MLP Coursework 4: Sentiment analysis for movie reviews

G20 - s1734034, s1771650, s1717977

Abstract

In the past works, we have already built our basic model for sentiment analysis of movie reviews. In this paper, we investigate more to improve the model. We expand the dataset we used, update the structure of the model with more hidden layers and bi-directional LSTM, evaluate the hyper-parameters dropout, and also explore the utility of attention model and batch normalization. And with these experiments, our model ameliorated a lot and got a better classification result.

1. Introduction

Sentiment analysis or opinion mining is the computational study of people's opinions, appraisals, attitudes, and emotions toward entities, individuals, issues, events, topics and their attributes. As for analysis the movie reviews, the task usually involves detecting whether a piece of review expresses a POSITIVE or a NEGATIVE sentiment about a movie. There is a demand for advanced analysis technique because of the online information explosion. With the rise of social media, increasingly number of movie reviews appear, which is hard for researchers or other managers to read through. At the same time, it is difficult to identify the review's polarity. When people write a review, there is no specific feature to present whether it is a good comment or a criticism while theatres or movie companies only care about the ratio of positive reviews. If we want to get the ratio automatically, the first job is to classify the reviews into different categories. So, our goal of this project is to build a neural network to mining the sentiment attributes from the reviews automatically. In last coursework, we have already established a basic model with LSTMs(Long Short-Term Memory networks). This paper will further describe this original deep learning model and focused more on the improvement of the model.

The basic cells of the network are still LSTM. Firstly, we modify the mistake of dropout. We explore the performance of dropout after recent deeper studies and found that the dropout used in the previous work is not the proper way it should be(Vu Pham & Kermorvant, 2014), so we change the location of it (We implement the dropout inside LSTM cells before, now we moved it behind the LSTM layer, which is outside the LSTM cell). And we found the most suitable one after testing the influence of different dropout ratio. Secondly, we add a reversed sequence layer after the forward layer, which means we use a bidirectional LSTM to retrain the data and see if it is better. Thirdly, we add more hidden layers to improve network's performance. Learning networks with more hidden layers seem usually performed better than the simpler ones(Kai Sheng Tai, 2015), so we add more layers after the first LSTM layer, and it finds out if this is useful. Finally, we use attention model(Karol Gregor, 2015) and batch normalization. We add a weight for each word in the movie reviews and normalization the LSTM output before feeding into next layer. Then, we retrain the model and get a final outcome of our experiment. The result is substantial, the accuracy on test set increased 6.2%.

The first part of this paper is the overview of this task. Then, we will introduce the new methods we used in this project. Next, we will illustrate the experiments specifically and report the results of different experiments. In the end, we will discuss other related works and give a final conclusion.

2. Methodology

For this project,we firstly expanded our dataset, and the new methods we use for this task is attention model and batch normalization.

Dataset
The first considerations when dealing with text data is preprocessing. Same as before, the dataset we used is Large Movie Review dataset provided by Stanford(Maas et al., 2011).This time, we enlarge our dataset with the test data set of Stanford, and also reprocess all the data. We replace each word with a corresponding number and delete the words that the frequency is under 5. Then we still use the Word2Vec to set word embedding for each word. When we built our baseline experiment in former coursework, we process the train data of the dataset and redivided it into different parts for training, validation and test. There are 50000 reviews for the whole dataset. We set the first 40000 reviews as training data, next 5000 reviews as development data and the final 5000 files as test data to report the performance of our final model. And we also fix the length of each reviews to be 500 words.

Attention model
Attention in Neural Networks has a long history, particularly in image recognition. (Denil et al., 2011).But only recently, there are some attention mechanisms made their way into recurrent neural networks architectures which are typically used in NLP (and increasingly also in vision). The most popular application area for attention in NLP is the Neural Machine Translation(Luong & Hieu Pham, 2015), which is not only relying on things like n-gram counts and trying to capture the higher-level meaning of a text. Here

we use another type of attention called hierarchical attention(Yang et al., 2016).

The basic idea of attention for NLP is that when we doing a classification task for a text, each word in the text will contribute to the final recognition result. And each word will also have different influence for the result, so we need to compute an alpha score for each word and combine them into the results.The overall architecture of the Hierarchical Attention Network (HAN) is shown in Fig. 1. It consists of several parts: a word sequence encoder, a word-level attention layer, a sentence encoder and a sentence-level attention layer. But here we regard each document as one sentence, so we only use this model for the word level.

