CLPLM: Character Level Pretrained Language Model for Extracting Support Phrases for Sentiment Labels

Raj Ratn Pranesh
Birla Institute of Technology
Mesra, India
raj.ratn18@gmail.com

Sumit Kumar
Birla Institute of Technology
Mesra, India
sumit.atlancey@gmail.com

Ambesh Shekhar
Birla Institute of Technology
Mesra, India
ambesh.sinha@gmail.com

Abstract

In this paper, we have designed a character-level pre-trained language model for extracting support phrases from tweets based on the sentiment label. We also propose a character-level ensemble model designed by properly blending Pre-trained Contextual Embeddings (PCE) models- RoBERTa, BERT, and ALBERT along with Neural network models- RNN, CNN and WaveNet at different stages of the model. For a given tweet and associated sentiment label, our model predicts the span of phrases in a tweet that prompts the particular sentiment in the tweet. In our experiments, we have explored various model architectures and configuration for both single as well as ensemble models. We performed a systematic comparative analysis of all the model's performance based on the Jaccard score obtained. The best performing ensemble model obtained the highest Jaccard scores of 73.5, giving it a relative improvement of 2.4% over the best performing single RoBERTa based character-level model, at 71.5(Jaccard score).

1 Introduction

Sentiment analysis has been a trendy topic for the last some decades. Whether its a graphical image or textual data, all types of an entity consists of something that conveys the sentiment.

With the recent development in machine-learning methods, new innovative and powerful models have developed in the field on natural language processing such heavy pretrained language models.

We designed a novel character-level pretrained language model framework which utilizes the transformers and character-level language models to extract sentiment phrases. The proposed model works in four steps- (i) token-level span prediction using transformer model, (ii) converting token-level representation to character-level representation, (iii) precise character-level span prediction using character-level neural network model, and (iv) retrieving selected sentiment phrases. We also proposed an ensemble character-level model that surpassed the single transformer models. Despite of being a simple idea, ensemble learning has always been successful in several tasks(Zhang and Ma, 2012). We conducted extensive performance analysis of various models and systematically presented our results in the paper.

The primary challenge was in the dataset. As shown in the table 2, for the word such as *likeeee*, the selected sentiment phrase can be *like* or *likee* or *likeee* or *likeeee*. So, by predicting the start and end tokens would not get us to the exact sentiment text span. So, we needed to consider the start and end characters to predict the correct sentiment phrase span. It motivated us to utilize the strong contextual understating of transformers and precise character-level text processing of character-level neural network models for span detection based on different sentiments. Through this study we aim at generating some insights about what exactly a person was thinking while generating any textual content. For example, if a user complains or trolls about a product then there is a need to understand the core reason that dissatisfies that customer about the product. Therefore extracting the phrases that trigger the sentiment can play a significant role in better understanding of the user-generated content.

2 Proposed Transformer Language Models

In this section, we have sequentially discussed and elaborated on the design of the proposed character level pre-trained language models. We first talked about the architecture and working of single Pre-trained Contextual Embeddings (PCE) based character model. Then we talked about the construction

and functioning of the character level ensemble model.

2.1 Character Level Transformer Model

The architecture of a character level pre-trained language model divided into two levels. In the experiment, we used 5-fold cross-validation(4-fold data for training and 1-fold for testing) for each PCE model for comparative analysis, but the overall design and workflow were usual in every single PCE model. So we have provided a generalized pipeline that is compatible with all PCE models.

2.1.1 Level 1

At phase 1, we will discuss the transformer's architecture and training procedure along with the conversion of token level predictions to character level predictions. Following are the steps involved in phase 1:

