CS 263 Final Project Report: Comparison of Different NLP Approaches for Multi-label Tweet Emotions Classification

Tao Liu Amir Dhillon Deniz Orkun Eren Chao Zhang

Abstract

Finding an effective way to determine emotion from social media platforms such as Twitter offers a valuable source of information for assessment. In this paper, we analyse the effectiveness of various approaches, namely, Universal Sentence Encoder, LSTM, Transformer, and BERT on Task E-c of the SemEval 2018 Task 1 competition to determine if they can be used to detect emotions from a tweet. From our observations, the best performance is achieved by utilizing the BERT approach. The code for this project is available at https://colab.research.google.com/drive/1zOZ1XGhPkBLRrkH6JG_LIwXfQjuozqgl?usp=sharing

1. Introduction

Emotion detection is considered as one of the most significant applications for Natural Language Processing. Twitter has increasingly become a commonplace medium for channeling live and up-to-date happenings. These messages are being read by the general public, but more importantly by organizations and agencies that have the ability to act upon such occurrences in a timely and effective manner. This makes performing automatic detection of tweets' emotion especially important. The traditional way for emotion detection is to use multi-class classification on one hot emotion labels. However, realistically, one tweet could have multiple emotions and thus using multi-label tweet emotion identification may present us a better approach. To achieve this goal, this paper examines the effectiveness of different natural language processing (NLP) approaches. The approaches utilised in this study are Universal Sentence Encoder, LSTM, Transformer and BERT. To test the aforementioned NLP approaches, data from the Emotion Tweet Datasets, hosted on CodaLab SemEval 2018 TaskE-c, is utilised (CodaLab). After processing the data and generating features from the text, each model is trained and tested on said data. Additionally, the test performance is noted to perform comparisons.

The organization of this paper is as follows: in Section 2, details about the problem are given; in Section 3, the employed methodologies are explained ; in Section 4, the performance of each approach is compared; in Section 6, the paper is concluded and suggestions about future work are given.

2. Problem Description

As mentioned earlier, the goal of our project was determining the main emotions of a given tweet, which is as described in Task E-c of the SemEval-2018 Task 1: Affect in Tweets competition. Here, the possible emotions for a tweet are: anger, anticipation, disgust, fear, joy, love, optimism, pessimism, sadness, surprise and trust. In total, there are 11 categories that a tweet may fall into.

What makes this task different from most classification problems is that this is a multi-label classification problem, meaning that a tweet can belong to multiple classes at once. For example, a positive tweet can belong to classes of optimism and love. In contrast in most classification tasks, there is only one label for a given data sample. This makes our problem more challenging as common classification approaches cannot be used to solve the problem. This led us to explore other options that are suitable for the task at hand such as using the sigmoid activation at the final layer instead of softmax and using binary cross entropy instead of regular cross entropy loss.

3. Methods

The datasets utilized in this project was obtained from the Codalab Competitions website. As aforementioned, the use of language as a medium of communication extends its bounds for us to measure the intensity of our written emotions and sentiments. The datasets published by the competition includes a training set with 6838 samples, a development set with 886 samples and a test set with 3259 samples. Each sample belongs to a multitude of 11 possible classes. All sets are available in a publicly accessible, downloadable format.

3.1. Universal Sentence Encoder

The Universal Sentence Encoder approach was used in our project as a means for sentence similarity and sentence classification (ana, 2020). As additional background, it is important to note that the Universal Sentence Encoder encodes text into high-dimensional vectors that can be used for text classification, semantic similarity, clustering and other language tasks (uni). What is noteworthy for our group operations is that Universal Sentence Encoding is well suited for Multi-task learning. This implies that generated sentence embeddings can be utilized for tasks like sentiment analysis, text classification, sentence similarity and the results can be fed back to the model to obtain better sentence vectors (sci).

To start using the Universal Sentence Encoder, we imported the necessary libraries which included seaborn, ntlk, and installing transformers. Following this, we loaded the publicly available datasets that are available on the Codalab website. After this, we proceeded with preprocessing our data which included extending contractions, removing hashtags and removing url fragments.

Following preprocessing, we loaded the essential Universal Sentence Encoder's TF Module Hub. By loading the above, we are able to evaluate embeddings.

