Different Deep Learning Models Applied to Sentiment Analysis

Junyan Qiu, Zihao Li, Mining Feng, Yue Tao, Yiqiang Zhong
University of Chinese Academy of Sciences, UCAS
Huairou district, Beijing, China
qiujunyan2018@ia.ac.cn

Abstract

Sentiment analysis or opinion mining, one of the classical research region in Nature Language Processing (NLP), is the computation, appraisals, and attitudes towards entities such as products, services, organizations, individuals, issues, events, topics, and their attitudes. In this paper, we propse three models for emotion classification, respectively, which are CNN, BiGRU+Attention and Transformer. And using pretrained word vector, L2 regularization and early stopping strategy to optimize. Experiments conducted on IMDb datasets demonstrate that the Transformer outperforms others on accuracy.

1. Introduction

Sentiment analysis is a language processing task that uses a computational method to identify content that has opinions. Unstructured text data on the web often carries an expression of user opinions. Sentiment analysis attempts to identify the author's views and expressions of emotions. A simple sentiment analysis algorithm attempts to divide a document into positive or negative based on the opinions expressed in the document. The document-level sentiment analysis problem is essentially as follows: Given a set of documents D, the sentiment analysis algorithm divides each document into one of two categories, positive and negative.

This paper mainly studies movies comment. For film review from IMDb, analyze its positive and negative emotions, understand the attitude of the audience. Sentiment analysis of movie review can help the audience of the movie, make decisions and save time.

As required, sentiment analysis can be realized as a classification problem (negative, positive, neutral) or regression problem (1-10, the value means the intensity of feeling). Existing research has produced numerous techniques for the multifarious task of sentiment analysis, which include both supervised and unsupervised methods and early papers used all types of supervised machine learning methods (such as Support Vector Machines (SVM), Maximum Entropy, Naive Bayes, etc.) and feature combination, as for unsupervised methods which includes exploiting sentiment lexicon, grammatical analysis, and syntactic patterns.

Nowadays, deep learning has shown its powerful representation ability and produced state-of-the-art results in many application domains, ranging from computer vision and speech recognition on NLP. Furthermore, lots of industrial circles are embracing deep learning actively. RNN neural network as the representative is wildly applied in the sentiment analysis, but it couldn't solve the problem of dependence among each part, especially when the sentences are long enough, the model performance will deteriorate dramatically. Therefore, LSTM network, a special type of RNN, was introduced to sequence modeling tasks. Through the design of called gate (input gate, output gate, forget gate), LSTM is capable of learning long-term dependencies. Nevertheless, the structure of LSTM is too complicated and GRU unit was invented, which merge the input gate and forget gate of LSTM as an update gate to slim the network. Comparing with LSTM, GRU can reach the same accuracy with fewer parameters. However, when the numbers of words over 30, the property of LSTM will also decrease sharply and Attention mechanism was added to Encoder-Decoder frame. Transformer, an encoding pattern essentially, is a new attention mechanism. It combines each part of input and output by a calculation so that eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies at the same time.

2. Problem Definiton

Since early 2000, sentiment analysis has been the most active area of research in data mining and natural language processing (NLP).One's character is always influenced by the suggestions and comments of

others[2].That's why our comments are so influenced by other comments, and whenever we need to make a decision, we often look for other people's comments. When we need to check the comments, we will begin to try to find the comments in many digital platform, such as social media, review sites, BBS, blogs and twitter. Although each web site contains a lot of comment text, normal readers will not be able to identify relevant websites, extract and purify from comments, so we can't make the right decision. This applies not only to any individual business person, but also to organizations, companies and political parties. This is why people need an automated intelligent emotion analysis system, which can accurately give the correct emotion and relevant information in a shorter time and thus obtain benefits.

The definition of emotion is an attitude, emotion or mood etc., and the definition of sentiment analysis is to identify the polarity of the comment text or the subjectivity and emotion of specific topics in documents or sentences[11].

It is the task of emotional analysis to mine people's comments and feelings on any topic of interest[9].

