Sentiment Analysis on a New Chinese Movie Review Dataset

Elizabeth Zhou
July 31st, 2021

Abstract

This paper proposes a sentiment analysis method based on word vectors, which solves the sentiment classification of a Chinese online movie review website created by Alibaba. Different from professional movie review, online movie reviews can reflect realistic and commercial value behind the movies. By using the methods, we can find out the trend of the movie reviews and implement some actions if the reviews are too negative. With hands-on practical code, we demonstrate limitations of fully connected neural network (FCNN) and show how embeddings improve recurrent neural network(RNN) and convolutional neural networks(CNN) for the classification of sentiment.

1 Introduction

With the rapid development of the Internet, the number of Internet users increases significantly. More and more viewers tend to post their impressions and comments on movies in blogs, forums, and online videos. Such large-scale spontaneous online movie reviews, due to their unique diversity and universality, have become an important reference element for studying movies in public praise and evaluating the quality of movies.
Technical research on word embedding has gradually developed since the beginning of this century, and it is proposed in the literature [1] to use a probabilistic language model that can assign continuous encoding to each word, and extended to other word orders similar to the training word sequence. This probabilistic language model is the subsequent word vector model training laid the foundation for research.
Therefore classifying the positive and negative attitude contained in the subjective text becomes an important application in language processing. And sentiment analysis is one of the important tasks in natural language processing.

2 Dataset

Instead of using the mostly commonly used Chinese movie reviews called "Douban", we used a new data set called "Cat's eye", a ticket-buying platform that was invested by Alibaba. We split the sentences into a training set and test set and provided a dictionary of sub-phrases and unique words. All phrases (and words) have a sentiment label as well. Sentiment is labeled on a 5 point scale of 1 (most negative) to 5 (most positive), with 3 being neutral which we remove the neutral sentiment from our analysis.

	training set	test set
sentences	16979	4245

3 Methods

Sentiment analysis is one of the important tasks in natural language processing. Based on traditional sentiment analysis methods, it usually requires a lot of manpower. With rapid changes in online movie reviews, reducing manual labeling and maintenance on movie reviews is an urgent problem to be solved.

Word embedding refers to a type of language model and feature learning method in natural language processing. These technologies can map words or phrases in corpus vocabulary to low-dimensional vectors related to vocabulary size. Word embedding is a method of unsupervised learning word derived from corpus[2]. Compared with the traditional bag-of-words model that uses words as simple text features, the word embedding method is a new type of feature extraction method. It is based on deep learning algorithms to learn the syntax and semantic information which can free people from manual text annotations.

Machine doesn't understand text so we need to convert the text in machine readable language and that is nothing but the numbers. To convert text into numbers we use two different methods, one is the basic class in Keras called Tokenizer. Before implementing Tokenizer, we use Chinese segmentation tool "jieba" to separate the sentences, and then use stopword list (from Harbin Institute of Technology) to filter out meaningless words. Second method we use is the Chinese word vector, a pre-trained vectors with different properties and use them for downstream tasks[3]. For Chinese, characters (Hanzi) often convey strong semantics. The Chinese vectorization considers using word-word and word-character co-occurrence statistics for learning word vectors. The length of character-level ngrams ranges from 1 to 4 (character feature)[4].

This paper proposes a word embedding method to calculate word vectors. This method is based on the word embedding principle of deep learning, and proposes a simple and effective method for calculating comment vectors for sentiment classification. Since the learning of word vectors is obtained by unsupervised deep learning, this method does not require a lot of text labeling work or maintenance of sentiment dictionaries, thus eliminating the huge manual text labeling in sentiment analysis tasks. The comment vector is a simple vector averaging operation of the word vector, and the calculation process is less expensive, so the implementation process of the method is very simple and effective.

