Sentiment Analysis with Emotes using Attention-Based Neural Networks

Ailin Wang[a]

[a]*Graduate School of Arts and Sciences, The University of Tokyo, Tokyo, Japan*

July, 2022

Abstract

Live streaming services that utilizes live multimedia sharing technology among people has become more and more prevalent. We developed a model using attention-based neural networks for word-level sentiment analysis of speech transcription data. We combined pre-trained Word2vec and positional encoding as the input for the proposed model. While sentiment analysis is a difficult task even for human, the result of the proposed model shows some level of sentimental understanding of the giving input text. We believe this technique can also be implemented in live streaming or even in online lecture to boost interaction between viewers.

1. Introduction

Live streaming services that utilize live multimedia sharing technology among people have become more and more prevalent. Twitch [3] is an American company that specializes in live video game broadcasting, including coverage of esports events. It also provides music streams, original material, and "in real life" feeds. Millions of people gather live each day on Twitch to chat, participate, and create their own entertainment. While users use text messages in the chat to interact with the streamer and other viewers, they also use emotes to express their feelings, when words are just not enough. This project focuses on extracting sentiment from speech transcription of streams. Sentiment analysis to predict emotes can be seen as a problem of multi-class classification, which a classifier model can solve.

1.1. Twitch Emotes

Users use emotes frequently to express their feelings while interaction with others on Twitch chat. Twitch emotes has become a part of internet culture, however, they are like a language of their own and are sometimes hard to understand for people that are not familiar with Twitch. Here are some examples of Twitch emotes and their usage and meanings. "Pog" features a man opening his mouth in surprise or excitement in an exaggerated manner. It is often used to express excitement. "LULW" is used to express laughter. It is also used along with the phrase "sez u" which means "says you" to expose the hypocrisy of the streamer. "Pepelaugh" is often accompanied by the Spanish phrase "El no sabe" which means "He does not know". It is often used when something is going to happen but the streamer is still unaware of that.

2. Methods

In this chapter, we discuss the theory behind the methods and techniques used in this project. In this project, pre-trained Word2vec embedding was used to capture the semantics of the input text, which enables word-level sentiment analysis. An attention-based encoder model serving as a classifier was used to determine which emote expresses the sentiment of the input text.

2.1. Word2vec

Word2vec [2] is an efficient way to create word embeddings for words from a large corpus while being able to capture the semantics. Its input is a text corpus and it outputs a set of vectors: feature vectors that represent words in that corpus. The fundamental idea of Word2vec is placing semantically similar words close together in the vector space, that is to say, the cosine similarity of the word vectors is smaller.

Word2vec is typically trained using one of the following two approaches: 1) continuous bag-of-words (CBOW), where the distributed representations of context (surrounding words) are combined to predict the word, and 2) Skip-gram, where the distributed representation of the input word is used to predict the context.

[0]https://cdn.frankerfacez.com/emote/210748/1
[0]https://cdn.frankerfacez.com/emote/139407/1
[0]https://cdn.frankerfacez.com/emote/263833/1

The training objective of the Skip-gram model is to find the word representations, i.e., to adjust the weights of the hidden layer in order to predict neighbouring words in a sentence. More formally, given a sequence of training words $w_1, w_2, w_3, ..., w_T$, the objective of the Skip-gram model is to maximize the average log probability,

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p\left(w_{t+j} \mid w_t\right) \qquad (1)$$

where c is the window size, which is a hyper-parameter to define the neighboring words of the training context.

2.2. Transformers

In this section, we discuss the rationale for using transformer as a natural language processing technique and the theory behind the transformer. The transformer model's efficiency in enabling parallelization is one of its most significant advantages. Transformer models enable for the calculation of relations of tokens to all other tokens on various standards independently in each head, requiring just some basic linear transformations, which makes parallelization possible. The original seq2seq transformer consists of several encoder and decoder blocks, each of which has multiple identical layers stacked on top of each block [4]. The Encoder block consists of several identical layers and each layer is made up of two sub-layers, which are multi-head self-attention mechanism followed by a simple, position-wise fully connected feed-forward network. Furthermore, the fully connected layers are prone to overfitting, so dropouts are often applied bettween each layers.

Positional Encoding. A positional encoding is required for the model to obtain information about the relative position of the tokens in the sentence, since attention layers see their input as a set of vectors, with no sequential order. Furthermore, word embeddings do not encode the relative position of tokens in a sentence. In transformer models, the positional encoding vector is added to the embedding vector at the bottoms of the model (encoder and decoder blocks). The formula for calculating the positional encoding is as follows:

$$\begin{aligned} PE_{(\mathrm{pos},2i)} &= \sin\left(pos/10000^{2i/d_{\mathrm{model}}}\right), \\ PE_{(pos,2i+1)} &= \cos\left(pos/10000^{2i/d_{\mathrm{model}}}\right). \end{aligned} \qquad (2)$$

where pos is the position and i is the dimension.