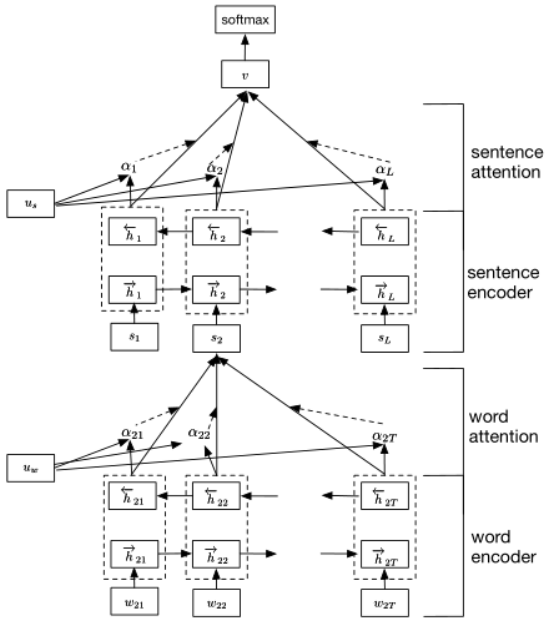

Figure 1. Hierarchical Attention Network, from Hierarchical Attention Networks for Document Classification

Given a sentence with embedded words $W_{it}, t \in [0, T]$, we first use a bidirectional LSTM to get annotations of words by summarizing information from both directions for words and incorporate the contextual information in the annotation. The bidirectional LSTM contains the forward LSTM $\overrightarrow{f}$ which reads the sentence S_i from W_{i0} to W_{iT} and a backward $\overleftarrow{f}$ which reads from W_{iT} to W_{i0}:

$$\overrightarrow{h}_{it} = \overrightarrow{LSTM}(W_{it}) \in [0, T], \tag{1}$$

$$\overleftarrow{h}_{it} = \overleftarrow{LSTM}(W_{it}) \in [T, 0], \tag{2}$$

We obtain an annotation for a given word W_{it} by concatenating the forward hidden state $\overrightarrow{h}_{it}$ and backward hidden state $\overleftarrow{h}_{it}$, i.e., $h_{it} = [\overrightarrow{h}_{it}, \overleftarrow{h}_{it}]$, which summarizes the information of the whole sentence centered around W_{it}.

Not all words contribute equally to the representation of the sentence meaning. Hence, The model introduces attention mechanism to extract such words that are important to the meaning of the sentence and aggregate the representation of those informative words to form a sentence vector:

$$u_{it} = tanh(W_w h_{it} + b_w), \tag{3}$$

$$\alpha_{it} = \frac{exp(u_{it}^\top u_w)}{\sum exp(u_{it}^\top u_w)}, \tag{4}$$

$$s_i = \sum \alpha_{it} h_{it}, \tag{5}$$

That is, the model first feed the word annotation h_{it} through a one-layer MLP to get u_{it} as a hidden representation of h_{it}, then the model measure the importance of the word as the similarity of u_{it} with a word level context vector u_w(a weight matrix) and get a normalized importance weight α_{it} through a softmax function. After that, the model computes the sentence(in here is the document) vector s_i as a weighted sum of the word annotations based on the weights. Then we could feed the s_i into a softmax function and get the final classification result.

Batch normalization
In a normal neural network, we need to choose the initial learning rate carefully. Also, we need to consider whether we need to use a dropout or other regularization methods when batch normalization is a wonderful method. It makes training multi-layered networks easier, allows higher learning rates and weight initialization less crucial. It could act like a regulariser, which is widely used in many tasks now. The main steps of it are :

Algorithm 1 Batch normalization

```
    Input: Values if x over a minibatch : $\delta \{x_1...m\}$;
    Input: learning parameters : $\gamma\beta$
    Output:$\{y_i = BN_{\gamma,\beta}(x_i)\}$
$\mu_\delta \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$                          $mini-batch-mean$
${\sigma_\delta}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\delta)^2$        $mini-batch-variance$
$\widehat{x_i} \leftarrow \frac{x_i - \mu_\delta}{\sqrt{{\sigma_\delta}^2 + \epsilon}}$                        $normalize$
    $y_i \leftarrow \gamma\widehat{x_i} + \beta \equiv BN_{\gamma,\beta}(x_i)$        $scale-and-shift$
```

It calculates the mean and variance of the mini-batch, and normalize the data with them. In the algorithm, ϵ is a constant added to the mini-batch variance for numerical stability. The BN transform can be added to a network to manipulate any activation. In the notation y $= BN_{\gamma,\beta}(x)$, we indicate that the parameters γ and β are to be learned, but it should be noted that the BN transform does not independently process the activation in each training example. Rather, $BN_{\gamma,\beta}(x)$ depends both on the training example and the other examples in the mini-batch. The scaled and shifted values y are passed to other network layers.