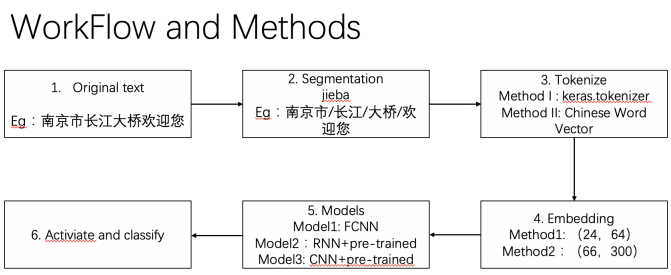

Figure 1: workflow

(1) Data Preprocessing: The corpus used in this article is Chinese corpus. Unlike English, the vocabulary of the Chinese corpus is not divided by spaces. Therefore, in text preprocessing, the text needs to be segmented first.
(2) Using the word embedding model to establish an n-dimensional vector for each word in the corpus.
(3)For each comment, use the vectors corresponding to all words in the comment to establish a corresponding comment vector.
(4) Use the classifier algorithm in machine learning to train the sentiment classifier with the comment vector in the training set. The classification algorithms used in this step include fully connected neural network, recurrent neural network and convolutional neural network.

3.1 Word Vector

This article uses the skip-gram model as a training model for word vectors. The skip-gram model is one of the most convenient word embedding models, a model that was proposed by T.Mikolov et al[5].
The training purpose of the skip-gram model is to find word representations that can help predict the surrounding words in a sentence or paragraph. The purpose of the skip-gram model is to maximize the average log probability:

$$\frac{T}{1}\sum\frac{T}{t=1}\sum -c \le j \le c, j \ne 0 log p(w_{t+j}|w_t)$$

Figure 2: formula 1

In the formula, w1, w2, w3,..., Wt are a given series of training words, and c is the size of the training text. The probability is defined as follows:

$$p(wo|wi) = \frac{exp(v'\frac{t}{wo}v_{wi})}{\sum_{w=1}^{W} exp(v'\frac{t}{w}v_{wi})}$$

Figure 3: formula 2

Where Vw and V'w are the input and output of the word vector respectively, and W is the total number of words.

The word vector calculated in this article is of the form (dim1,dim2,..., dimn) n-dimensional vector, each word in the corpus is represented by a word vector.

3.2 comment vector

Based on the word vector, this paper proposes a simple and effective method for calculating the comment vector. The comment vector is of the form (dim1,dim2,..., dimm) n-dimensional vector, the calculation method of the comment vector is shown in formula (1):

$$rv_j = \frac{\sum_{wierj} WV_i}{m}$$

Figure 4: formula 3

In formula (3), rv represents the comment vector of the jth comment, rj represents the jth comment in the corpus, wi Indicates the ith word in comment j, WVi represents the word vector of the word Wi, and m is the number of words in the comment.
The comment vector is actually the sum of the word vectors of each word in the comment. The comment vector integrates the syntactic and semantic information of all words in the corresponding comment, and becomes an effective text feature in sentiment classification, providing a new simple and effective feature extraction method for sentiment classification.

4 Models

Neural network is a nonlinear statistical data modeling tool, commonly used to model the complex relationship between input and output. A neural network is composed of a large number of neurons and interconnections between them. Each neuron represents a specific activation function. Every connection between two neurons represents a weight for the signal passing through the connection.
A neural network is made up of multiple neurons. The output of the neural network varies according to the connection mode of the network, the weight value and the activation function. In this article, the input of the input layer is the comment vector rvj=(rvj1,rvj2,...,rvjn) in the training set, and the output of the input layer is used as the input of the hidden layer and is passed in order according to the network structure.

4.1 Fully-Connected Network Without Embedding

A fully connected neural network consists of a series of fully connected layers that connect every neuron in one layer to every neuron in the other layer; and it can be used to extract, recognize, or characterize the sentiment content of the text and to classify it as positive or negative. Fully connected neural network is the simplest recurrent neural network if we unroll the time axis.

For FCNN, we basically use the plain vanilla model to have a first guess, and we get a 50 percent accuracy. We are going to compare this approach to a recurrent neural networks, a convolutional neural network and the combination with word embeddings.

4.2 LSTM + Word Embedding

Word embedding was used to obtain vector values from each term list. Vector values were used for sentiment classification using the Long Short Term Memory (LSTM) method. In this model, each label of the term list document would be processed into a vector value as a feature value. The feature value was obtained from the average vector value in the Word embedding representation in each term.The representation of Word embedding is used for initialization of each term using local tokenizer provided by keras or Chinese word vector provided by Shen Li.
LSTM method flow which is divided into 4 components by the sentiment classification [6]. Components in LSTM were used as embedding words with a binary softmax classifier which gives vector values for sentiment. The value of vectors in the binary softmax classifier would be multiplied by the matrix of other weight to produce values of 0 and 1 [7]. Effectively, it could give value parameters to positive sentiment with 1 and 0 to the negative sentiment.