Language Model Processing: We separately used following Pre-trained Contextual Embedding (PCE) models to build Level 1 model: BERT-base-uncased(Devlin et al., 2018a), BERT-large-uncased-WWM(Devlin et al., 2018a), ALBERT-large-v2, ALBERT-base-v2(Lan et al., 2019), RoBERTa-base(Liu et al., 2019) and RoBERTa-large. All the pre-trained models were fine-tuned[1] on SQuAD2.0(Rajpurkar et al., 2018) dataset. The training data was consist of two components-tweet text(t) and sentiment label(s). For an input in the PCE model, the data was structured by concatenating t and s with separator token and classification token added at the beginning. So for BERT and ALBERT the input sequence was `[CLS]`s`[SEP]`t`[SEP]` whereas for RoBERTa it was `<s>`s`</s></s>`t`</s>`. We extracted $AvgPool$ by performing average pooling and $MaxPool$ by performing max-pooling over the output values from hidden states at the same index of each (n-1) layer(except the embedding layer). The $AvgPool$ and $MaxPool$ were concatenated together to form a combined linear vector which was then passed to two fully connected layers, one of size 1024 with tanh activation and next one of size 128. Followed by a multi-sample dropout(Inoue, 2019) layer and finally a softmax activation layer. The model outputs the probabilities of tokens for being start and end of the sentiment phrases span in the tweets. Custom loss as described here 3.3 was used by modifying the cross-entropy loss. The Jaccard score was calculated on test data to measure the performance of the trained model using the test data as described here 3.1.

Character Level Prediction: Once the transformer finished training we had five train models weight because of 5-fold cross-validation. We took the mean start/end token level predictions of all five models on the whole dataset(train+test)b. To make the token level predictions compatible with character-level neural network model we converted the token level start/end predictions to character level start/end predictions. This is done by firstly removing the padding and sentiment label tokens and then assigning each of the characters in the tweet their respective token probabilities received from the transformer(PCE) model.

2.1.2 Level 2

Character Level Neural Network: We designed three character-level neural network models for processing the character level probabilities, and during the experiment3 each model was tested separately. As shown in figure 1, all three models take three inputs: start/end char probabilities, character stream(character level tokens of input tweet), and sentiment label. The models working are described sequentially. (i) In recurrent neural network (RNN) model, we parallelly passed the start/end char probabilities(2 features) through a BiLSTM layer of size 32 along with the characters and sentiment label were through separate embedding layers of each size 32. The three outputs were then concatenated and passed through two BiLSTM layers with a size of 64 each. The output from two BiLSTM were then concatenated with skip connection and passed through a fully connected layer(size=64) and multi-sample dropout(Inoue, 2019)(p=0.5). Finally, a softmax layer that outputs the character level start and end probabilities. (ii) In convolutional neural networks(CNN) model the start/end char probabilities(2 features) were passed into a 1D convolution layer with batch normalization and the characters and sentiment label were through separate embedding layers of each size 32. All three outputs were concatenated and passed through four 1D convolution layer of size 64 with batch normalization, followed by a fully connected layer(size=64) and multi-sample dropout(Inoue, 2019)(p=0.5) and finally a softmax layer. (iii) In the WaveNet

[1]Pre-trained models are available at https://huggingface.co/models

model, similar to CNN model the start/end probability vector, character vector and sentiment vector after concatenation were passed through three WaveBlocks(size=64) with batch normalization, followed by a fully connected layer(size=64) and multi-sample dropout(Inoue, 2019)(p=0.5) and finally a softmax layer.

Retrieving Prediction: The final step in extracting sentiment phrases from the tweet was to convert the character-level start/end probabilities received from the character-level neural network model into the start/end indexes which represent the span of the selected phrases. Once we have the `start index` and `end index`, our model outputs the selected phrases between the `start index` and `end index`. During training, we found that sometimes the `start index > end index`, means that our model was unable to extract any phrases so in output we return the entire tweet as the selected phrase. During training custom loss 3.3was used, and Jaccard scores3.1 was calculated over test data.

2.2 Character Level Ensemble Model

For building the ensemble model, we stacked together with a set of pre-trained language models and character-level neural network models together blended in such a manner that it outperformed all the baseline single character-level model. As shown in figure2 ensemble design is similar to character level transformer model and divided into following two levels:

2.2.1 Level 1

At level 1, we trained each character level transformer model separately following the prior method and stacked the predictions to form a single start/end character level prediction. Following were the steps:

Ensemble Language Model Processing: After training each character level transformer model using 5-fold cross-validation, we had five trained model weights from each transformer model. We predicted a mean token level start/end probabilities using all five model weights and stored them.

Character Level Prediction: As described in the above character level transformer model, we converted the stored token level start/end probabilities from each transformer model into character level start/end probabilities which were then stacked and concatenated together to form character-level ensemble probabilities($Char_{E1}$). While concatenating, we made sure that the probabilities corresponding to the same tweet were combined together.

