Sentiment Analysis By Using Different Structures on IMDB Dataset

Qiancheng Wu
Department of Computer Science
UC, Irvine
Irvine, United States
qiancw1@uci.edu

Zerui Li
Department of Informatics
UC, Irvine
Irvine, United States
zeruil1@uci.edu

Yi Zhou
Department of Informatics
UC, Irvine
Irvine, United States
zhouy46@uci.edu

Abstract

In this report, we test different ways to do sentiment analysis on classic IMDB dataset which provides a set of 25,000 highly polar movie reviews for training, and 25,000 for testing. We try to combine different embedders such as Glove, ELMo with different architectures like GRU, LSTM, Bi-LSTM, CNN and so on. We also try to train our own embedders on the dataset, and it seems perform better than some classic embedders trained by others like Glove. Besides, we also try to re-implement a new architecture called Deep Pyramid Convolutional Neural Network(DPCNN) on IMDB dataset which is not used by others before. The DPCNN doesn't give us the high accuracy as we expect, but it shows its potential on doing sentiment analysis task.

1 Introduction

Sentiment analysis, as a classic natural language processing(NLP) problem, refers to using the text analysis, computational linguistics to systematically identify and quantify affective states of subjective information. Its widely used for customer material such as survey and feedback. Basic sentiment analysis includes classifying the polarity of a sentence to figure out whether its positive or negative or its emotional states such as happy, sad, angry.

In fact, What other people think has always been an important piece of information for most of us during the decision-making process. Long before the invention of the World Wide Web, Its quite common for us to ask our friends to recommend a good restaurant, a fantastic tourist attraction or a good job opportunity. That is the base of the recommendation system. Nowadays, lots of companies such as Yelp, Netflix, IMDB are trying to using a large amount of data in order to provide their customers with a more accurate recommendation. However, its not quite that common when those data are regular and easy to be understood by a computer. In fact, most of the data those companies collect from social media like Twitter, are just sentences like Jane Austens books madden me so much that I cant conceal frenzy from reader which is quite difficult for a computer to understand whether this person like Jane Austen or not. The early study used keywords flagged by volunteers manually to determine the emotional-affective of a sentence. Whereas, such method is unlikely to cover all the keywords when datasets are becoming larger and sentences become more complex.

During recent years, with the improvement of machine learning especially deep neural network, more and more advanced techniques have been proved useful to teach a computer to learn what a natural sentence means. In this paper, we use a classic dataset provided by IMDB, which includes peoples comments to movies, to test the performance of some classic deep learning NLP structure such as LSTM, RNN, CNN on this dataset. We also try different embedders such as Glove, Elmo and even try to train embedder from scratch on the dataset. Besides, we also use an advanced structure called Deep Pyramid Convolutional Neural Network(DPCNN), given by Tong Zhang et al. on ACL 2017(8) which has been proved efficient on many other datasets, on IMDB dataset.

2 Related Work

Although the area of sentiment analysis has enjoyed a huge burst of research activity, their has been a steady under current of interest for quite a while. One could count early projects on beliefs as forerunners of this area.(2)(3) Later work focused mostly on interpretation of metaphor, narrative, point of view, affect et al(4)(5)(6). The year 2001 seems to mark the beginning of widespread awareness of this problem, the number of paper about this area rises at that time, and subsequently

there have been literally hundreds of paper published on the subject(1).

Recent years, with the rising of deep neural network, people gradually use different kinds of deep learning architecture to improve the accuracy and efficiency of doing a sentiment analysis task. Mathieu Clich(7) from Bloomberg got a F1 score of 0.685, which is currently ranked 1st , on Twitter dataset by combing LSTMs and CNNs. In 2017, Tong Zhang from Tecent AI lab(8), promoted DPCNN model which shows the potential of shallow convolutional neural network. In 2018, Baoxin Wang from joint library of HIT and iFLYTEK, prompted Disconnected Recurrent Neural network, which improves the performance of previous model on Amazon dataset by nearly 3% accuracy(9). Jeremy Howard et al. from fast.ai prompted ULMFit model in 2018, which get the highest accuracy on Yelp and Imdb dataset(10).