3. Experiments

The structure of the baseline model we covered in the previous coursework can be found in Fig.2. While the model is simple, we find word vector for every input word form the dictionary we pre-trained, and then feed the words into LSTM cell by order, and use the last output of LSTM layer with a softmax function to get the final result. The loss is calculated by softmax cross-entropy and uses Adam learning rule to update learning rate through time. As we talked before, we only used half of the data in the data set for the previous baseline. As we enlarged our data set, we retrain our baseline model with the new data set and the result is showed in Table.2(with the label no-dropout). It is obvious that the model overfits the training data and the gap between training and validation set is huge. Though we did add dropout in our baseline, it seems does not work well. So, the first part of our experiment explores the use of dropout. This is the Hyperparameters of all the experiments(if changes we will explain in the experiments):

HYPERPARAMETERS	VALUE
WORD EMBEDDING DIMENSION	100
LEARNING RULE	ADAM
LEARNING RATE	0.0001
FIXED LENGTH OF REVIEW	500
BATCH SIZE	100
EPOCH	50

Table 1. Hyperparameters of experiments

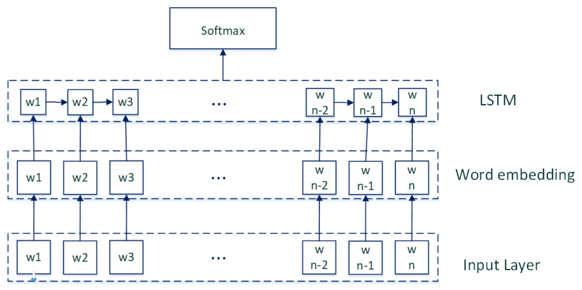

Figure 2. The structure of baseline experiment

3.1. Dropout

Dropout is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks. While for dropout in LSTMs, the dropout operator corrupts the information carried by the units, forcing them to perform their intermediate computations more robustly. At the same time, we do not want to erase all the information from the units. It is especially important that the units remember events that occurred many timesteps in the past(Zaremba et al., 2014), so the dropout should add out of the LSTM cell, it could be added after the input before continuing feed into LSTM layer, and it could also be added after each LSTM layers(for multilayers). For this task, we implement dropout after each LSTM layer, and the results of the experiments are shown in Fig. 3 & 4 and Tab. 2:

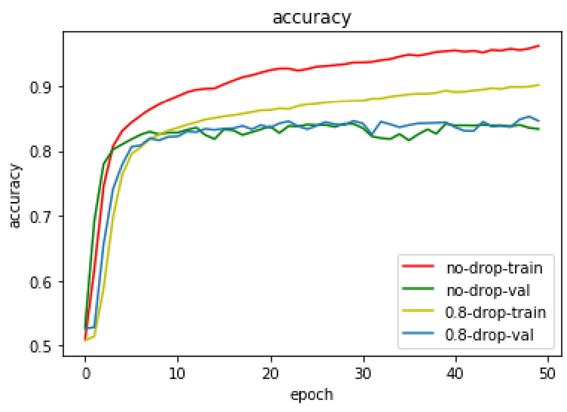

Figure 3. Comparison about baseline and dropout

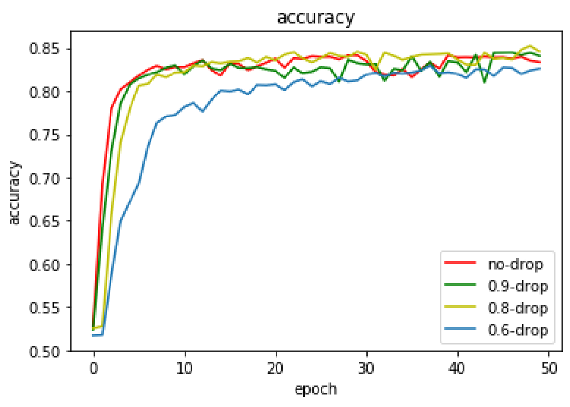

Figure 4. Validation accuracy of different dropout ratios

DROPOUTRATIO	TRAINLOSS	VALELOSS	TRAINACC	VALACC
NO-DROPOUT	0.0302	0.2876	98.9%	82.7%
0.9-DROPOUT	0.0602	0.2386	90.9%	83.1%
0.8-DROPOUT	0.0701	**0.2071**	89.5%	**83.7%**
0.6-DROPOUT	0.1042	0.2098	88.7%	81.6%

Table 2. Final results of different dropout ratio

As we can see from the figure 3, after applying dropout to the model, the gap between training and validation accuracy become smaller, the problem of overfitting improved a lot. Also after the experiment with different dropout ratio, we find that 0.8 is the most suitable one for this model, could see from figure 4 table 2, the accuracy of validation set raised 1% and also the validation loss reduced 0.08, so we set dropout ratio to be 0.8 for the next experiments.