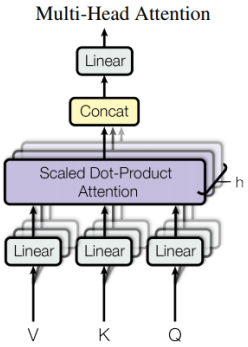

Figure 1: Multi Head Attention. Figure from Ashish Vaswani et al. [4]

.

Scaled Dot-Product Attention. An attention function can be defined as mapping a query and a set of key-value pairs to an output, all of which are vectors. The resulting output is calculated as a weighted sum of the values, with each value's weight determined by the query's compatibility function with its corresponding key. The particular attention mechanism used by transformer models are called "Scaled Dot-Product Attention", where the dot products of q and k are scaled down by $\sqrt{d_k}$ (dimension of queries and keys). Assuming that q and k are d_k-dimensional vectors whose components are independent random variables with mean 0 and variance 1, then their dot product, $q \cdot k = \sum_{i=1}^{d_k} u_i v_i$, has mean 0 and variance d_k. Since we would prefer these values to have a variance of 1, we divide the dot product by $\sqrt{d_k}$. Formally we have a function to calculate the attention that takes three inputs: Q (query), K (key), and V (value),

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}_k\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \quad (3)$$

Multi-head Attention. Figure 1 is the figure of Multi-head Attention mechanism from the original paper of transformer. The model can jointly attend to input from various representation subspaces at various points thanks to the splitting of Q, K, and V into several heads as opposed to a single attention head, as shown in this figure. Each head has a lower dimensionality after the split; therefore the overall computing cost is equal to a single head's attention with full dimensionality. In conclusion, Multi-head Attention only uses some basic linear transformations.

3. Experiments

3.1. Data Acquiring and Processing

The subtitles of five YouTube videos of past streams and Twitch chat text data were obtained to generate training data. 14 Twitch emotes were used for sentiment analysis. We processed the Twitch chat data to determine which emote was frequently sent by the viewers during that period to determine the target emote. Then we traced back the subtitles before that period to determine the input text for the training data. A total number of 1813 pairs of input text and target emotes were generated as the training data. Then cross-validation of 4 folds was used during the training, and the soft voting of these 4 results was used during testing, where a test data set containing 30 curated samples was used.

3.2. Pre-trained Word2vec Embeddings

The pre-trained Word2vec embeddings for the English language used in this project were obtained from Wikipedia2Vec [5] with the following hyperparameter settings: window = 5, iteration = 10, negative = 15, dimension = 100.

3.3. Proposed Model

The original Sequence-to-sequence (seq2seq) models are encoder-decoder models popular in translation tasks. However, the attention-based model in the proposed model only utilizes the encoder part, generating logits for property prediction. The reason behind that is rather than developing a sequence of words, some kind of average layer is better suitable for classification. And instead of predicting the next word, information from the whole input text is used, thus there is no need for any masking in the attention block.

Figure 2 shows the architecture of the proposed model. The proposed model can be divided into 2 segments. The first segment, comprised of an encoder of a transformer, feeds a multi-head self-attention unit with pre-trained Word2vec embeddings combined with positional encoding. The weights for the input embeddings are locked after input, so the training does not update the pre-generated embeddings. The logits obtained from the first segment are then averaged by a global average pooling layer in the second part of the model, passing through position-wise feed-forward networks with dropouts, resulting in a value with a specific dimension according to the task. For multiclass classification, Sigmoid activation is used at the final dense layer.

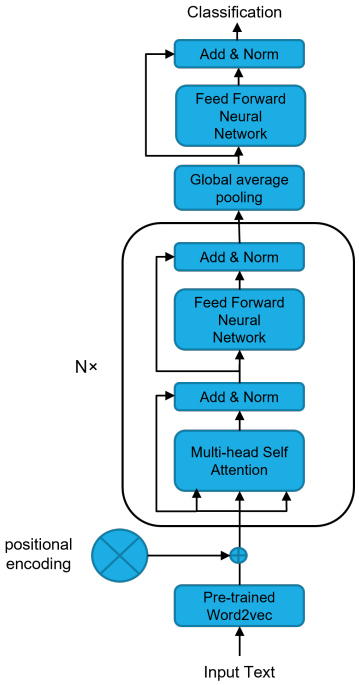

Figure 2: Diagram of the Proposed Model Architecture

.