Now, sentiment analysis can be applied to almost every possible field, such as products, social activities and political election services, market research, social media, advertising, recommendation systems, email filtering, stock market predictions, upcoming movie reviews, sentiment predictions, book reviews, etc.

In this paper, we study how to use some methods of deep learning to conduct emotional analysis of movie reviews.

Pre-training has greatly impacted on CV tasks such as classification, object detection, segmentation, but existing in NLP still require task-specific modifications and training from scratch. And this paper s a universal solution universal in the sense that we'd like a single architecture and training method, minimal hyper-parameter tuning, minimal preprocessing requirements, and good results without boatloads of task-and domain-specific data.

Recently, Elmo, GPT, BERT, GPT 2.0 are refreshing the records of all kinds of NLP tasks over and over again by using pre-train strategy. And those methods have a common and distinct peculiarity is integrating numerous external information, so the parameters will get about hundreds of millions, which will cause overfitting problem when face small dataset and ULMFiT could solve it properly.

3. Proposed Method

3.1. CNN

Our traditional such as naive bayes classifier and SVM, most of the text representation method is to transform it into "word bag model", mainly based on word frequency to do appear in the text, this also leads to the sequence information between the word and the word lost, we after word segmentation, the sentence itself is equivalent to cut into piece, after the word and the word combinations tend to have a local semantics. An important issue here is the contradiction between granularity and semantics. If the granularity is too large, then it's too sparse, just like forcing n-gram, it doesn't make a lot of sense, if the granularity is too small then it doesn't make a lot of sense [6]. In the case of CNN, after the convolutional layer, the semantics of each k word combination are put together to obtain a relatively accurate sentence vector.

3.1.1 Model of CNN

Input layer: As can be seen from the figure 1, the input layer is the word vectors corresponding to the words in the sentence arranged from top to bottom. For example, if there are n words and the dimension of the word vector is K, the matrix is the matrix of n* K. This matrix can be both static and dynamic, I checked some blogs, one of them explained that static is the size of the word vector is fixed, non-static means after the back propagation, the error caused by the word vector is fine tuned, for the words not logged in, please add some padding here.

The first layer of the convolution layer: After the input layer passes the convolution layer of the convolution kernel of h*k, the Feature Map with column number 1 is obtained, where h represents the number of longitudinal words and k represents the dimension of the word vector. After convolution, feature is obtained by activating the function f.Remember to ci.It is the value convolved by the h words adjacent to $\boldsymbol{x}_i : \boldsymbol{I} + \boldsymbol{h}_1$. The value after activation is also the output of the current layer. The value after the convolution:$\boldsymbol{w}\boldsymbol{x}_i : \boldsymbol{i} + \boldsymbol{h} - 1 + \boldsymbol{b}$;The output value of sentence embedding: $\boldsymbol{c}_i = \boldsymbol{f}(\boldsymbol{w}\boldsymbol{x}_i : \boldsymbol{i}+\boldsymbol{h}_1+\boldsymbol{b})$;windows size: $\boldsymbol{h}$. After that, a sentence with the length of n will has word windows like this $[\boldsymbol{x}_1 : \boldsymbol{h}, \boldsymbol{x}_2 : \boldsymbol{h}+1, \boldsymbol{x}_3 : \boldsymbol{h}+2, ..., \boldsymbol{x}_{n-h+1} : \boldsymbol{n}]$, and will become $\boldsymbol{c} = [\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_{nh+1}]$, dimensionality=$(1, \boldsymbol{n} - \boldsymbol{h} + 1)$ after convolution. Then, Max pooling is conducted to select the most important feature. The pooling scheme can be selected according to the length of the sentence.

