NLP Report

Rohit Bhal
112073893
rbhal@cs.stonybrook.edu

1 Introduction

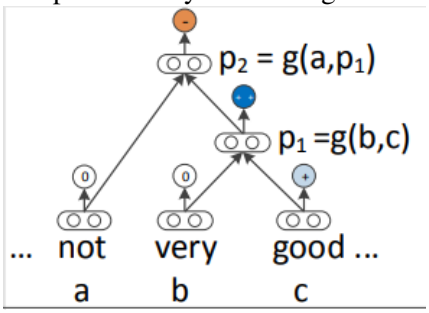
Text classification is an essential task for Natural Language Processing (NLP) with many applications, such as information retrieval, web search, ranking and spam filtering, in which we need to assign single or multiple predefined categories to sequence of text. Semantic word spaces have been very useful but cannot express the meaning of longer phrases. It is very difficult to properly express the sentiment about the larger texts as well as the fine-grained text of a sentence. Sentiment analysis of a sentence is challenging due to the limited contextual information that they normally contain. Today text classification research starts from designing the best feature extractors to choosing the best possible machine learning classifiers. Recently, models based on Neural Networks have become increasingly popular. The deep neural network based approach convention, in most cases, is an input document represented as a sequence of words, and each sequence is then represented as one-hot vector, each word in the sequence is projected into a continuous vector space by multiplying it with weight matrix, forming a sequence of dense, real valued vector. Convolutional Neural Network (CNN) has recently accomplished a remarkable performance on the essentially significant task of sentence classification. In the first stage, each layer will extract features from small overlapping windows of the input sequence and pools over small non overlapping windows by taking the maximum activation in the window. Another successful model applied is Recursive Neural Network (RNN) for NLP and it is confirmed that RNN is able to capture long-term dependencies even in the case of a single layer. Today RNN is the lead approach for many NLP applications. The combination of CNN and RNN also explored for image segmentation tasks In order to extract a feature vector of the sentence, the model relies on an external parser.

2 Summary

2.1

The paper written by Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng and Christopher Potts is on Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. Here, they use the manually created Stanford Sentiment TreeBank to further progress their understanding on compositionality in tasks such as Sentiment detection. To solve this problem, they introduce a new technique Recursive Neural Tensor Network(RNTN) which outperforms all the state of the art techniques in single sentence positive/negative detection from 80% to 85.4%. Also, the accuracy of prediction fine-grained sentiment labels for all phrase reaches 80.7%. Lastly, it is one of the model that accurately captures the effects of negation at various tree levels. The semantic vector spaces introduced were unable to capture the meaning of longer phrases properly and do proper sentimental analysis. Sentiment accuracies even for binary positive/negative classification for single sentences has not exceeded 80% for several years. From business perspective, the analysis of sentence can help us give deep insight about the customer mind and preferences. It allows us to have an overview of the wider public opinion about certain topics. The sentimental analysis is very difficult due to number of reasons such as lack of proper model to describe the meaning of a word, dataset and resources to train the ML model with. it has been difficult to capture sentiment of shorter texts as well as long sentences.

One of the key idea used in the paper is RNTN. The model computes the compositional vector representation of phrases of variable length and syntactic type. The representations are then used as features to classify the phrases. One of the idea was to find a single powerful composition function with a fixed number of parameters.A more direct, possibly multiplicative, interaction would allow the model to have greater interactions between the input vectors. When an n-gram is given to the compositional models, it is parsed into a binary tree and each leaf node, corresponding to a word, is represented as a vector. Recursive neural models will then compute parent vectors in a bottom up fashion using different types of compositionality functions g.



The main idea is to use the same, tensor-based composition function for all nodes. We define the output of a tensor product h $\in R^d$ via the following vectorized notation and the equivalent for a tri gram based model:

$$p_1 = f\left(\begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix}\right)$$

,d is the dimension of each word representation, $V^{[1:d]} \in R^{2dx2dxd}$ is the tensor that defines multiple bilinear forms and W $\in R^{dx2d}$ and V is the vocabulary.