4.3 CNN

CNN is one of the representative algorithms of deep learning algorithm. It includes convolution calculation and is a feedforward neural network with deep structure. Scientists have been working on convolutional neural networks since the 1980s and 1990s. After entering the 21st century, CNN have developed rapidly with the introduction of deep learning theory and the improvement of computer equipment, and people have begun to apply CNN to computer vision and natural language processing[8].
CNN was constructed through imitating the biological visual perception mechanism, which is able to perform supervised learning and unsupervised learning. The convolution kernel parameter sharing in the hidden layer and the sparsity of the connections between layers enable the CNN to obtain lattice point features with a small amount of calculation[9].
Each input in the input layer is a sentence [10]. However, this sentence is a sentence after word segmentation. In addition, the input is the word vector of each word in the sentence and one word vector corresponds to one row of the input layer in the above figure. Suppose that the comment text sentence is preprocessed into n words, each word is converted into a vector through Word2Vec word embedding, which is mapped into an m-dimensional vector, and the word sequence in the sentence is spliced and mapped into n*m-dimensional matrix.
After the convolution operation, the pooling layer performs pooling processing on each eigenvector, and a multi-dimensional vector is converted into a value after pooling processing, which is used as an element of the pooled vector[11]. The pooling method used by the pooling layer is the maximum pooling method; that is, the sequence output from the convolutional layer is input to the pooling layer. The maximum pooling method will select the largest element in the sequence and eventually obtain a new vector y.

5 Results and Analysis

model	accuracy(pp)
FCNN	49.42
RNN	75.23
CNN	75.15
RNN+Embedding	78.74
CNN+Embedding	77.59
CNN+Embedding+Ngram	78.65

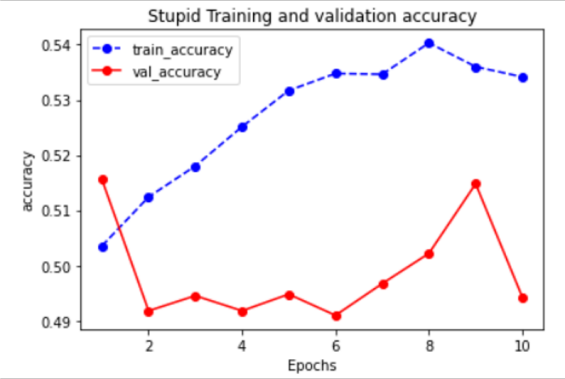

Figure 5: FCNN, guess

Plain vanilla model represents the basic guess, and we see here we have a 49.42 percentage accuracy.

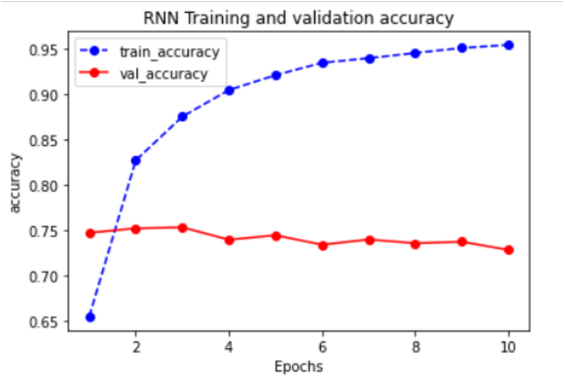

Figure 6: RNN

RNN model displays a relatively high accuracy which reaches at 75 percentage.

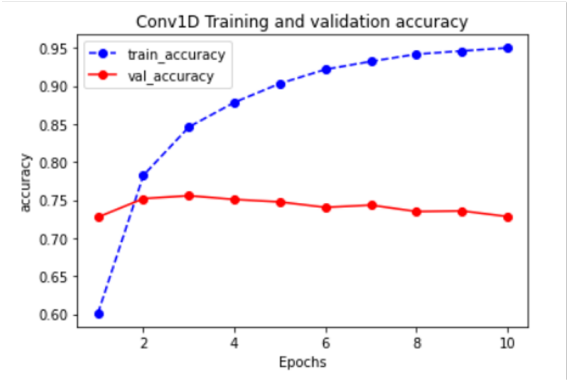

Figure 7: CNN

CNN model has lower accuracy than that of RNN but faster calculation time.