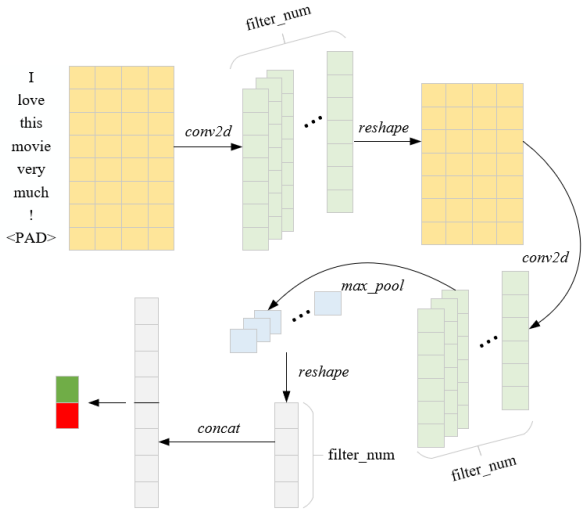

Figure 1. Model of CNN for classification

Pooling Layer: Here, the Pooling layer is said to be the max-over-time Pooling method. In fact, this method is to extract the maximum value from the previous Feature Map. When using the maximum Pooling method, we generally believe that the maximum value is extracted from the Pooling layer, which is generally the most representative or important. And it's going to be a one-dimensional vector.

Full connection layer + softmax layer: The one-dimensional vector after pooling is connected to a softmax layer for classification by means of full connection, and Dropout is used in the part of full connection to reduce overfitting.

3.2. BiGRU+Attention

In this project, we focus on sentence-level classification. The overall architecture of the Attention Network is shown in Fig 2. It consists of tol parts: a word sequence encoder and a word-level attention layer.

3.2.1 Word Encoder

Given a sentence with words, we first embed the words to vectors through an embedding matrix. We use a bidirectional GRU (Bahdanau et al., 2014)[1] to get annotations of words by summarizing information from both directions for words, and therefore incorporate the contextual information in the annotation. The bidirectional GRU contains a forward GRU and a backward GRU.

$$\boldsymbol{x}_t = \boldsymbol{W_e w_t}, t \in [1, T] \tag{1}$$

$$\overrightarrow{\boldsymbol{h}}_{\boldsymbol{t}} = \overrightarrow{\boldsymbol{GRU}}(\boldsymbol{x_t}), t \in [1, T] \tag{2}$$

$$\overleftarrow{\boldsymbol{h}}_{\boldsymbol{t}} = \overleftarrow{\boldsymbol{GRU}}(\boldsymbol{x_t}), t \in [T, 1] \tag{3}$$

3.2.2 Word Attention

Not all words contribute equally to the representation of the sentence meaning. Hence, we introduce multiplicative attention 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. Specifically,

$$\boldsymbol{u_t} = tanh(\boldsymbol{W_w h_t} + \boldsymbol{b_w}) \tag{4}$$

$$\boldsymbol{\alpha_t} = \frac{exp(\boldsymbol{u_t^\top u_w})}{\sum_t exp(\boldsymbol{u_t^\top u_w})} \tag{5}$$

$$\boldsymbol{v} = \sum_t \boldsymbol{\alpha_t h_t} \tag{6}$$

3.2.3 Sentence Classification

The word vector is a high level representation of the sentence and can be used as features for sentence classification:

$$\boldsymbol{p} = softmax(\boldsymbol{W_c v} + \boldsymbol{b_c}) \tag{7}$$

And, we use the cross entropy of the correct labels as training loss.

3.3. Transfomer

To solve the long-term independence and reduce the computation in sequence transduction, Google proposes a new network architecture, the Transformer[13], using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, which allows for significantly more parallelization and can reach a new state of the art in translation quality. In