3.2. Bidirectional LSTM

When dealing with sequence classification tasks, we have access to both past and future input features for a given time. We can thus utilize a bidirectional LSTM network. In doing so, we can efficiently make use of past features (via forwarding states) and future features (via backward states) for a specific time frame. The forward and backward passes over the unfolded network over time are carried out in a similar way to regular network forward and backward passes, except that we need to unfold the hidden states for all time steps. At previous experiment, we only use the final output of forwarding sequence for classification. For now, we combine the outputs of forwarding and backward sequences and concatenate them as the input for the softmax layer to do the final recognition. The results are shown below:

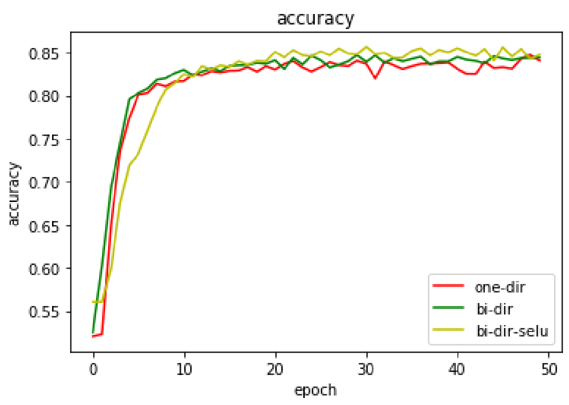

Figure 5. Validation accuracy of different LSTM models

Model	TrainLoss	ValeLoss	TrainAcc	ValAcc
ONE-DIR	0.0701	0.2071	89.5%	83.7%
BI-DIR	0.0652	0.2036	89.7%	84.0%
BI-DIR-SELU	0.0611	**0.1871**	89.9%	**84.5%**

Table 3. Final results of different LSTM-directions

After we implement the model with bidirectional LSTM, it seems that the improvement is very limited (see from figure 5 table 3), the validation accuracy only raised about 0.03%, so we add a dense layer with Selu activation function after the bidirectional LSTM layer. After applying the dense layer, the results improved a little and the accuracy of validation goes up about 0.8%. The validation loss reduce about 0.02. With more layers, the results seem to become better, so we decide to add more LSTM layers to the model and figure out if it works.

3.3. Multilayer LSTM

Neural network with more layers usually have better results, but more hidden layers will probably cause vanishing gradient problem(Hochreiter, 1998), LSTM nets are in principle capable to store past inputs to produce the currently desired output. This recurrent net property is used in time series prediction and process control. Practical applications involve temporal dependencies spanning many time steps between relevant inputs and desired outputs. In this case, however, gradient descent learning methods take too much time, the learning time problem appears because the error vanishes as it gets propagated back. So we tried the different number of LSTM layers, and try to find the best one, the experiment results are here:

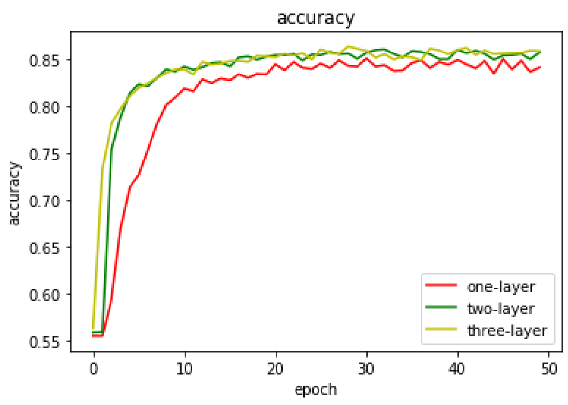

Figure 6. Validation accuracy of different LSTM layers

Model	TrainLoss	ValeLoss	TrainAcc	ValAcc
ONE-LAYER	0.0611	0.1871	89.9%	84.5%
TWO-LAYER	0.0652	0.1686	89.8%	85.7%
THREE-LAYER	0.0711	**0.1671**	89.9%	**85.9%**

Table 4. Final results of different LSTM-layers

From table 4 and Figure 6 we can know, the model with three LSTM layers have the best performance, the validation accuracy increase 1.4% compared with the single layer one, and also the validation error decreased 0.02. But the difference between two layers and three layers is very small, the three layers one is slightly better, so we pick this one for next experiments. Consider the small difference, we believe that if we set the model with four LSTM layers, the results may not better than this one. Because of the limitation of time, we test it in a smaller dataset, with 15000 reviews as the training set, and 2000 reviews, and also train a three-layer model with other same parameters to do the comparison.The result shows that when using four LSTM layers, the validation accuracy reduced 0.5% compared with the three layer's result, that is because of the vanishing gradient problem, there are some ways to fix it, we will try in future works.