3 Classic Model

Our approaches include word embedding choices, CNN and LSTM. For word embeddings, we aim to pick some of the existing techniques and compare their performance. GloVe and Elmo are considered for now and we may have more to evaluate. BERT, FastText and Word2Vec can be used for further comparison. However, we focus on GloVe and Elmo first since the comparison between models is the main concern for us in this task. For convolutional neural networks, our model is depicted as Fig. 1. The input for the network is each review or tweet in the dataset. Each word is a d-dimension word embedding and hence the input is a matrix of size l * d where l is the number of words. We use padding so that the number of words is fixed in each input. In our case, l is 200 and d is 200, too. As Fig. 1 shows, the kernel has size s * d. In our case, we have 3 kernel sizes 2, 3 and 4. Multiple kernels are used so that different features can be learned, where the number of filters is set to 200 in our code. Then we apply a max-pooling operation to each convolution to extract the most important feature. Finally, the concatenation operation is performed to combine all the polled value to a single vector of size s * number of filters, which is 600 in our case.

For LSTM, we use bidirectional LSTM in the model and a single forward LSTM as the base model. Fig. 2 shows a single forward layer of LSTM. For each embedded word in the sequence, an LSTM unit is applied and calculate the hidden state by the embedding and previous cell hidden state. For our bidirectional model, a backward layer is added and the output in the combined output of both directions. For the continuing project, we are also considering using GRU in replace of LSTM.

3.1 DataSet

We used the datasets from IMDB Review Dataset, this dataset contains movie reviews along with their associated binary sentiment polarity labels. It is intended to serve as a benchmark for sentiment classification. The core dataset contains 50,000 reviews split evenly into 25k train and 25k test sets. The overall distribution of labels is balanced (25k pos and 25k neg). It also includes an additional 50,000 unlabeled documents for unsupervised learning. Furthermore, in the labeled train/test sets, a negative review has a score less than 4 out of 10, and a positive review has a score larger than 7 out of 10. Thus reviews with more neutral ratings are not included in the train/test sets.(11)

3.2 Experiments

Our experiments consider the performance of several models and configurations. We have used LSTM, Bidirectional LSTM, Glove as Embedder, CNN, GRU and Bidirectional LSTM + CNN to evaluate their performance on the IMDB reviews dataset with Pytorch. In this experiment if we do not mention the embedder specifically that means it trains its own embedder on the dataset.

Models	Accuracy	F1 Score
LSTM	0.8605	0.8529
Bi-LSTM	0.7448	0.752
Glove	0.6052	0.5929
GRU	0.7336	0.7318
CNN	0.8073	0.8178
CNN+Bi-LSTM	0.8735	0.852
ELMO+LSTM	0.8621	0.8703
ELMO+GRU+MAX+AVG	0.8900	0.8968

At first, for the LSTM model, the parameters we used is embedding-size = 200, max-words = 200, dropout = 0.5. Further, we have implemented Bi-LSTM, GRU, Glove, CNN and CNN + LSTM with the same parameters as above. The procedure to train these models is basically the same. First, we need to import the IMDB dataset and di-