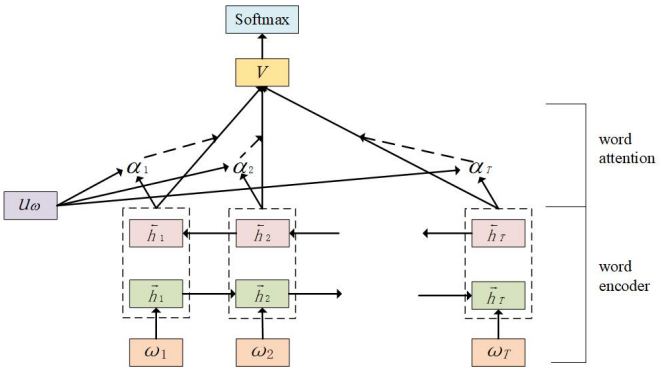

Figure 2. Model of BiGRU+Attention

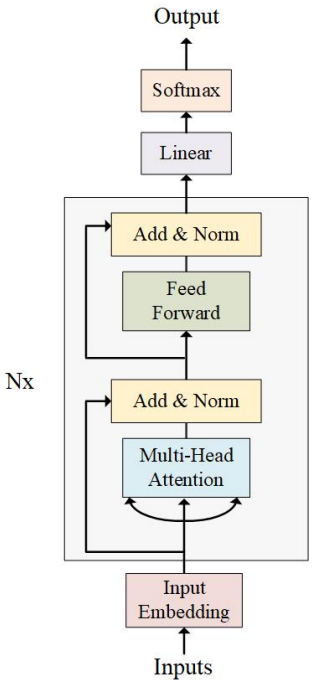

Figure 3. Model of Transformer

this project, we apply and modify the model to complete sentiment analysis. It follows the overall architecture as shown in Figure 3.

3.3.1 Encoder

We set the encoder to a stack of 3 identical layers (Figure 4). Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection around each of the two sub-layers, followed by layer normalization.

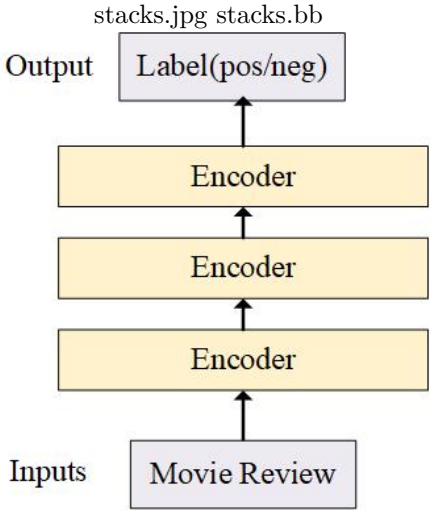

Figure 4. Encoder stacks

3.3.2 Attention Mechanism

Scaled Dot-Product Attention The input consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values(Figure 3). Self-Attention can get rid of the restriction of length, ignore the distance between words, directly calculate dependencies, and learn the internal syntactic and semantic structure of a sentence.

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V} \qquad (8)$$

Multi-Head Attention We calculate Scaled Dot-Product Attention running in parallel in 5 different attention heads. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 5. It allows the model to jointly attend to in-

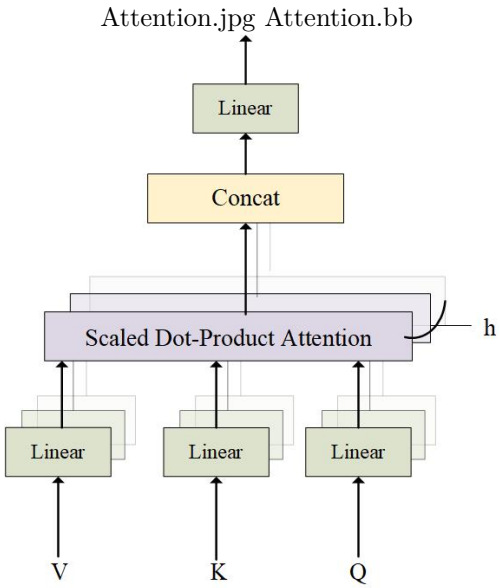

Figure 5. Multi-Head Attention consists of several attention layers running in parallel

formation from different representation subspaces at different positions.

3.3.3 Residual Dropout

We apply dropout[12] to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply dropout to the sums of the embeddings in the encoder decoder stacks. For the base model, we use a rate of $P_{drop} = 0.1$.

