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

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

Sumit Kumar
Birla Institute of Technology
Mesra, India
sumit.atlancey@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. For the past few years, development in methods and techniques has always helped us in understanding the study of sentiment and ways to solve complicated issues that comprised of individual sentiments. We live in an era, where people often share their opinions over the internet, numerous platforms are present on the internet that comprises sentiment triggering data, primarily textual data.

With the recent development in machine-learning methods, new innovative and powerful models has been developed in the field on natural language processing such heavy pretrained language models. Language models are machine learning framework that shows the probability distribution over sequences of contextual data consisting of tokens and characters. These language models have been shown to learn remarkably well on downstream tasks including machine translation, question-answering, text classification, and summarization. Everyday we deal with lots of things that are a component of natural languages and these models have helped a lot in various researches involving natural languages. Language models such as the Bag-of-words model, recurrent neural network, attention networks, and transformers have been always promising in the analysis of textual data.

In the last few years, researchers have devoted their time to the analysis of a content's sentiment over natural languages, demystifying one's perception. In this paper, we tackle a new challenge of extracting sentiment support phrases/words from the tweets based on it's associated sentiment. This task can be framed as span detection task in question-answering. Adaption of state of the art models has given a good head start for the analysis of such problems, but based on the dataset given, we came up with the application of character-level based language model. Since question answering system consists of span detection and classification techniques to find rightful answers for various questions. Based on such methods, we use the span classification technique on our dataset to detect tokens that trigger ones sentiment.

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. In spit of being a simple idea, ensemble learning has always been successful in number of 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 1, for the word such as *likeeee*, the selected sentiment phrase can be *like* or *likee* or *likeee* or *likeeee*. So, by predicting the strat and end tokens would not get us to the exact sentiment text span. So, we needed to consider the start and end characters in order to predict the correct sentiment phrase span. This 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 an product then there is a need to understand the core reason that dissatisfies that customer about the product. Therefore extracting the phrases that triggers the sentiment can play a very important role in better understanding of the user generated content.

The rest of the paper is divided into following major sections: (i) section2 gives a detail description of the proposed models, (ii) section3 provides an overview of dataset used, evaluation method and experiment settings, (iii) section4 contains experiment results and brief discussion, (iv) section5 summarizes related works and finally in (v) section6 we concluded the paper.

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 is divided into 2 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 common 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 level 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 level 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

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

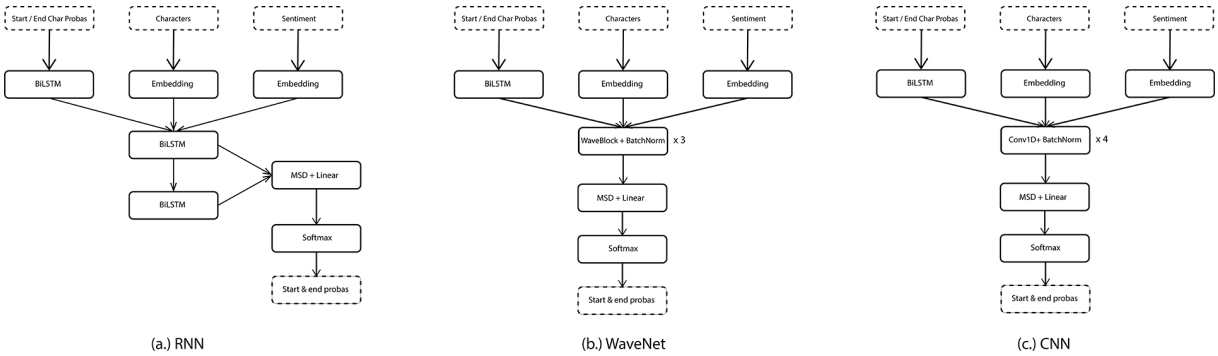

Figure 1: Level 2 Character Level Neural Network (a)Recurrent Neural Network (b)WaveNet (c)Convolution Neural Network

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

In level 2, we discuss the processing of the output obtained from Level 1, feature aggregation, and architecture of character-level neural network models. The character-level models were trained using a 5-fold cross-validation method over the character level start/end predictions. Following are the steps involved in level 2:

Character Level Neural Network: We designed three character-level neural network models for processing the character level probabilities and during experiment 3 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 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 whole tweet as