3.4. Attention

If words were inherently important or not important, models without attention mechanism might work well since the model could automatically assign low weights to irrelevant words and vice versa. However, the importance of words is highly context dependent. The classification task for a movie review sometimes mostly depends on some particular words, especially adjectives. In the past experiments, we only use the output of the last time step and feed it into softmax to get classification results. LSTM could keep the information from previous steps, but that may not enough, so we use the word level attention(as we discuss in the methodology). We calculate an alpha score for each step's output, and multiply them with each step's output, and continued with a mean pooling to get a final input for next layer, the results are below:

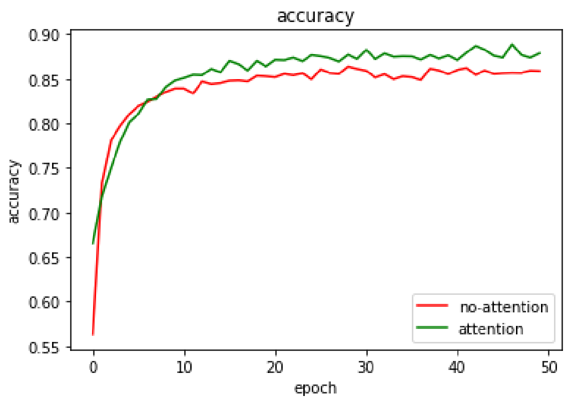

Figure 7. Validation accuracy of attention

MODEL	TRAINLOSS	VALELOSS	TRAINACC	VALACC
NO-ATTENTION	0.0711	0.1671	89.9%	85.9%
ATTENTION	0.0511	**0.1271**	94.3%	**88.5%**

Table 5. Final results of attention

The performance of attention model is remarkable(Table 5 and Figure 7). the validation accuracy raised about 2.6% and the loss reduced 0.06. Compared with all the other methods we used, attention mechanism is the most powerful one. When individuals dealing with pictures or conversions, they usually focused on some main parts of the items. This is the same idea of attention model, so this is probably why it is so significant.

3.5. Batch normalization

The last part of our experiments is batch normalization, it could be used in different places in the neural network. We add batch normalization after attention layer and before feed it into next selu layer. We initialize γ to be all one, and β to be all zero, and set to be 0.01 (see methodology for the parameters). The results are shown in Tab.6.

MODEL	TRAINLOSS	VALELOSS	TRAINACC	VALACC
NO-BATCH	0.0511	0.1271	94.3%	88.5%
BATCH-NORMAL	0.0611	0.1320	93.3%	88.6%

Table 6. Final results of batch normalization

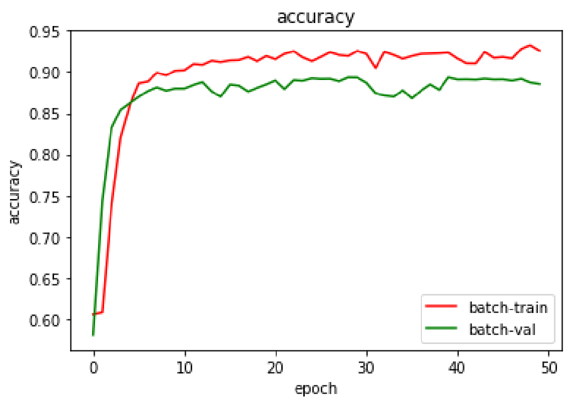

Figure 8. Accuracy of batch normalization

We can know from the table 6. Batch normalization did not improve the validation accuracy very much, the results only changed about 0.1%, but as we can see from the figure 8, the difference between training and validation is much smaller, that means batch normalization could regularize the data and make it less overfit the training data. Another reason why it did not enhance the validation accuracy strongly is that, we only apply batch normalization before a selu activation layer, so the influence is not very big. For Recurrent neural networks, batch normalization could use inside the recurrent cells, see(Tim Cooijmans et al., 2016), which would be more efficient and more useful in improving the accuracy.

3.6. Test results of final model

After these experiments and comparisons, we run the model on the test set with the parameters we selected. The hyperparameters are same as baseline. We use multi-LSTM layers(three layers) instead of one single LSTM layer and apply dropout after each LSTM layer, use attention model for all the outputs of the last LSTM layer and use batch normalization for attention outputs. Then, pass into a selu activation layer and feedforward to softmax. The whole structure is explained in Fig.9. The final results of the test set are reported in Tab.7. And the error of test set reduced about 0.2, and accuracy raised 6.2%.

