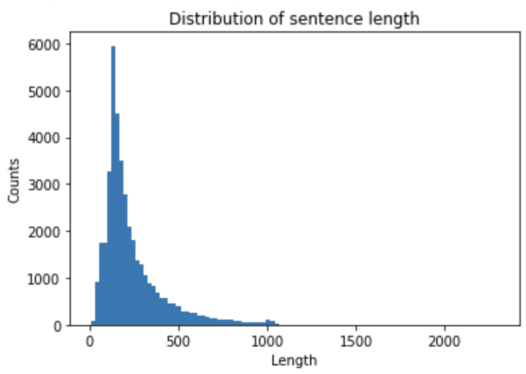

Figure 5. Sentence length distribution of IMDB dataset

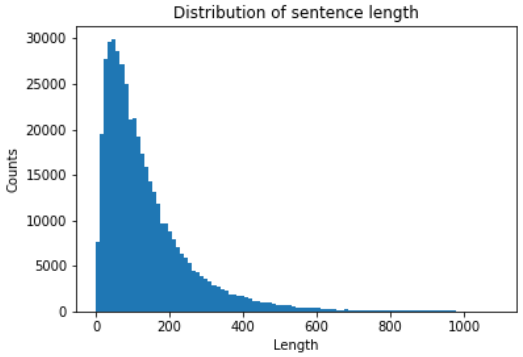

Figure 6. Sentence length distribution of Yelp dataset

You can also use other functions to generate positional embedding. Figure 4 shows the input embedding vector of word.

Loss function To calculate the loss of the function, we use the **Binary cross entropy** [3] loss using *BCEWithLogitsLoss* method provided by PyTorch because this version is more numerically stable than using a plain Sigmoid followed by a *BCELoss* as, by combining the operations into one layer, we take advantage of the log-sum-exp trick for numerical stability. In the site, the loss is described as:

$$L = l_1, l_2, ...{l_N}^T$$

$$l_n = -w_n \left[y_n . log\left(\sigma\left(x_n\right)\right) + \left(1 - y_n\right) . log\left(1 - \sigma\left(x_n\right)\right) \right]$$

Adam optimizer We use Adam optimizer on the model to update the model weights on training data.

3. Experiment

In this section we present the transformer model results on the sentiment classification task on IMDB and Yelp datasets with different configurations with respect to the model parameters like number of heads, number of hidden layers etc. We also showcase the comparison of the model performance with other off-the-shelf base models such as logistic regression, simple neural network, CNN, LSTM and GRU.

3.1. Datasets

IMDB: IMDB movie review dataset is a binary sentiment classification dataset. It contains 50,000 highly polar movie reviews. The dataset has the columns as *id, type, review, label, file*. For this project, the dataset is split into 40,000 training, 5,000 validation and 5,000 testing entries randomly. Figure Figures 5 shows the sentence length distribution of IMDB dataset.

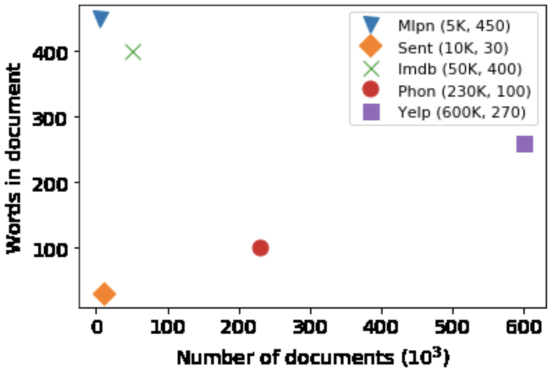

Figure 7. Analysis of the 2 datasets [6]

Yelp: The Yelp review dataset is a binary sentiment classification dataset. It is constructed by considering stars 1 and 2 negative, and 3 and 4 positives. Negative polarity is class 1, and positive class 2. There are 2 columns, *label* and *review text*. The dataset is split into 4,47,999 training, 1,12,000 validation and 37,999 testing entries randomly. Figure Figures 6 shows the sentence length distribution of IMDB dataset.

Dataset analysis: Though, both the datasets are review based binary classification datasets, they have different characteristics. As shown in Figure 7, the IMDB dataset might have lesser number of documents but the size of documents is higher. And the Yelp dataset has comparatively higher number of documents but the size of the documents is much smaller. This might lead to over-fitting problem when used in simple networks.