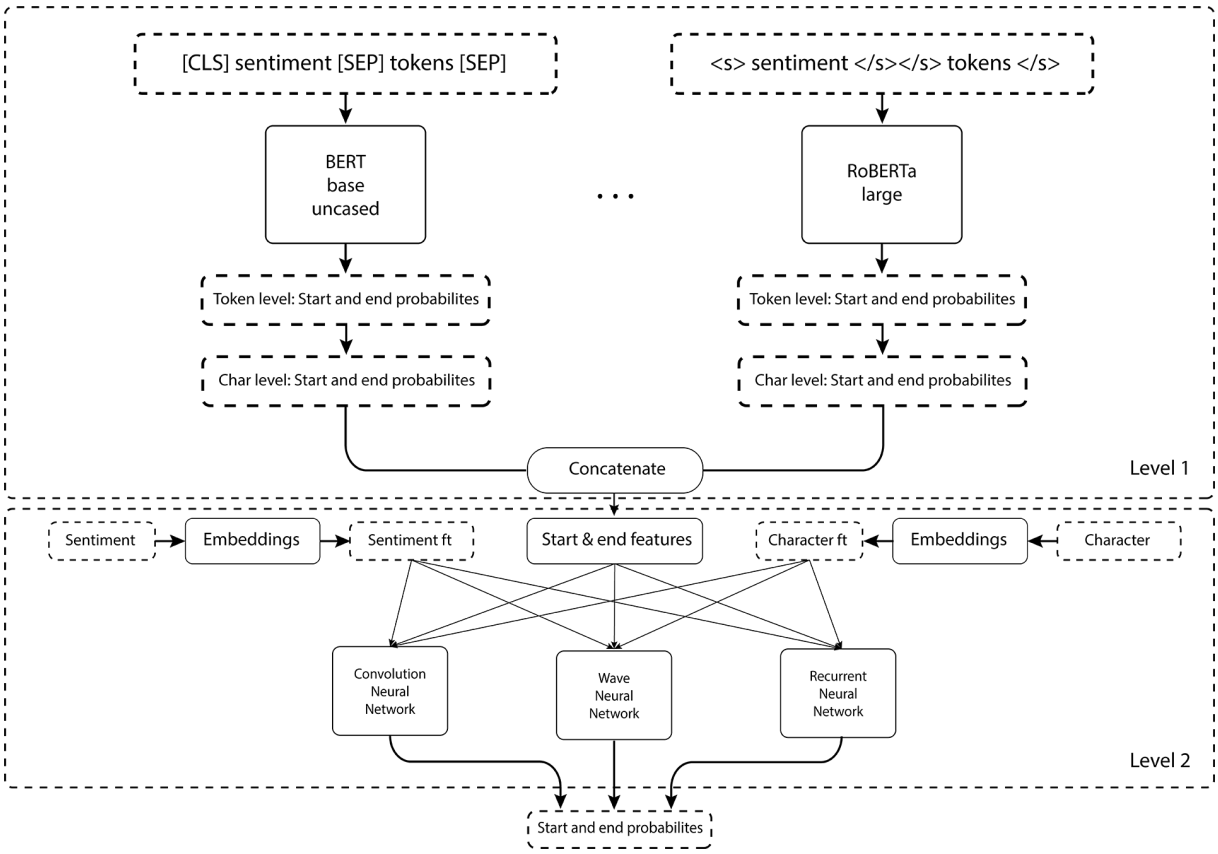

Figure 2: The proposed architecture: Ensemble Model. In level 1, all the transformer models are placed parallelly. In level 2, char-NN models are placed parallelly.

the selected phrase. During training custom loss 3.3 was used and Jaccard score 3.1 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 above-mentioned 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($Char_{E2}$) were 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 **Jaccard score** as evaluation metrics. It was based on the Jaccard index or Jaccard similarity coefficient which is defined as the size of the intersection divided by the size of the union of two label sets. We used the word-level Jaccard score for precise comparison of predicted phrases and the ground truth span. We used stratified k-fold cross-validation for training and evaluation of models. 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. Therefore the mean Jaccard score of k-fold cross-validation is,

$$Jaccard\ score_{\text{mean}} = \sum_{j=1}^{k} \frac{Jaccard\ score_j}{k}$$

3.2 Dataset

We used Tweet Sentiment Extraction competition dataset[2] publicly available on the Kaggle[3] website. The dataset is comprised of three parts1- (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) of the dataset was used for training the model and 1 fold i.e. 20%(5,496 tweets) of the dataset used for the validation. The Cross-validation and Jaccard score 3.1 was calculated for each 5 fold during the experiment.

3.2.1 Dataset Preprocessing

Before conducting the experiments the tweets and ground truth answers were normalized by converting to lowercase representation. The tweets in the dataset were free from any unwanted elements such as username, mentions, links, so no further cleaning was required.

3.3 Experiment Setting

In this section, we have discussed the experiment settings, hyperparameters settings, optimization strategies, and loss function used for model training and performance analysis.

Model training: At level 1: (i) each model was trained separately using 5-fold cross-validation method for 5 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 5 epochs, (ii) each model had following configurations: max sequence length = 150, training batch size = 128, validation batch size of 512. Trained for 5 epochs with learning rate of 5e-3, (iii) character-level ensemble model was trained using 5 fold cross-validation for 5 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 pre-

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

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

text(given)	sentiment(given)	selected_text(target)
I really really like the song Love Story by Taylor Swift	positive	like
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 1: Dataset: texts(given) represents tweets, sentiment(given) represents associated sentiment, selected_text(target) represents extracted phrases.

dictions 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 labeling. On top of this, we used Stochastic Weight Averaging (SWA)(Izmailov et al., 2018) for better generalization to improve the training stability.