MODEL	LOSS	ACC
BASELINE TEST RESULTS	0.3035	81.9%
FINAL MODEL TEST RESULTS	0.1012	88.1%

Table 7. Results of test set

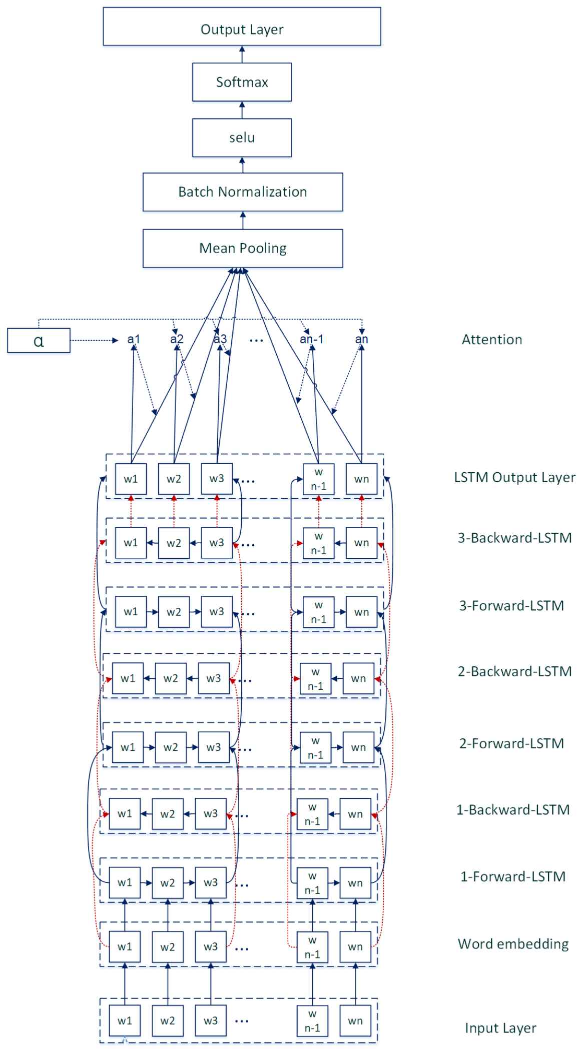

Figure 9. The structure of final model

4. Related work

Semantic Vector Spaces

Semantic Vector Space (SVS) has become the standard approach for automatic modelling of lexical semantics in NLP. The word meaning is captured according to the word frequency distribution of the simultaneous context words in the large corpus, which applies to traditional linguistic fields. (Sagi et al., 2009; Cook & Stevenson, 2010)Each word has different meanings in different sentences, thus the core semantic concepts are sentences, not words. There is some paper attempt to build a combination model using distributed semantic representation as input such as noun-verb combinations or adjective-noun combinations. (Baroni & Zamparelli, 2010) Ultimately, the meaning of the longer phrases are captured until the entire sentence. The matrix-vector recursive neural network (MV-RNN) model (Socher et al., 2012) classifies the emotion tags of movie reviews and classifies semantic relationships, such as cause-effect or topic-message between sentences. One of the tasks in their model predicts the distribution of sentiments in adverb-to-adjective movie reviews, such as unbelievably sad or really awesome.

Deep Learning

Deep learning models have successfully applied in semantic analysis, machine translation and text summarization. In the past few years, applying RNN on deep learning has achieved certain success in speech recognition, language modelling, translation, picture description and other issues. LSTM is a special type of RNN that can learn long-term dependency information and it was proposed by (Hochreiter & Schmidhuber, 1997) and was recently improved and promoted by Alex Graves.(Graves & Schmidhuber, 2005) He found that bidirectional LSTM networks outperform one directional one, and it run much faster than standard RNN and time-windowed MLP. Each training sequence is forward and backward respectively by two LSTMs, and both are connected to one output layer. This structure provides complete past and future context information for each point in the output layer input sequence. There is a problem with this model. When the context is larger, the final state vector will lose more information. Attention is a weight vector (usually the output of softmax) whose dimension is equal to the length of the context. The larger the weight, the more important it is to represent the corresponding context. In addition, the purpose of attention is not limited to text generation or language modelling, it can also apply on QA (question answering) system. (dos Santos et al., 2016; Tan et al., 2015)

Sentiment Analysis

Text sentiment analysis research began in the 1990s. Hatzivassiloglou and McKeown (Hatzivassiloglou & McKeown, 1997) discovered that conjunctions restrict the semantic orientation of adjectives in corpus data, attempting to make emotional tendentiousness judgments on words. Turney (Turney, 2002) and Pang (Pang et al., 2002) made early contributions in the field of textual sentiment analysis, they tried different methods to analysis positive and negative reviews on products and movies. Sentiment Analysis has two aspects, Document-level Sentiment Classification and Aspect-level Sentiment Analysis. The document-level tries to classify the entire document into 'positive' or 'negative' class. Unlike our datasets that have divided data into these two groups, the vast majority of data on the web in the real world is unstructured text. SentiWordNet approach categories the words in WordNet into a positive and negative group, then assigning the weight of score to each word. (Baccianella et al., 2010)The aspect-level sentiment analysis aggregates sentiment on entities mentioned within documents or aspects of them. Singh, Piryani, and Uddin(Singh et al., 2013) identify a list of aspects of movie domain since a particular aspect is expressed by different words. Then they created an aspect-vector for all aspects, then use SentiWordNet to compute its sentiment polarity. Also, introducing attention can capture information that is important for resolving the sentiment of different aspects. Attention mechanism overcomes the problems of traditional encoder-decoder structures when the input sequence is very long, it is difficult for the model to learn a reasonable vector representation; and how to encode input sequences into a fixed vector representation. This model can be applied in different fields, such as machine translation, picture description, and speech recognition.(Bahdanau et al., 2014; Xu et al., 2015; Chorowski et al., 2015)