2.2.2 Level 2

At level 2, the character-level neural network ensemble comes in play. The character-level ensemble model was trained using a 5-fold cross-validation method. Following are the steps involved in level 2:

Character Level Ensemble Neural Network: At this step, we created an ensemble of character-level neural networks using three models: RNN, CNN, and WaveNet. Each of the three character level neural network parallelly receives three inputs: (i) character-level ensemble start/end probability($Char_{E1}$), (ii) character stream, and (iii) sentiment label. Each outputs the character-level start/end probabilities which were then again concatenated and averaged to form a final character-level ensemble start/end probability($Char_{E2}$)

Retrieving Prediction: As discussed above 2.1.2, the start/end probability($Char_{E2}$) was used to extract the sentiment phrases from the tweet. Custom loss and Jaccard scores were used here.

3 Experiment

In this section, we have discussed the dataset, and it's preprocessing step along with the evaluation matrix used in our experiment. We have sequentially explained the experiment setup and model configuration in detail.

3.1 Evaluation Methods

For the evaluation of our proposed single and ensemble character-level pre-trained language models, we used the **Jaccard score** as evaluation metrics. For a given model in training, in each validation fold, $Jaccardscore$ was calculated using the predicted string and ground-truth string and then at the end of the training mean Jaccard score $Jaccard\ score_{\text{mean}}$ of k validation fold was calculated which was used as the final model score.

$$Jaccard\ score = \frac{1}{n}\sum_{i=1}^{n} jaccard(gt_i, dt_i)$$

where, n is the number of tweets in a set, gt_i is the i^{th} ground truth and dt_i is the i^{th} predicted value.

3.2 Dataset

We used Tweet Sentiment Extraction competition dataset[2] publicly available on the Kaggle[3] website. The dataset is comprised of three parts2- (i) tweets (ii) one of the three sentiment classes(Positive, Negative, and Neutral) associated with each tweet (iii) the phrases/words extracted from each tweet that support the sentiment label in the tweet. The total number of tweets in the dataset were 27,481, consisting of 10,992 neutral tweets/8,244 positive tweets/8,245 negative tweets. For the experiment, we used 5 fold stratified cross-validation in which 4 folds i.e. 80%(21,985 tweets) data was for training and 1 fold i.e. 20%(5,496 tweets) data used for validation.

3.3 Experiment Setting

Model training: At level 1: (i) each model was trained separately using 5-fold cross-validation method for five epochs with the batch size of 64(BERT-base, RoBERTa-base), 32(ALBERT-large, Distil-RoBERTa-base) and 16(BERT-large-WWM, RoBERTa-large), (ii) the tokenized input sequence was truncated or padded up to the max length of 100 and Adam(Kingma and Ba, 2014) optimizer was used with learning rate = 3e-5 and weight decay = 0.001. At level 2: (i) character-level model were trained using 5-fold cross-validation for five epochs, (ii) each model had following configurations: max sequence length = 150, training batch size = 128, validation batch size of 512. Trained for five epochs with learning rate of 5e-3, (iii) character-level ensemble model was trained using 5 fold cross-validation for five epochs with the batch size = 8, learning rate = 5e-4 and character-level model configuration was same.

Custom loss (Jaccard-based Soft Labels): Since Cross-Entropy does not optimize Jaccard directly, we tried different loss functions to penalize far predictions more than close ones. We developed a custom loss function that modifies cross-entropy label smoothing by computing Jaccard on the token level. We then use this new target labels and optimize Kullback–Leibler (KL) divergence(Kullback and Leibler, 1951). Alpha here is a parameter to balance between usual cross-entropy and Jaccard-based labelling. On top of this, we used Stochastic Weight Averaging (SWA)(Izmailov et al., 2018) for better generalization to improve the training stability.

Regularization setting: At level 1: Based on the experiments, the best regularization parameters for every architecture were selected. For improving generalization and accelerating training, we used Multi-Sample Dropout(Inoue, 2019)(MSD). Each Transformer had an MSD layer with a probability of 0.5 except for RoBERTa-large, which was 0.6. We add the Gaussian Noise, with σ = 0.02, to the output layer of the transformers. Based on BERT-paper(Devlin et al., 2018b) we assigned attention probability dropout(0.1) to every transformer. At level 2: As discussed here 2.1.2, for each character-level NN model in the ensemble we used MSD(Inoue, 2019) with the dropout probability value = 0.5.