$$y_k^{start\_soft} = \alpha y_k^{start\_old} + (1-\alpha)\frac{jaccard(sel\_text, text[k:true\_end])}{\sum jaccard(sel\_text, text[i:true\_end])}$$

$$y_k^{end\_soft} = \alpha y_k^{end\_old} + (1-\alpha)\frac{jaccard(sel\_text, text[true\_start:k])}{\sum jaccard(sel\_text, text[true\_start:i])}$$

Regularization setting: At level 1:Based on 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 = 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, table2 contains the $Jaccard\ score_{\text{mean}}$ of each transformer combined with the RNN model as Level 2 character-NN 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 2: Models' Scores(in %) on Test Data

According to the table 2, 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 perfect blending of transformer models the *ensemble model* was able to achieve a $Jaccard\ score_{\text{mean}}$ of 73.5%.

5 Related Work

Sentiment Analysis: In the past few years, plenty of work has been done in the field of natural language processing which involves analyzing and exploring sentiments hidden in textual representa-

tions. For example, various approaches to build a sentiment classifier have been formulated in the recent past. (Hu and Liu, 2004; Pang and Lee, 2004) used feature-based methods, (Socher et al., 2012, 2013; Tai et al., 2015) used recursive neural networks, (Kim, 2014) used convolution neural network and (Liu et al., 2015) formulated using recurrent neural networks.

Recently with the introduction of large pre-trained language models such as BERT(Devlin et al., 2018b), ALBERT(Lan et al., 2019), RoBERTa(Liu et al., 2019) researchers were able to design and build a state of the art models to perform various tasks related to natural language processing. For instance, in paper(Araci, 2019), the author used a pre-trained language model(BERT) for financial sentiment analysis. In the paper(Yu and Jiang, 2019), the author builds a multimodel inspired by BERT architecture for detecting sentiment polarity on tweets. (Yang et al., 2020) utilized an integrated transformer framework, called a simple transformer, to conduct multi-label sentimental classification by fine-tuning the pre-trained language model on the labeled data.

Character-level models: Efforts have been made in the past for building character-level neural network models. Most of the models follow the translation model architecture design proposed by the authors in this paper(Sundermeyer et al., 2014). Training a recurrent neural net (RNN) over the mini-batches consisting of text sequences, with a short sequence length of around 200 tokens. For capturing the context of a length longer than batch sequence, batches are supplied in sequential order, along with the previous batches hidden states are passed to the current batches. This method is called as "Truncated Back-propagation Through Time" (TBTT). But this method presents complex training processes and performance limitations. To overcome the RNN based model limitations, the author(Al-Rfou et al., 2019) presented a transformer's self-attention layer(Vaswani et al., 2017) based non-recurrent character-level model which surpassed the RNN based character-level models. Hence the character-level RNNs are not even nearly as powerful as the transformer.

Stacking Ensemble Language models: The main idea behind designing an ensemble model is to combine a set of single models in order to obtain a more accurate and reliable model in comparison with other single models. Stacking is as known as a stacked generalization, based on the principle of meta-model learning which involves combining the prediction from different models which results in minimum generalization error. Since each distinct model used in the ensemble have a different architecture which the ensemble model exploits because of the minimal probability of constituent models to collude and give wrong output(Witten and Frank, 2002). For example, in the paper(Malmasi and Dras, 2017), the author proposed a stacked deep learning model for native language identification using textual data. The author in paper(El-Geish, 2020) designed an ensemble model using pre-trained language models for a question-answering task. The best ensemble model configuration was able to beat all the single transformer models.

6 Conclusion

In this paper, we proposed a character-level language model consisting of a pre-trained language 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. The results achieved by our models are far from perfect and can be improved, therefore there is still room for improvement in the proposed method. Future work and possible experiments that can be done such as including new transformers fine-tuned on user-generated content. More extensive pre-processing of the dataset to reduce possible noises and post-processing techniques for improving model accuracy can be also done.

References

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2019. Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3159–3166.

Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.

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*.

Mohamed El-Geish. 2020. Gestalt: a stacking ensemble for squad2. 0. *arXiv preprint arXiv:2004.07067*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

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*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

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*.

Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuan-Jing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2326–2335.

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*.

Shervin Malmasi and Mark Dras. 2017. Native language identification using stacked generalization. *arXiv preprint arXiv:1703.06541*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *arXiv preprint cs/0409058*.

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

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 1201–1211.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. 2014. Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 14–25.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Ian H Witten and Eibe Frank. 2002. Data mining: practical machine learning tools and techniques with java implementations. *Acm Sigmod Record*, 31(1):76–77.

Qiang Yang, Hind Alamro, Somayah Albaradei, Adil Salhi, Xiaoting Lv, Changsheng Ma, Manal Alshehri, Inji Jaber, Faroug Tifratene, Wei Wang, et al. 2020. Senwave: Monitoring the global sentiments under the covid-19 pandemic. *arXiv preprint arXiv:2006.10842*.

Jianfei Yu and Jing Jiang. 2019. Adapting bert for target-oriented multimodal sentiment classification.

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