In order to select a model to characterize the relationship between a word and its context, we need to capture the context information of a word in the word vector. Word embedding is a general term for language modelling and feature learning techniques in NLP that map words or phrases in a vocabulary to a vector of real numbers. For example, Apple corresponding vector is [0.1 0.6 -0.5 ...]. The reason why we want to turn each word into a vector is to facilitate calculations, such as "find the synonym of the word Apple", and we can do this by seeking the most similar vector of the word Apple in the cos distance. There are two ways to generate the vector, count-based method and predictive method.(Baroni et al., 2014) Word2vec is a typical predictive model created by Google for generating word vectors, relying on skip-grams or continuous word bags (CBOW) to build neural word embedding. The vector can be subsequently used for many natural language processing applications, such as analyzing Weibo message, Twitter message and comments on clothing products. (Xue et al., 2014; dos Santos & Gatti, 2014; Zhang et al., 2015) The goal of CBOW is to predict the probability of the current word based on the context. However, Skip-gram predicts the probability of the context based on the current word. Doc2vec is another tool that Mikolov propose to calculate longer content vectors based on word2vec.(Le & Mikolov, 2014) Apart from adding a paragraph vector, this method is almost equivalent to Word2Vec. There are also two methods of Doc2vec model: Distributed Memory (DM) and Distributed Bag of Words (DBOW). DM attempts to predict the probability of a word given context and paragraph vectors. During the training of a sentence or document, the paragraph ID remains unchanged, sharing the same paragraph vector. DBOW predicts the probability of a set of random words in a paragraph given only a paragraph vector.

Other work on the same data
Le and Mikolov (Le & Mikolov, 2014)propose two extended methods which are represented paragraph as vectors. Bag-of-words (BOW), which split the paragraph into words, uses words as the dimension of a vector space, and the frequency of occurrence of each word in the paragraph as the value of the corresponding dimension of the text vector. The disadvantage of BOW is that it ignores the order in which words appear in the text, and does not consider the semantic information of words. Another method, bag-of-n-grams, considers the word order but increases the dimensions and exacerbates data sparseness. In the case of considering semantics, assuming the word vectors contain the semantic information of words, generate a new vector that averages the weight of words in a document. The goal of PV-DM maximizes output probability of target word, using a stochastic gradient descent method to achieve convergence. The difference is that a segment vector is added to the input layer, and the newly added segment vector can simply be seen as adding a new word as input. While predicting a word, use the paragraph vector where the word located as a new input. PV-DBOW takes a paragraph vector as input, and then randomly samples the word sequence (context) from the vector's corresponding paragraph as output. Similar to skip-grams, this method reduces the amount of input layer parameters. They found that the combination of PV-DM and PV-DBOW can produce the best results.

Future work
In our experiment, we can realize that there is little improvement between one direction LSTM and bidirectional LSTM. In the future, there is two method we can try. Firstly, we can add a CRF layer after bidirectional LSTM layer. Unlike LSTM, CRF can consider long-term contextual information. It considers more about the linear weighted combination of local features of the entire sentence. Moreover, we can add a one-dimensional convolutional neural network after the CRF layer. Since the content in our dataset is English, English words are made up of letters that hide several features that can extract by CNN convolution operations. Also, for now, our classification work is based on the word level, for next studies, we will try to develop a model to do the task on document level. When we build a model based on document level, we can try to use doc2vec instead of word2vec. The learned vector can be used to find the similarity between the sentence/paragraphs/documents by calculating the distance, or it can further tag the document.

5. Conclusion

This study presents a regional LSTM model to predict the sentiment tendency of movie reviews. The key idea of this task is to classify the documents as negative or positive automatically. By applying dropout, bidirectional LSTM, more hidden layers, attention model and batch normalization, the proposed model outperformed the previous model and the test accuracy raised 6.2%. But there is still a lot of other methods could be used to improve, as we mentioned in the related work, CNN-LSTM model is a great way to raise the accuracy, and other recurrent networks like GRU are all worth trying. The future work will be firstly focused on exploring the problem appeared in our model, as we discussed in the last section and also try other methods to improve the performance of our model.