4 Result and Discussion

In this section, we have discussed the experiment results and presented a performance analysis of character-level transformer models and ensemble models. During our experiment, we combined the level 1 model, with every level 2 model pairwise. As a result, we found out that the RNN based model surpassed other models by achieving a higher Jaccard score during testing. As a result, table1 contains the $Jaccard\ score_{\text{mean}}$ of each transformer combined with the RNN model as Level 2 character-NN model.

According to the table 1, among various single transformer models, the RoBERTa-large model shows good results by having a $Jaccard\ score_{\text{mean}}$ of 71.5%. RoBERTa-base achieved a score of 71.4%, which was very close to RoBERTa-large. Other models also shows promising results, like BERT-large-uncased-WWM and ALBERT-large-V2 had equal $Jaccard\ score_{\text{mean}}$ of 71.1%, whereas the ALBERT-base-V2 and BERT-base-uncased achieved the $Jaccard\ score_{\text{mean}}$ of 70.5% and 71.0% respectively.

Our proposed *ensemble model* outperformed all the single transformer by an average of 2.4%. With a perfect blending of transformer models, the *ensemble model* was able to achieve a $Jaccard\ score_{\text{mean}}$ of 73.5%.

5 Conclusion

In this paper, we proposed a character-level language model consisting of a pre-trained language

[2]Dataset is available at https://www.kaggle.com/c/tweet-sentiment-extraction/data

[3]https://www.kaggle.com/

model and character-level neural network for extracting sentiment support phrases from human-generated tweets based on the associated sentiment labels. We also designed a stacking ensemble model by carefully blending multiple PCE transformer models like BERT, RoBERTa, ALBERT, and character-level neural network models like CNN, RNN, and WaveNet. We conducted a comparative performance analysis of all character-level transformer models and ensemble models. We found that with the optimal combination of transformer models and character-level neural network models, the ensemble architecture generalizes better and outperforms the single transformer models at every level. The ensemble model showed promising results, having a Jaccard Score of 73.5%, which is better than any single transformer model.

Models	*Jaccard Score*$_{mean}$
BERT-base-uncased	71.0
BERT-large-uncased-WWM	71.1
ALBERT-base-V2	70.5
ALBERT-large-V2	71.1
RoBERTa-base	71.4
RoBERTa-large	71.5
Ensemble-Model	**73.5**

Table 1: Models' Scores(in %) on Test Data

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018a. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018b. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Hiroshi Inoue. 2019. Multi-sample dropout for accelerated training and better generalization. *arXiv preprint arXiv:1905.09788*.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Cha Zhang and Yunqian Ma. 2012. *Ensemble machine learning: methods and applications*. Springer.

6 Appendix

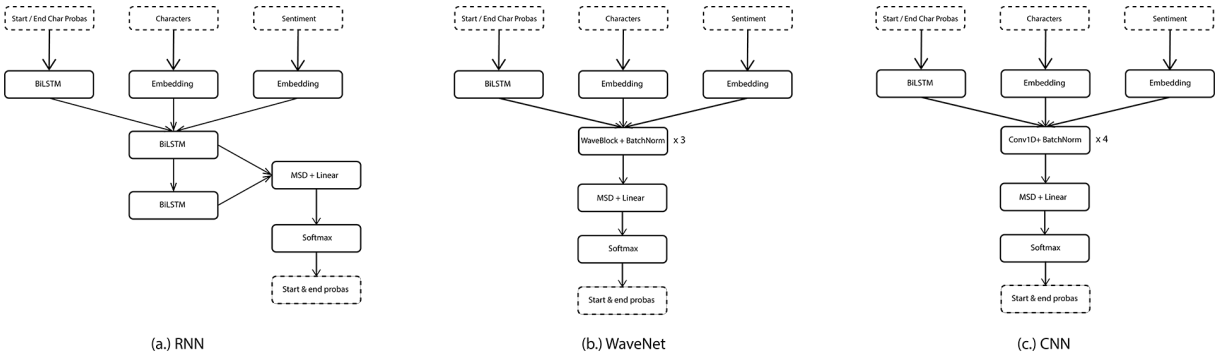

Figure 1: Level 2 Character Level Neural Network (a)Recurrent Neural Network (b)WaveNet (c)Convolution Neural Network

text(given)	**sentiment(given)**	**selected_text(target)**
I really really likeeee the song Love Story by Taylor Swift	positive	likee
I need to get my computer fixed	neutral	I need to get my computer fixed
Sooo SAD I will miss you here in San Diego	negative	Sooo SAD