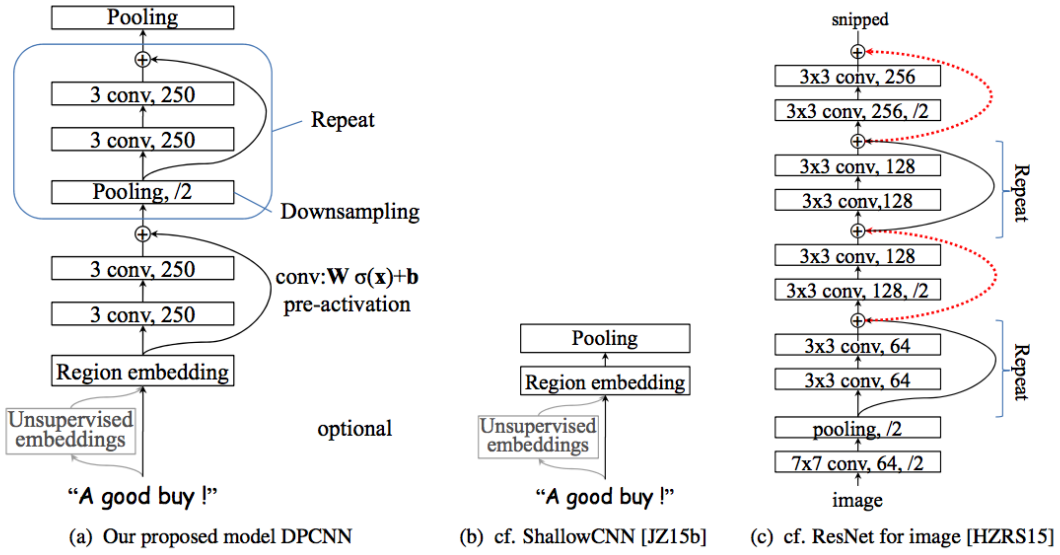

Figure 1: DPCNN Architeture

vide them into 25000 training samples and 25000 test samples. And we manage to use the index to represent each word with labels. Next, we pad sequences to the same length. Whats more, the next part is to train and configure the model. We implement the sequential model and set the corresponding parameters same as stated previously. Besides, about the Glove embedding, we first download the glove.6B to our environment and process them line by line to get embedding index. And with the help of training data, we could form an embedding matrix. As you could see from the table below, the performance of single Glove is not satisfying and we think it may because of the overfitting and this embedding method is not fit for our dataset.

For the complicated model CNN + LSTM, we mainly used model.add to add multiple layers to get the model. Additionally, for the CNN + LSTM model, we use max-words-filter-sizes to be the parameter as pool size in order to maximize the pool size. Besides, we use ReLU and sigmoid as our activation function in core dense layers. Moreover, to prevent overfitting, we utilize dropout to drop some neural units. Additionally, we also add flatten to this model at the end. And we also draw a figure of the process of our model.

For the Elmo embedding model, it uses a pre-trained language model and generate a weighted average LSTM outputs of every level according to the input sequence. Since it takes a pre-trained language model and combines two vector of 512, the output vector size is constrined to 1024. Therefore in this model we have a different embedding size of 1024, while keeping the same sequence length as 200. The Elmo + LSTM model, which is simply ELMO embedding plus one LSTM and a full connected layer, does not improve the performance of the original LSTM model. However, the ELMO + GRU model outperforms all the model. Besides a GRU and ELmo embedding layer, this model take the output vectors and use max pooling and average pooling, respectively. The pooled vectors are combined before sent to the final dense layer.

3.3 Conclusion

From the result table, we can easily find that LSTM model performs much better than simple GRU or CNN model, whereas when we try a Bi-LSTM the accuracy decreases a lot. By changing different embedders, we conclude that there is no guarantee that the embeddders trained by others could perform better than embedders trained on specific task. Glove vectors perform much worse than we expected before on this task. Such phenomenon proves "No Free Lunch Theorm"NFL convincingly. During the process of training model by using Glove, we also find the model can get 100% accuracy on training set while get the 60% accuracy on test set which implies that Glove embedding can cause overfitting more frequently. However, not all pre-traiend embedders perform bad on this task, by combining Elmo and GRU we gain the accuracy of 89% on test set.

4 Deep Pyramid Convolutional Neural Network

During the process of this project, we also find an interesting architecture called Deep Pyramid Convolutional Neural Network(DPCNN), prompted by Tong Zhang (8) in ACL 2017. During the final week of this project, we try to reimplement such architecture and use it on our IMDB dataset.

4.1 Model Description

DPCNN is a low-complexity word-level deep convolutional neural network architecture for text categorization. In the literature, several deep and complex neural networks have been propoesed for sentiment analysis task. But, people graudally realize the fact that the associated computational complexity increases as the networks go deeper and moreover, it's shown that shallow word-level CNNs are more accurate and much faster than the state-of-the-art character-level CNNs. Based on such fact, Rie Johnson and Tong Zhang prompted DPCNN architecture in 2017.

The key features of DPCNN are as follows:

- Downsampling without increasing the number of feature maps.
- Shortcut connections with pre-activation and identity mapping for enabling training of deep networks.
- Text region embedding enhanced with unsupervised embeddings (embeddings trained in an unsupervised manner) (Johnson and Zhang, 2015b) for improving accuracy.

The concrete architecture of DPCNN is shown in figure 4.1, that is what we refer to in this project.

As we can see in figure, downsampling with the number of feature maps fixed After each convolution block, we perform max-pooling with size 3 and stride 2. That is, the pooling layer produces a new internal representation of a document by taking the component-wise maximum over 3 contiguous internal vectors, representing 3 overlapping text regions, but it does this only for every other possible triplet (stride 2) instead of all the possible triplets (stride 1). This 2-stride downsampling reduces the size of the internal representation of each document by half.

According to Zhang's paper, DOCNN fixes the number of feature mapsthe total computation time is bounded by a constant twice the computation time of a single block, which makes our deep pyramid networks computationally attractive.