Similarly, p2 can be computed using the below equation:

$$p_2 = f\left(\begin{bmatrix} a \\ p_1 \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} a \\ p_1 \end{bmatrix} + W \begin{bmatrix} a \\ p_1 \end{bmatrix}\right)$$

We can use the word vectors immediately as parameters to optimize and as feature inputs to a softmax classifier. For classification into five classes, we compute the posterior probability over labels given the word vector via:
y^a = softmax(W_sa),
where $W_s \in R^{5xd}$ is the sentiment clasification matrix.
The main task is to compute the hidden vectors $p_i \in R^d$ in a bottom up fashion.

For fine grained analysis, RNTN gets the highest performance, followed by MV-RNN and RNN. This shows that RNN works very well on shorter texts. The combination of the new sentiment treebank and the RNTN pushes the state of the art of Full sentence binary sentimental analysis on short phrases up to 85.4%. RNTN obtains 80.7% accuracy on fine-grained sentiment prediction across all phrases and captures negation of different sentiments and scope more accurately than previous models. I liked the idea in this paper. The idea of breaking down longer phrases into smaller and get the sentimental analysis of each phrases as a feature to predict the sentiment of the whole sentence into positive or negative helps RNN. I also liked the idea of representing the phrase as a binary tree where each word is a leaf node and then from bottom up fashion, we calculate the sentiment of smaller constituents of phrases and use that representation as a feature to predict the label of it's parent and continue doing so till we find the label for the whole sentence. It makes sense to include the sentimental analysis of each word in the phrase as a feature to predict the label of whole sentence as it adds more information about the sentence. One of the extension to this work can be to include an additional RNN layer. It would help more in finding the correct representation of smaller constituents and improved sentimental analysis too. Second extension that I can think of is including character representation as well beside the word representation so as to have more information about the sentiment of the word and improve it a sentence level as well.
Word embedding is one of the idea being used here which I learnt in the class. Here, the RNN models tries to find the correct word representation using word embeddings.

2.2

Sentimental analysis of shorter texts such as twitter and single sentences is challenging. Therefore, the author proposes a new deep convolutional neural network that exploits from character to sentence level information to perform better sentimental analysis of shorter texts. The problem

is challenging because of the limited contextual information they contain and solving this requires techniques that combine the small text context with prior knowledge and uses more than just bag of words for representation.Additionally, to fill the gap of contextual information , it is more suitable to use methods that can exploit prior knowledge from large sets of unlabeled texts.The advent of online social networks has produced a crescent interest on the task of sentiment analysis for short text messages.

One of the key idea is the deep CNN using Character to Sentence Convolutional Neural Network (CharSCNN) to extract relevant features from words and sentences of any size. Given a sentence, CharSCNN computes a score for each sentiment label t $\in$ T. To score a sentence using CharSCNN, it is passed through two convolutional layer which extracts features from character- to sentence level with increasing level of complexity. The first layer transforms word into vectors embeddings which is composed of two items:
1) Word Level Embedding
2) Character Level Embedding
A word w is converted into a vector u_n = [r^{wrd} r^{wch}]. After each word is converted to word and character level embedding. It is again passed through another convolutional layer to get sentence level representation for the given pairs of word and character embedding of each word. Finally, the output from the last convolutional layer is sent through two neural layer to get a score for each sentiment label t $\in$ T. Another key idea is that we can use pre-trained word level embeddings trained using unsupervised techniques which gives a better word representation and helps in improving the CharSCNN model.

The results suggest that the character-level information is not much helpful for sentiment prediction in the SSTb corpus.Regarding the fine-grained sentiment prediction, our approach provides an absolute accuracy improvement of 2.6 over the RNTN approach proposed by (Socher, 2013b), which is the previous best reported result for SSTb. CharSCNN, SCNN and Socher et al.s RNTN have similar accuracy performance for binary sentiment prediction. Initializing word-embeddings using unsupervised pre-training gives an absolute accuracy increase of around 1.5 when compared to randomly initializing the vectors.