Table 2: Dataset: texts(given) represents tweets, sentiment(given) represents associated sentiment, selected_text(target) represents extracted phrases.

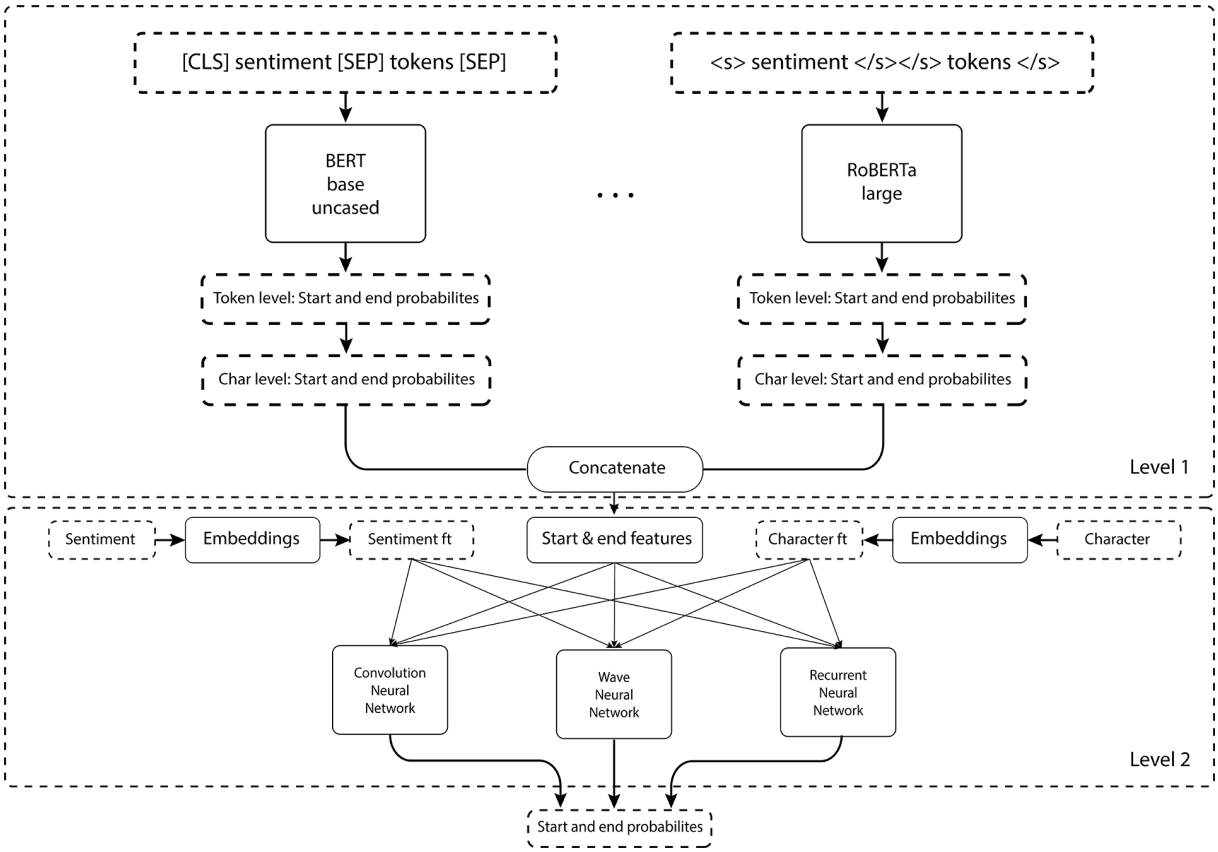

Figure 2: The proposed architecture: Ensemble Model. In level 1, all the transformer models are placed parallelly. In level 2, char-NN models are placed parallelly.