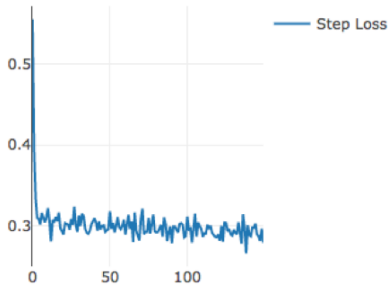

Figure 2: LSTM Loss

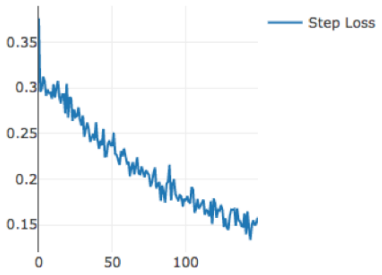

Figure 3: DPCNN Loss

Figure 4: Step Loss

4.2 Experiment

We build our DPCNN model by using Pytorch on IMDB dataset. It seems that no one used such dataset to train DPCNN before, so we are not sure whether we are in the right way.

We use Pytorch to build the DPCNN architecture according to the previous paper and then we load our IMDB data can convert it to a 25000 x 250 matrix, with the size of vocabulary equals 80160 and when the length of a sentence is less than 250, we use 0 to pad. And then, according to the paper, we use a 250-dimension, which initialized randomly, word embedder to embed each word in sentences. We choose the cross-entropy loss as our loss function.

We choose to train DPCNN in 5 epochs, and it gain 93.47% accuracy on training dataset, and 84.31% accuracy on test data set. Its accuracy on test data set is not that high as we expect, that might be because the number of epochs we choose is a little bit small, we also draw the plot to show the relationship between our loss and the number of steps.

As we can see the loss of DPCNN decreases more faster than LSTM, and it seems that LSTM model converge to a local mimum while the DPCNN doesn't which shows the great potential of DPCNN model.

5 Conclusions and Anticipated Problems

For the classic models, from our multiple training and comparisons, we could find that CNN + Bi-LSTM and ELMO + GRU has the best performance compared to the others. What's more, the test score for ELMO + GRU is approaching 0.9 which is satisfying. Besides, the failure of glove embedding on this task proves the "No Free Lunch Theorem" in a convincing way. Before this task, we thought glove embedder, which is trained on billions of sentences can undoubtedly do a good job on any datasets. However, its performance on validation and test datasets are both around 0.6. We consider that it may result from our small epochs and small pre-training text for Glove. And the result shows that training our own embedders from the scratch can usually gain a higher accuracy.

In this project, we also re-implement DPCNN architecture by training this model on IMDB dataset which is not used by others before. Although, it doesn't give us the high accuracy as we expect and it takes quite a long time to train, but it shows its potential to do sentiment analysis for it has a high speed to decrease its loss . Besides, this model is easy to be constructed from the scratch.

Sentiment analysis, as a classic problems, deserves our effort to do further research in the future. We believe, some advanced models such as ULMFiT or MT-DNN can also does this task on IMDB dataset well.

References

[1] . B. Pang and L. Lee, Opinion Mining and Sentiment Analysis, Foundations and Trends in Information Retrieval, vol. 2, nos. 12, 2008, pp. 1135.

[2] Jaime Carbonell. Subjective Understanding: Computer Models of Belief Systems. PhD thesis, Yale, 1979.

[3] Yorick Wilks and Janusz Bien. Beliefs, points of view and multiple environments. In Proceedings of the international NATO symposium on artificial and human intelligence, pages 147171, New York, NY, USA, 1984. Elsevier North-Holland, Inc.

[4] Marti Hearst. Direction-based text interpretation as an information access refinement. In Paul Jacobs, editor, Text-Based Intelligent Systems, pages 257274. Lawrence Erlbaum Associates, 1992.

[5] Alison Huettner and Pero Subasic. Fuzzy typing for document management. In ACL 2000 Companion Volume: Tutorial Abstracts and Demonstration Notes, pages 2627, 2000.

[6] Mark Kantrowitz. Method and apparatus for analyzing affect and emotion in text. U.S. Patent 6622140, 2003. Patent filed in November 2000.

[7] Cliche M. BB_twtr at SemEval-2017 task 4: twitter sentiment analysis with CNNs and LSTMs[J]. arXiv preprint arXiv:1704.06125, 2017.

[8] Johnson R, Zhang T. Deep pyramid convolutional neural networks for text categorization[C]//Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017, 1: 562-570.

[9] Wang B. Disconnected Recurrent Neural Networks for Text Categorization[C]//Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2018, 1: 2311-2320.

[10] Howard J, Ruder S. Universal language model fine-tuning for text classification[J]. arXiv preprint arXiv:1801.06146, 2018.

[11] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).