I loved the idea of using the word embeddings to extract character level information to improve sentimental analysis over fine grained sentiment prediction. It is a very good approach to first use word embedding and character level embeddings to extract features related to them and feed it to the next layer as a feature to get a better representation and prediction of sentimental analysis of the whole sentence. Probably, the word and character level representation holds information vital which can improves the model to accurately predict the label for each sentence. One of the improvement that I think could be done would be to represent the sentence in binary tree form and learn character and word level embeddings using the previous words representation as an input too. Another one would be to used domain specific unlabeled text to have better word representation to improve the accuracy. Also, we could add analyze more about the character level embeddings and add some features to improve the model. Few of the idea being used in this paper that I learnt in the class are word embeddings as well as CNN layer.

2.3

Learning good vector representations for sentences is a challenging task and an ongoing research area. Sentiment analysis for short texts becomes a challenge because of the limit of the contextual information they usually contain. Moreover, learning long-term dependencies with gradient descent is difficult in neural network language model because of the vanishing gradients problem. In this paper, we propose ConvLstm, neural network architecture that employs Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) on top of pre-trained word vectors. ConvLstm exploit LSTM as a substitute of pooling layer in CNN to reduce the loss of detailed local information and capture long term dependencies in sequence of sentences.
Recently, models based on Neural Networks have become increasingly popular. Convolutional Neural Network (CNN) has recently accomplished a

remarkable performance on the essentially significant task of sentence classification. However, these models require professionals to specify an exact model architecture and set accompanying hyper-parameters, including the filter region size. We observed that employing convolutional to capture long term dependencies requires many layers, because of the locality of the convolutional and pooling.As the length of the input grows, this become crucial. We propose a neural language model that leverages both convolutional and recurrent layers to efficiently perform text classification tasks. The work is also inspired from the fact that recurrent layers are able to capture long-term dependences with one single layer. We utilize a recurrent layer LSTM as substitutes for the pooling layer in order to reduce the loss of detailed local information and capture long-term dependencies.

One of the key idea is that recurrent layers are able to capture long-term dependences with one single layer. We use recurrent layer LSTM as substitutes in order to reduce the loss of detailed local information and capture long-term dependencies. The other one is the use of both convolutional and recurrent layers to efficiently perform the sentiment classification tasks. Lastly, using pre-trained word embeddings obtained by unsupervised learning on large number of texts. Using LSTM is a good idea as it captures the information about the past output which can help in efficiently prediction the label on the future words in a sentence. The convolutional layer can extract high-level features from input sequence efficiently.However, it requires many layers of CNN to capture long-term dependencies. Therefore, it is a good way to substitute number of CNN layers with LSTM to capture those features as well as use the past word's label to correctly predict the label for the sentence. The use of pre-trained embeddings on large unlabeled text as well as domain text is helpful in finding better word representation which in turn helps to extract useful features to accurately predict the label for the given words and sentence.
The CNN used in the model is a slight variant of CNN. Let x_i to be the k-dimensional word vector corresponding to the i-th word in the sentence. A sentence of length n padded where necessary is represented as:

$$x_1 = x_1 \oplus x_2 \oplus \ldots \oplus x_n,$$

where $\oplus$ is the concatenation operator and the convolutional operation consists of filter w $\in R^{hk}$, which is applied to a window of h words to produce a new feature.
For instance, feature c_i is generated from a window of words $X_{i:i+h-l}$ by:

$$c_i = f(w.x_{i:i+h-1} + b).$$

The feature is generated by passing the sequence to a nonlinear function such as tanh, ReLU with a bias term and the filter w.
The feature sequence is feeded to a single LSTM layer to output the label.The memory cell is consist of four main components:input, output, forget gates and candidate memory cell.