References

Baccianella, Stefano, Esuli, Andrea, and Sebastiani, Fabrizio. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In Chair), Nicoletta Calzolari (Conference, Choukri, Khalid, Maegaard, Bente, Mariani, Joseph, Odijk, Jan, Piperidis, Stelios, Rosner, Mike, and Tapias, Daniel (eds.), *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA). ISBN 2-9517408-6-7.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua.

Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Baroni, Marco and Zamparelli, Roberto. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 1183–1193. Association for Computational Linguistics, 2010.

Baroni, Marco, Dinu, Georgiana, and Kruszewski, Germán. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pp. 238–247, 2014.

Chorowski, Jan K, Bahdanau, Dzmitry, Serdyuk, Dmitriy, Cho, Kyunghyun, and Bengio, Yoshua. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pp. 577–585, 2015.

Cook, Paul and Stevenson, Suzanne. Automatically identifying changes in the semantic orientation of words. In *LREC*, 2010.

Denil, Misha, Bazzani, Loris, Larochelle, Hugo, and de Freitas, Nando. Learning where to attend with deep architectures for image tracking. 2011.

dos Santos, Cicero and Gatti, Maira. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 69–78, 2014.

dos Santos, Cıcero Nogueira, Tan, Ming, Xiang, Bing, and Zhou, Bowen. Attentive pooling networks. *CoRR, abs/1602.03609*, 2(3):4, 2016.

Graves, Alex and Schmidhuber, Jürgen. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6): 602–610, 2005.

Hatzivassiloglou, Vasileios and McKeown, Kathleen R. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics*, pp. 174–181. Association for Computational Linguistics, 1997.

Hochreiter, Sepp. Recurrent neural net learning and vanishing gradient. *International Journal of Uncertainty*, 6(2): 107–116, 1998.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

Karol Gregor, Ivo Danihelka, Alex Graves Danilo Jimenez Rezende Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv:1502.04623*, 2015.

Le, Quoc and Mikolov, Tomas. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pp. 1188–1196, 2014.

Luong, Minh-Thang and Hieu Pham, andChristopher D. Manning. Effective approaches to attention-based neural machine translation. *arXiv:1508.04025*, 2015.

Maas, Andrew L., Daly, Raymond E., Pham, Peter T., Huang, Dan, Ng, Andrew Y., and Potts, Christopher. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

Pang, Bo, Lee, Lillian, and Vaithyanathan, Shivakumar. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 79–86. Association for Computational Linguistics, 2002.

Sagi, Eyal, Kaufmann, Stefan, and Clark, Brady. Semantic density analysis: Comparing word meaning across time and phonetic space. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pp. 104–111. Association for Computational Linguistics, 2009.

Singh, Vivek Kumar, Piryani, Rajesh, Uddin, Ashraf, and Waila, Pranav. Sentiment analysis of movie reviews: A new feature-based heuristic for aspect-level sentiment classification. In *Automation, computing, communication, control and compressed sensing (iMac4s), 2013 international multi-conference on*, pp. 712–717. IEEE, 2013.

Socher, Richard, Huval, Brody, Manning, Christopher D, and Ng, Andrew Y. 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*, pp. 1201–1211. Association for Computational Linguistics, 2012.

Tan, Ming, Santos, Cicero dos, Xiang, Bing, and Zhou, Bowen. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*, 2015.

Tim Cooijmans, Nicolas Ballas, Gulcehre, Cesar Laurentand CaGlar, and Courville, Aaron. Recurrent batch normalization. 2016.

Turney, Peter D. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 417–424. Association for Computational Linguistics, 2002.

Vu Pham, and Theodore Bluche and Kermorvant, Christopher. Idropout improves recurrent neural networks for handwriting recognition. *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, 2014.

Xu, Kelvin, Ba, Jimmy, Kiros, Ryan, Cho, Kyunghyun, Courville, Aaron, Salakhudinov, Ruslan, Zemel, Rich, and Bengio, Yoshua. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pp. 2048–2057, 2015.

Xue, Bai, Fu, Chen, and Shaobin, Zhan. A study on sentiment computing and classification of sina weibo with word2vec. In *Big Data (BigData Congress), 2014 IEEE International Congress on*, pp. 358–363. IEEE, 2014.

Yang, Zichao, Yang, Diyi, Dyer, Chris, He, Xiaodong, Smola, Alex, and Hovy, Eduard. Hierarchical attention networks for document classification. 2016.

Zaremba, Wojciech, Sutskever, Ilya, and Vinyals, Oriol. Recurrent neural network regularization. *arXiv:1409.2329*, 2014.

Zhang, Dongwen, Xu, Hua, Su, Zengcai, and Xu, Yunfeng. Chinese comments sentiment classification based on word2vec and svmperf. *Expert Systems with Applications*, 42(4):1857–1863, 2015.