We then utilized the Random Forest Classifier. Random Forest Classifier is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve accuracy and impose control over over-fitting. After this, the MultiOutputClassifier was implemented, which allowed us to have multiclassification support (sci). This strategy consists of fitting one classifier per target,which allows multiple target variable classifications.

Lastly, the model was evaluated with Jaccard Similarity to identify the similarity score for each label.

3.2. LSTM

The next approach that we employed was termed ASGD Weight-Dropped LSTM (AWD-LSTM) (Stephen Merity, 2017), and it was used to classify and express the mental state that was included in a tweet. AWD-LSTM is a type of recurrent neural network that utilizes DropConnect for the purpose of regularization. Additionally, for the purpose of optimization, AWD-LSTM uses NT-ASGD, which stands for non-monotonically triggered averaged SGD, and returns an average of the weights from the most recent iterations. Variable length backpropagation sequences, variational dropout, embedding dropout, weight tying, independent embedding/hidden size, activation regularization, and temporal activation regularization are among the various regularization procedures employed. By measuring the binary cross-entropy loss, we are able to load and fine-tune the language model.

To apply LSTM for this multi-label emotion classification task, we divided the entire procedure into three sections: EDA (Exploratory data analysis) and data cleaning, ULMFiT (Universal Language Model Fine-tuning for Text Classification) (Jeremy Howard, 2018), Inference and Estimate Accuracy. In the first part, we illustrated the distribution of the eleven required emotions and word counts for each emotion label. Then, special characters, punctuation, accented characters, emojis, and stop words were eliminated from the train, deviation, and test data sets. Additionally, we transformed each word to lowercase and removed the leading space.

In the second part, we utilized the ULMFiT approach to train a language model and then transfer its knowledge to a final classifier. ULMFiT comprises of 3 stages: (1) language model pre-training : we merged the test data set and the deviation data set into a single data set and utilized 30 percent of this data set for the validation set. Using a sequence length of 40 and a batch size of 64, we trained an AWD-LSTM model on the target data set to capture general aspects of the text in different layers; (2) language model fine-tuning: we performed fine-tuning to learn its distributions utilizing discriminative fine-tuning and optimal hyperparameters: learning rates of 1e-2, dropout probability for hidden state between layers of 0.2, dropout probability for LSTM stack input of 0.6, embedding layer dropout probability of 0.1, and hidden-to-hidden weight dropout probability for LSTM layers of 0.5; (3) Classifier fine-tuning: we trained the model with discriminatory learning rates such as 5e-2 and gradual unfreezing. After fine-tuning unfrozen layers for one epoch, we went for next lower layer and continued until all layers are complete and convergence is achieved in the last iteration. In order to avoid overfitting, we only performed training for 5 epochs when unfreezing all the layers. In the third part, we classified the test dataset using the fine-tuned model and estimated the accuracy of the classification using the Jaccard index.

3.3. Transformer

The transformer is a machine learning model that uses a self attention mechanism to give different weights to each part of an input data. Compared to recurrent neural network architectures that process data sequentially, transformer models have the advantage of processing the entire input at once and deciding which parts of it are important for solving the problem at hand. They have proven to be immensely successful in natural language processing (Vaswani et al., 2017) and modern big pre-trained models such as BERT or GPT all utilize the transformer architecture.

In our project, we wanted to train a transformer from scratch to determine the prevalent emotions in a tweet and see if our

implementation could give similar performance to bigger pre-trained models such as BERT. To utilize the transformer, we defined our own transformer block, which consisted of a multi-head attention layer followed by a dropout layer. The output of this was then combined with the input via a residual layer and fed to a layer normalization layer. The output of this was then passed through a feed forward layer and then another dropout layer. This concluded the transformer block. The output of the block was fed to a dense layer after average pooling and dropout then another dense layer with sigmoid activation to generate class probabilities. Another aspect of our model was to generate the embeddings necessary to prepare the input to the transformer. For this, we created our own embedding layers to generate the token and position embeddings. The dimensionality of the embeddings were left as a hyperparameter. As model input, we loaded the dataset and removed characters related to websites and not the tweet itself. We then lemmatized our Tweets using WordNet lemmatizer and tokenized the obtained data. The results was given as input to our model.