3.4. Model Training

Here we discuss the detailed settings for the training of the proposed model. Dimension of the transformer model is equal to the dimension of the input embeddings, thus $d_{model} = 100$. The transformer depth was set to $N = 4$. The number of heads in the multi-head attention block was set to 5. And the dimension of the position-wise feed-forward networks was set to 100. The dropout rate was set to 0.1.

Adam optimizer [1] was used with the following settings: $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-9}$.

Learning rate setting can be tricky in training deep neural networks. Setting the learning rate too high will result in the parameter vector bouncing around chaotically and being unable to settle into a deeper region of the loss function. While setting the learning rate too low will result in a waste of computation time. Thus, it is typically beneficial to anneal the learning rate. Our study applied the decaying learning rate during the course of training according to the original transformer model [4]. This results in a changing learning rate that linearly increases for the first warmup steps and training steps and decreases thereafter proportionally to the inverse square root of the step number. Warmup steps were set to 5% of the total training steps. An early stop was implemented to avoid over-fitting and improve training time, so the

model stops training when the monitored metric has stopped improving during the last 20 epochs.

4. Results

During testing, the proposed model was evaluated by accuracy and the area under the curve (AUC) of the receiver operating characteristic (ROC) curve, where a larger value means better performance. Cross entropy was used as the loss function. Training loss of 5.49 ± 5.21, validation loss of 2.44 ± 0.03 were recorded during testing. The model was tested with a curated test set containing 30 samples, and the test result was obtained from the soft voting of 4 cross validation folds. This resulted in an accuracy of 16.7% and AUC-ROC of 0.523.

5. Discussion

In this section, we discuss the results from the model using some of the samples in the test set, and some features of model that needs to be improved in the future.

The phrase "I love NFT, NFT is the best invention in the world." is labled with the emote "Batchest". In this case, the model made a correct classification. "Batchest" is often used to mock surprise and excitement. For example, it is often used when NFT (Non-fungible token) is mentioned.

The phrase "Imagine buying an NFT worth a million dollor when you can just take a screenshot." is labled with the emote "LULW". In this case, the model made a correct classification. "LULW" is often used to express laughter.

The phrase "Uh this is ill." which was said when the streamer's in-game character was killed is labled with the emote "LULW" because viewers usually laugh at the streamer's bad performance. In this case, the model output the emote "Sadge", which is also acceptable due to the phrase's depiction of sadness.

The phrase "Well I'll tell you what, this whole thing with the virus with BLM with ANTIFA that was to hurt president Trump" which was said by an individual with controversial opinions in a video, and it is labeled with the emote "KKona". "KKona" or "KKonaW" is often used when something related to the US such as hamburgers are mentioned. And it is often used to mock conservatism. In this case, the model made an output of emote "Batchest".

While sentiment analysis is a difficult task even for human, the result of the proposed model shows some level of sentimental understanding of the giving input text. However, there are several drawbacks in the proposed model and its methods. First, the speech transcription of streams is sometimes inaccurate. And in the case when there are several speakers, it is difficult to know which audio input was the viewer reacting to. Thus, more advanced audio-to-text technology that is both accurate and able to differentiate several speakers is needed. Second, the data processing technique needed to be improved because it is hard to know which phrase was the viewers reacting to, which often leads to inaccurate input text and overlapping of the input text. Last but not least, this model only takes text data into account, which means video and audio data are absent in the input data. These data can be learnt from different models and output a single result with ensemble learning.

6. Conclusion

We developed a model using attention-based neural networks for word-level sentiment analysis of speech transcription data. We combined pre-trained Word2vec and positional encoding as the input for the proposed model. While sentiment analysis is a difficult task even for human, the result of the proposed model shows some level of sentimental understanding of the giving input text. We believe this technique can also be implemented in live streaming or even in online lecture to boost interaction between viewers.

References

[1] D. P. Kingma and J. Ba, *Adam: a method for stochastic optimization*, 2014, 10.48550/ARXIV.1412.6980.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, ≪Efficient estimation of word representations in vector space≫, (2013).

[3] *Twitch.tv*, https://www.twitch.tv/p/en/about/, Accessed: 2022-07-31.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, ≪Attention is all you need≫, CoRR **abs/1706.03762** (2017).

[5] I. Yamada, A. Asai, J. Sakuma, H. Shindo, H. Takeda, Y. Takefuji, and Y. Matsumoto, ≪Wikipedia2Vec: an efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia≫, in Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations (2020), pp. 23–30.