3.3.4 Position-wise Feed-Forward Networks

In addition to attention sub-layers, each of the layers in our encoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

$$FFN(\boldsymbol{x}) = max(0, \boldsymbol{x}\boldsymbol{W_1} + \boldsymbol{b_1})\boldsymbol{W_2} + \boldsymbol{b_2} \qquad (9)$$

3.3.5 The Final Linear and Softmax Layer

We use the usual learned linear transformation and softmax function to convert the output to predicted sentiment label.

4. Training

4.1. Word Vector

Word vector model can learn syntactic and semantic information from large scale unlabeled corpus.

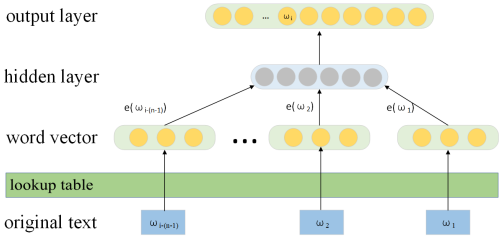

Figure 6. Model of NNLM

Word vector-based neural network model also improves the performance of many natural language processing tasks. It is an important part of neural network technology in deep learning of natural language processing technology.

4.1.1 Feedforward Neural Net Language Model (NNLM)

The probabilistic feedforward neural network language model has been proposed in [3]. It consists of input, projection, hidden and output layers, as shown in Figure 6. At the input layer, N previous words are encoded using 1-of-V coding, where V is size of the vocabulary. The input layer is then projected to a projection layer P that has dimensionality N × D, using a shared projection matrix. As only N inputs are active at any given time, composition of the projection layer is a relatively cheap operation. The NNLM architecture becomes complex for computation between the projection and the hidden layer, as values in the projection layer are dense.

4.1.2 Continuous Bag-of-Words Model(CBOW)

The CBOW architecture[10] is similar to the feedforward NNLM, where the non-linear hidden layer is removed and the projection layer is shared for all words (not just the projection matrix); thus, all words get projected into the same position (their vectors are averaged). Furthermore, it uses future and history words at the input, where the training criterion is to correctly classify the current (middle) word. It can be seen in Figure 7.

4.1.3 Continuous Skip-gram Model

Instead of predicting the current word based on the context, Skip-gram Model[10] tries to maximize classification of a word based on another word in the same sentence. More precisely, as shown in Figure 8, it uses each current word as an input to a log-linear

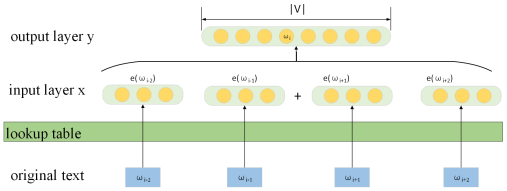

Figure 7. Model of CBOW

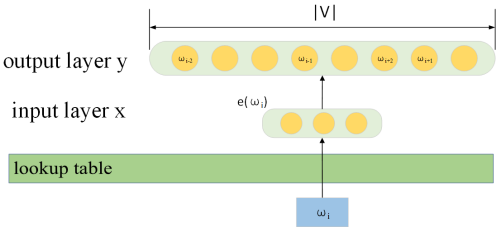

Figure 8. Model of Skip-gram

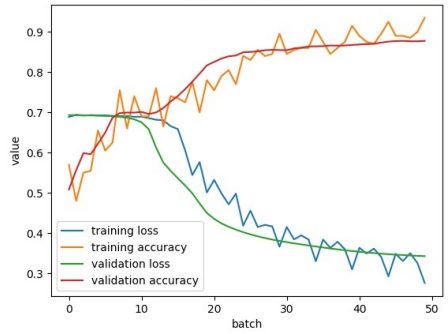

Figure 9. Attention without pretrained word vector

classifier with continuous projection layer, and predict words within a certain range before and after the current word.

In this project, we use C-BOW to pretrain our word vector. As is illustrated in Figure 9,10, pretrained word vector will accelerate the process of convergence. While this may lower the model performance. For Example, test accuracy of BiGRU+Atttion with pretrained word vector is 83.79%, which is lower than that without pretrained word vector. This is because pretrained word vector does not precisely grasp the essence of each word. To utilize the pretrained word vector without lower the performance, we can initialize the lookup table with pretrained word vector, and do fine-tuning during the training process together with other weight.