To determine the optimal hyperparameters, we performed search through the following hyperparameter space: Embedding dimensions: [10,20,30,40], number of attention heads: [1, 2, 3], dimension of the feed forward layer in the transformer block: [10, 20, 30] an dimension of the feed forward layer between transformer block and output layer: [40, 50, 60]. To determine the best parameters, each combination was trained for 5 epochs. In the end, the best results were obtained with an embedding dimension of 30, 3 attention heads, a feed forward layer with 30 neurons in the transformer block and and a feed forward layer with 50 neurons between the transformer block and the output layer. A model with these parameters was then trained for 7 epochs with the binary cross entropy loss to get the final model.

3.4. BERT

As our next approach, we utilized the pre-trained Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) model to determine if a tweet is related to one or more emotions like anger, and the other ten emotions.

BERT is a transformer-based machine learning model that was developed by Google. It consists of a varying number of encoding layers and self-attention heads. The model was trained on a dataset consisting of 3.3 billion words. During training, two tasks were utilized. The first task was masked language modeling, where 15% of the input tokens were masked and the model was asked to determine what these were. The other task was next sentence prediction, where BERT was asked to predict if a chosen next sentence was probable given the first sentence.

For our implementation, we utilized the base BERT uncased model, which consisted of 12 encoders and 12 self-attention heads. After loading the model, we performed fine-tuning for multi-label emotion tweet identification to achieve optimal performance.

To utilize BERT for multiple-label tweet emotion identification, we first needed to pre-process the data. To do so, we checked if any duplicate entries in the dataset existed. Following this, we changed the text data to a lower case formatchanged all the states abbreviation back to original, removed all characters related to website links and removed special characters used in Twitter such as @s and hashtags. Finally, since emoji could be a huge factor for identifying emotions, we outsourced the emot package and transferred emojis to words for better understanding by the model. As BERT works with fixed-length sequences, we needed to determine the maximum length of the sequences before performing encoding. To do so, we used the BertTokenizer to tokenize each Tweet and observed the lengths of the resulting vectors. It was determined that most vectors have a length of less than 40, so this was chosen as our maximum length. Then, we utilized the BertTokenizer to encode each text to a vector of length 40. This pre-processing was repeated for the given train, validation and test data.

After pre-processing, we moved on to model training. As our pre-trained model, we used the uncased base BERT model with a classification layer at the end. To fine-tune this model, we performed training on the train set using a learning rate of 2e-5 with the Adam algorithm. The learning rate was specifically selected to be low lower to avoid accidental overfitting. Since this is a multi-label classification problem, the sigmoid activation was used at the final layer. After performing training for many epochs, it was observed that the model overfit to the training set. To overcome this problem, training in our experiment was only performed for 7 epochs.

4. Results and Analysis

After training our models, we performed comparisons to determine which one performed the best on the test set. However, due to the multi-label nature of our problem, we could not use commonly used metrics such as accuracy since there are multiple labels per data point and since the majority of the classes is 0 per data sample the accuracy would have given us unreliable results. Instead, we utilized the **Jaccard index** to measure the performance of our models. This metric works by dividing the intersection of the predicted labels and the actual labels by the union of the two. For each data point, we calculate the amount of classes where both the label and the prediction are true (has value 1). We then divide this value by the amount of classes that are true when the prediction and the label are united. We average

this score over all data points to get the model performance. The testing results for all approaches can be found on Table 1.

Method	Test Set Jaccard Score	Prediction time(s)	Parameter Amount
Universal Sentence Encoder	0.383	6.1e-4	1100 trees Height: 38111
LSTM	0.4846	1.8e-3	63,150
Transformer	0.4620	3.6e-4	406,091
BERT	0.5484	2e-3	109,490,699

Table 1. Testing results for different methods

Overall, we can see that all models give satisfactory results, especially when considering that the state of the art approaches to this task manage to obtain a Jaccard Index of around **0.5788** (Baziotis et al., 2018). Since the performance of almost all of our approaches are close to this value, we can determine that our chosen approaches are suitable to perform multi-label classification in order to determine the prevalent emotions in a tweet.