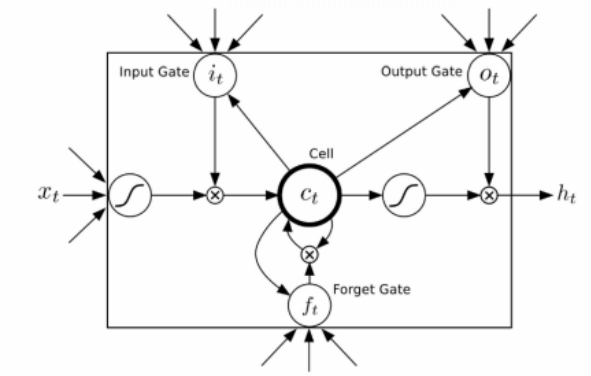


Figure 3. LSTM framework shows the network using only the last hidden dimension for sentiment prediction.

We use pre-trained word embedding vectors obtained from an unsupervised neural language model to improve the accuracy of the model. It can capture syntactic and semantic information, which are very important to sentiment analysis.

The ConvLstm model archives comparable performances with significantly less parameters. We achieved better results compared to convolutional only models. Also using LSTM as an alternative for the pooling layers in CNN gives the model enhancement to capture long-term dependencies. The ConvLSTM achieves 88.3% on full sentence binary sentiment classification on SSTB dataset and 47.5% on fine grained sentiment analysis.

I loved the idea of using LSTM layer instead of sever CNN layers to extract features from the past labels in a given sentence. LSTM works on the principle of using the past output

and this is a very good way to solve sentiment analysis as the analysis of the future word as well as the sentence depends on the past words labels too. Also, the idea of using pre-trained model is once again proved effective. One of the extension to this work can be done by using bi-directional LSTM instead of simple LSTM which can extract features of future words as well. Another extension could be to add character level embedding as well to the input layers so that the network could learn character level features which could help in correctly predicting the label for the sentence. Also, the idea word embedding learnt in the class is again being used in this model. Also, LSTM are being used to extract features in a sentence which is a good idea as LSTM stores information regarding past outputs.

3 Discussion

The first paper is the paper on which our project idea is based upon. The paper considers word embedding as well as previous word label for phrase and sentiment analysis. But it has not considered taking into account of character level word embeddings to extract features related to it. Also, it could be further improved by taking pre-trained word embedding vector on large amount of unlabeled text using unsupervised learning. Another way to improve would be to add another neural layer to the network to extract more complex features.
The second paper outperforms the first paper by using CNN which takes input of word embedding and character embedding to extract features from character- level to- sentence level. The idea is good and it could be extended in the first paper. Also, it has introduced us to the idea of using word embedding as a pre-trained model by using unsupervised ML. The model is not taking into consideration the previous word's label in the current input. Therefore, it could be improved by using LSTM as one of the extra or substitute layer in the architecture as it is better at learning features based on past output which is very important in sentiment analysis. We will try to use character embeddings derived from word embeddings as well as use pre-trained word embedding vector to improve the baseline score in the first paper.
The third paper introduces the latest technique being used in NLP, i.e., LSTM. The model architecture outperforms the first paper both in terms of fine grained and binary sentence sentiment analysis. It introduces us to the technique of using LSTM instead of multiple CNN layers to extract long term features in the sentence. The LSTM is used after CNN layer which extracts the features of the sentence. It gives us a good idea of using a single LSTM after a CNN layer instead of multiple layers. One of the shortcomings is that the paper didn't consider taking character level embedding into the input which could give the network some character level features too. Thus, it could have improved the model for sentimental analysis.

We can see that for shorter texts, having character level embedding improves the accuracy whereas CNN model doesn't work well for long texts. We will try to incorporate the idea of using pre-trained models in our model, taking character level embedding as one of the input, using LSTM and extracting sentence level features on the first paper and improve the baseline score.

4 References

1.Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng and Christopher Potts 2013.Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank .

2.Ccero Nogueira dos Santos and Mara Gatti 2014.Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts .

3.Abdalraouf Hassan and Ausif Mahmood 2017.Deep Learning Approach for Sentiment Analysis of Short Texts.