3.2. Evaluation metrics

We evaluate our models using Accuracy (A), F-1 score (F1), Precision (P), Recall (R), Loss (L) and ROC curve. An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at

Model	Accuracy	Loss
Simple neural network	0.746	0.536
Convolutional neural network	0.850	0.336
Recurrent neural network	0.593	0.670
LSTM	0.849	0.340
Bidirectional LSTM	0.851	0.338
GRU	0.856	0.331
Transformer	**0.893**	**0.272**

Table 1. Performance of baseline models and the transformer model on IMDB dataset.

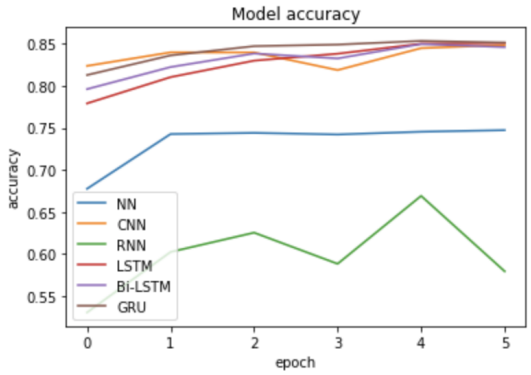

Figure 8. Accuracy graph of baseline models

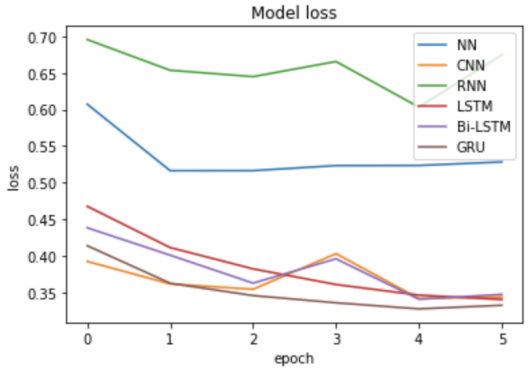

Figure 9. Loss graph of baseline models

all classification thresholds. An ROC curve plots TPR vs. FPR where TPR (True Positive Rate) is the recall and FPR (False Positive Rate) given by:

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

Layer	Head	A	F1	P	R	L
1	1	0.864	0.875	0.810	0.951	0.344
1	2	0.890	0.893	0.867	0.920	0.278
1	4	0.873	0.881	0.827	0.943	0.312
1	8	0.876	0.882	0.839	0.930	0.312
2	1	0.876	0.877	0.864	0.891	0.297
2	2	0.871	0.876	0.839	0.917	0.311
2	4	0.860	0.867	0.828	0.910	0.332
2	8	0.871	0.876	0.845	0.909	0.311
3	**1**	**0.893**	**0.893**	**0.888**	**0.899**	**0.272**
3	2	0.864	0.873	0.818	0.936	0.332
3	4	0.885	0.882	0.907	0.857	0.279
3	8	0.889	0.887	0.899	0.876	0.277

Table 2. Performance of transformer model on IMDB dataset.

3.3. Implementation details

Feature extraction: We start the preprocessing by removing punctuation and stop-words followed by tokenizing the text. Then the tokens are converted to number series and encoded using pre-trained *GloVe* embedding with a vocabulary size of 20,000, resulting in word embedding. This along with the positional embedding, acts as the input for the transformer model.

Configurations: For both IMDB and Yelp datasets, we have used different combinations for number of hidden layers and heads of the transformer model. The following are the configurations in the format of 'number of hidden layer': 'layers', 'number of heads': 'heads' used in experimentation:

'layers'= 1, 'heads'= 1, 2, 4 and 8
'layers'= 2, 'heads'= 1, 2, 4 and 8
'layers'= 3, 'heads'= 1, 2, 4 and 8

Attention maps: In order to visualize the attention mechanism of the model, we extract the attention weights of the model and map it to the text using a color map. We have implemented the incorporation of color maps using this Git repository Higher the intensity of the background color of the word, higher the attention given to the word.

4. Results

Table 1 shows the performance of baseline models for *IMDB* dataset with setting the number of epochs = 6, vocabulary size = 2000 and embedding size = 100. The *Tokenizer* is set to consider 5000 most frequent words. All the models use *Adam optimizer* and *Binary cross entropy* loss with 128 hidden units and *Sigmoid* activation.

The performance of the baseline models are compared using the evaluation metrics *Validation accuracy* and *Validation loss* and plotted as shown in Figures 8 and 9 .