In terms of performance, we can observe that the BERT approach provides the best score. This is expected as BERT is a huge model that has undergone extensive pre-training. Therefore, it is capable of extracting relevant features that can be leveraged to perform multi-label classification. It is followed by the other deep learning approaches: LSTM and transformer. The worst performance is obtained when the Universal Sentence Encoder is utilized. These results are in accordance with the notion that for NLP applications, deep learning models offer better performance and among deep learning models the ones with more parameters tend to perform better.

In terms of average prediction speed per sample, we can see that the Universal Sentence Encoder approach offers the best performance. This is expected since this approach uses a Random Forest model instead of using deep learning based models, which makes it easier to perform predictions. It is followed by the transformer model and the LSTM model. As the transformer does not process data sequentially and instead looks at the entire input, it offers faster performance. Finally, BERT appears to be the slowest approach due to its sheer size.

Finally in terms of the amount of parameters, we can see that BERT has the most amount, followed by the transformer and LSTM. As these are deep learning approaches, their performance relies on the amount of parameters they have so it is expected for these models to be bigger. On the other hand, the Universal Sentence Encoder approach uses a Random Forest model to make predictions, which makes it difficult to compare this approach with the other ones in terms of parameter size as this is not a deep learning model. However, examination of the model reveals that it consists of a total of 1100 trees with a combined height of 38111. As each split in a tree is basically an if statement, this model is much easier to store when compared with the others.

5. Conclusion

In this work, the effectiveness of different approaches to determining the prevalent emotions of a Tweet was analysed. After performing data processing and training, the performance of the created models were tested on the test set provided by the SemEval 2018 Task 1 competition. The best performance was obtained by the BERT approach, which achieved a Jaccard Index score of **0.5484**, which is significantly close to the state-of-the-art. In terms of prediction time and the amount of parameters, the Universal Sentence Encoder offers the best results, with an average prediction time per sample of **6.1e-4** seconds and with a total of **1100 trees** in the final model.

Determining which model to use is dependent on the use case and operation constraints of the application the Tweet analysis will be performed in. If performance is in question, it is best to use the BERT approach. However, if the model is to be deployed in a setting with limited storage capacity such as a smartphone, it makes more sense to utilize the LSTM approach. Finally, if the user wants as close to real-time operation as possible, it is advisable to use the transformer approach as it offers relatively high performance while also offering a high prediction speed.

Overall, the high scores obtained by our models shows that multi-label classification can be performed on Tweets to determine the prevalent emotions. As an interesting research question to continue this work, finding methods that can offer similar performance to the BERT model while requiring less parameters and prediction time can be considered. An approach that comes to mind is to use model distillation as proposed by Hinton et. al. to create a smaller version of the BERT model. (Hinton et al., 2015).

6. Division of Work

Tao Liu: Implement the BERT model, lead team on project written and experiment analysis.

Chao Zhang: Implementation of the LSTM model and writing relevant parts of the report.

Amir Dhillon : Implementation of the Universal Sentence Encoder and writing relevant parts of the report.

Deniz Orkun Eren: Implementation of the transformer from scratch approach and writing the relevant parts of the report.

References

Multiclass and multioutput algorithms. URL https://scikit-learn.org/stable/modules/multiclass.html.

Universal sentence encoder. URL https://tfhub.dev/google/universal-sentence-encoder/4.

4 sentence embedding techniques one should know: With python codes, Dec 2020. URL https://www.analyticsvidhya.com/blog/2020/08/top-4-sentence-embedding-techniques-using-python/.

Baziotis, C., Athanasiou, N., Chronopoulou, A., Kolovou, A., Paraskevopoulos, G., Ellinas, N., Narayanan, S., and Potamianos, A. Ntua-slp at semeval-2018 task 1: Predicting affective content in tweets with deep attentive rnns and transfer learning, 2018. URL https://arxiv.org/abs/1804.06658.

CodaLab. Semeval-2018 task 1: Affect in tweets (ait-2018). URL https://competitions.codalab.org/competitions/17751.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Jeremy Howard, S. R. Universal language model fine-tuning for text classification. 2018.

Stephen Merity, Nitish Shirish Keskar, R. S. Regularizing and optimizing lstm language models. 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. 2017. URL https://arxiv.org/pdf/1706.03762.pdf.