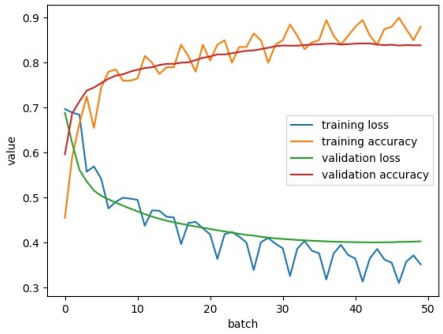

Figure 10. Attention with pretrained word vectors

4.2. L2 Regularization

When training models with sufficient representational capacity to overfit the task, we often observe that training error decreases steadily over time, while the error on the validation set begins to rise again. To avoid overfitting, we introduce L2 regularization[14] in the project. The L2 parameter norm penalty, also known as weight decay drives w closer to the origin by adding the regularization term.

4.3. Early Stopping

We also introduce early stopping strategy[8], which can be considered a type of regularization method in that it can stop the network from overfitting. The idea behind early stopping is relatively simple. (1)Split data into training and test sets. (2)At the end of each epoch (or, every N epochs), evaluate the network performance on the test set. If the network outperforms the previous best model, save a copy of the network at the current epoch. (3)Take as our final model the model that has the best test set performance. As is shown in Figure 11, 12, the whole training epoch is 50, while we used early stopping taht is 17. Test accuracy without early-stopping is 81.04%, and that with early-stopping is 82.44

5. Experiments

5.1. Datasets and Tasks

We compare and evaluate the performance of our four models on the movie review IMDb dataset[7]. This dataset contains movie reviews along with their associated binary sentiment polarity labels(pos and neg). 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).

Our evaluation task is predict whether a given re-

Figure 11. CNN training process

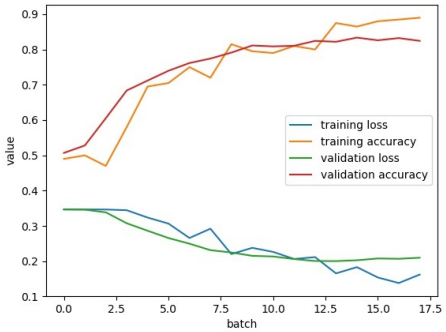

Figure 12. CNN training process with early stopping

view is positive or negative given the review text. We used the CNN, attention, Transformer model to simulate the datasets IMDb emotion classification, and through the convergence of losses, precision dimension, contrasting the efficiency of these four models. In addition, ULMFiT model is introduced the pre-training and fine-tuning strategy to text classification tasks, we pre-train a general model in wiki-103 dataset and fine-tuning it on IMDb dataset, then training a classfication about sentiment analysis.

5.2. Results and Analysis

We use pre-trained word embedding from word2vec in the training dataset. Embedding dimension of word vector is set to 128 and word hidden dimension is set to 32. We split training data into training set(90%) and validation set(5%). We fine-tune the classifier for 50 epochs and train all models. Adam is used as optimizer. L2 regularization with lambda 0.001 is used. Learning rate is 0.001. Table 1 shows the classification performance of our models. It demonstrates that CNN has lowest accuracy and very short training time. Transformer and BiGRU+Attention have similiar accuracy and training time.

Model	CNN	BiGRU+Attention	Transformer
Test accuracy	0.8104	0.8720	0.8804
Training time	19mins	2h 44mins	2h 28mins