Next, we experimented on the proposed transformer

Layer	Head	A	F1	P	R	L
1	1	0.910	0.908	0.936	0.881	0.220
1	2	0.934	0.935	0.923	0.947	0.168
1	4	0.935	0.934	0.942	0.927	0.166
1	8	0.935	0.934	0.941	0.928	0.171
2	**1**	**0.935**	**0.934**	**0.941**	**0.927**	**0.165**
2	2	0.933	0.934	0.925	0.943	0.177
2	4	0.931	0.930	0.945	0.916	0.175
2	8	0.929	0.927	0.961	0.895	0.179
3	1	0.922	0.919	0.963	0.878	0.192
3	2	0.932	0.932	0.924	0.941	0.172
3	4	0.927	0.925	0.949	0.902	0.186
3	8	0.927	0.928	0.917	0.939	0.184

Table 3. Performance of transformer model on Yelp dataset.

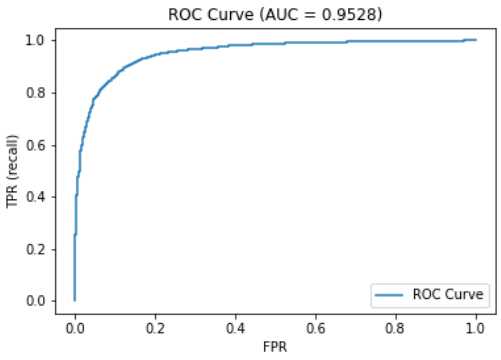

Figure 10. ROC curve

model with different configurations and the results are as shown in the Tables 2 and 3 achieving a 89% accuracy on IMDB dataset and 93% accuracy on Yelp dataset. The network performs well, compared to the basic neural networks. We have also plotted the ROC curve in the experimentation as shown in Figure 10.

The most significant aspect of the model, i.e. the attention mechanism can be visualized in a human interpretable way. Figure 11 shows an example of a customer's movie review from the IMDB dataset. Each word is associated with a background color with varying densities of *Blue*. The darker shades can be interpreted as more significant words in determining the essence of the text and hence have higher weightage while determining the sentiment of the customer. Words like *better*, *good*, *well* have received higher attention compared to insignificant words like *and*, *should* and *by*. Similarly Figure 12 shows an example of a customer's review of a Doctor from the Yelp dataset.

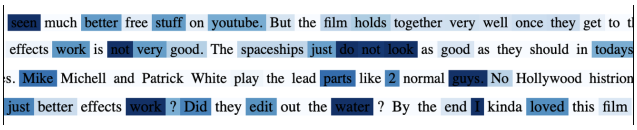

Figure 11. Visualization of attention mechanism in transformers on IMDB dataset.

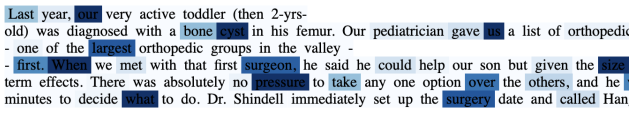

Figure 12. Visualization of attention mechanism in transformers on Yelp dataset.

5. Conclusion

Sentiments deal with the psychological aspect of an individual and hence, it is very difficult to comprehend. In order to understand the intention and in-turn the sentiment associate with it through text is a very challenging task. Hence, a model that can perform such a difficult task is required to provide certain reasoning for its predictions. In this work, we have displayed the potential of attention mechanism for the task of sentiment analysis which performs better than other deep neural networks with vanilla setting and also, provided the visualization of the model's reasoning in terms of attention maps. Larger models like Google's BERT performs much better than the transformers [1] but, such models are quite huge. Also, in order to understand the attention mechanism, transformers are a good start. The baseline models over-fit on Yelp dataset. In future, the regularization techniques like early-stopping, dropout etc. can be implemented to train those models on Yelp dataset.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 6

[2] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation, 2015. 2

[3] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip H. S. Torr, and Puneet K. Dokania. Calibrating deep neural networks using focal loss, 2020. 4

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. 1, 2, 3

[5] M. Wang, Y. Zhu, S. Liu, C. Song, Z. Wang, P. Wang, and X. Qin. Sentiment analysis based on attention mechanisms and bi-directional lstm fusion model, 2019. 3

[6] Erion Çano. *Role of Data Properties on Sentiment Analysis of Texts via Convolutions*, pages 330–337. 03 2018. 4