Table 1. Performance of our models

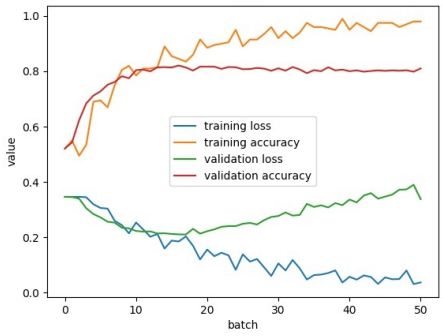

Figure 13. CNN traning process

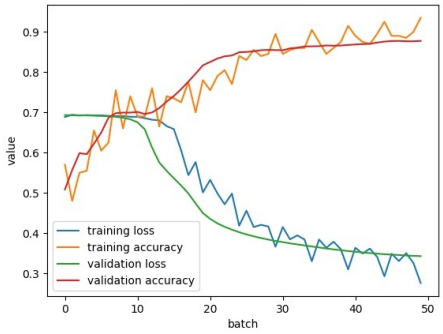

Figure 14. BiGRU+Attention training process

For CNN, it can be seen from the figure 13 that the training effect is good and the accuracy is high. At the beginning, the vibration is relatively large, and then it gradually tends to be stable, and the accuracy is stable at about 0.8.

For BiGRU+Attention, we can see that training loss curve and training accuracy curve have greater shock in Figure 14. Sometimes when loss not decrease and suddenly it become big and big, as opposed to accuracy. We had to use big batch size.

For Transformer, after 50 epochs, the training loss is 0.1611 and the validation accuracy is 0.8804. One special thing, which can be seen in Figure 15, is that it can be trained significantly faster than other models. In other words, it can reach good accuracy by training a very few epoch and converge very quickly. We

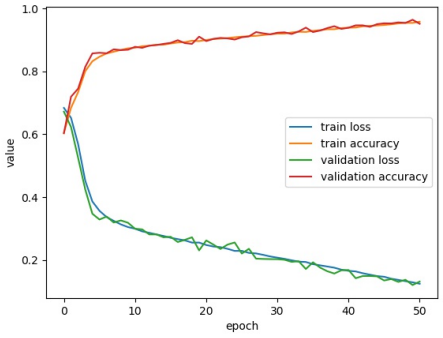

Figure 15. Transformer training process

suspect that attention mechanism helps to detect the prominent features.

6. Conclusion

In this work, we applied three models to sentiment analysis, which are CNN, BiGRU+Attention(RNN) and Transformer. CNN turns out to be less efficient on time sequence problems than RNN. While RNN generally takes much more time to train as it does not suit for parallel computation. Transformer porpose a novel approach, to deal with NLP tasks. It turns out to have good results and much smoother than RNN. Here we can learn something from them.

1. The size and quality of datasets is important in deep learning models.
2. Complex models do not necessarily outperform simple ones.
3. Without big datasets, some measures must be taken to avoid overfitting.
4. Working on deep learning, one should get good comutational resources.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv e-prints, Sept. 2014. 3

[2] Rachana Baldania. Sentiment Analysis Approaches for Movie Reviews Forecasting: A Survey. In 2017 INTERNATIONAL CONFERENCE ON INNOVATIONS IN INFORMATION, EMBEDDED AND COMMUNICATION SYSTEMS (ICIIECS). IEEE Madras Sect; IEEE Computat Intelligence Soc, 2017. International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Karpagam Coll Engn, Coimbatore, INDIA, MAR 17-18, 2017. 2

[3] Y Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. Journal of Machine Learning Research, 3:932–938, 2000. 5

[4] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. CoRR, abs/1409.1259, 2014.

[5] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555, 2014.

[6] Yoon Kim. Convolutional neural networks for sentence classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014. 2

[7] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. pages 142–150, 2011. 6

[8] C Lee, Giles Overfitting, Rich Caruana, Steve Lawrence, and Letitia Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. Advances in Neural Information Processing Systems 13, NIPS 2000, 2001. 6

[9] B Liu. Sentiment analysis: Mining opinions, sentiments, and emotions. pages 1–367, 2015. 2

[10] Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. Proceedings of Workshop at ICLR, 1, 2013. 5

[11] Rajni Singh and Rajdeep Kaur. Sentiment analysis on social media and online review. International Journal of Computer Applications, 121:44–48, 2015. 2

[12] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. Journal of machine learning research, 15(1):1929–1958, 2014. 5

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017. 3

[14] Andrew Y. Ng. Feature selection, l 1 vs. l 2 regularization,. 2004. 6