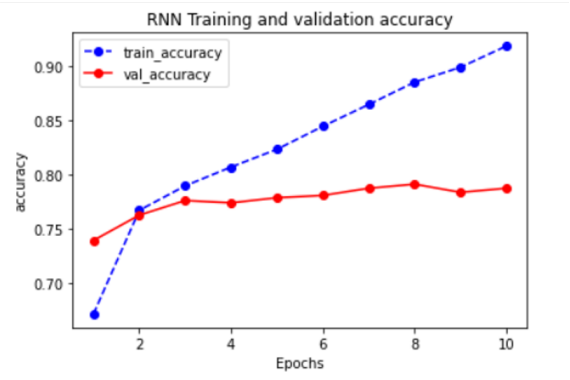

Figure 8: RNN+embedding

RNN and embedding consume significant time with CPU calculation. If we consider use GPU to improve computing power, then we can increase dataset size, and adopt deeper/wider RNN to improve accuracy.

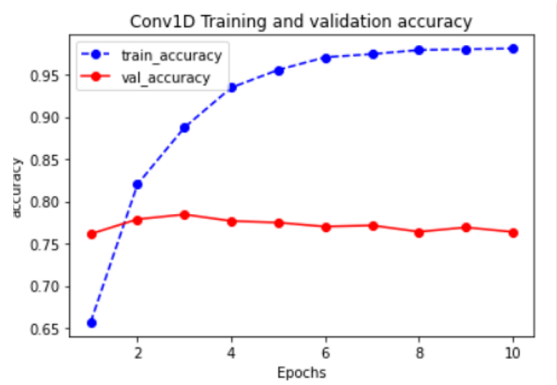

Figure 9: CNN+word embedding

The embedding matrix is initialized randomly. With CNN we can manipulate the model more easily. CNN's results is inferor than RNN. But because of the saving in computing power; in our case we can use a more complex model with bigger dataset to train the model.

6 Conclusion

By implementing the basic model, RNN and CNN, we found out that RNN has the best accuracy but takes the longest time. CNN has the second best result but saves much more time. In such large dataset, it is easy to over fit, but we can use droupout to do L1, L2 regularization. Because Chinese word vector pre-trained the vectorization, so it sets the embedding layer as non-training, therefore expand the dimension of CNN model, increasing the accuracy. And therefore this model is less likely to be overfit. In general Chinese segmentation is much harder than English segmentation due to:
a. Chinese does not have the blank.
b. The granuality of the Chinese matters.
c. Trending new words/slang can also influence the accuracy.
localized tokenizer will filter out these words for regarding them as stopwords; Chinese word vectors will filter out these words because it does not have these words in its dictionary. For our next step, we will add some special keywords that mentioned above to the jieba dictionary, and use BERT model for better results.

References

[1] Bengio Y, Ducharme R,Vincent P, et al. A neural probabilistic language model [J]. The Journal of Machine Learning Research, 2003, vol.3: pp.1137-1155

[2] Andrew L. Mass, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts, Learning Word Vectors for Sentiment Analysis, [C]. In: The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011). Stroudsburg, PA, USA: Association for Computational Linguistcs. 2011, pp 142-150

[3] Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, Xiaoyong Du, Analogical Reasoning on Chinese Morphological and Semantic Relations, ACL 2018.

[4] Yuanyuan Qiu, Hongzheng Li, Shen Li, Yingdi Jiang, Renfen Hu, Lijiao Yang. Revisiting Correlations between Intrinsic and Extrinsic Evaluations of Word Embeddings. Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data. Springer, Cham, 2018. 209-221.

[5] H. H. Do, P. Prasad, A. Maag, and A.Alsadoon, "Deep Learning for Aspect-Based Sentiment Analysis: A Comparative Review", International Journal of Expert Systems with Applications, Vol.118, pp.272-299, 2019.

[6] A. Chaudhuri, and S. K. Ghosh, "Sentiment Analysis of Customer Reviews Using Robust Hierarchical Bidirectional Recurrent NeuralNetwork", International Journal of Artificial Intelligence Perspectives in Intelligent Systems, Vol.464, pp.249-261, 2016.

[7] Bing Liu. 2012. Sentiment Analysis and Opinion Mining. Morgan Claypool, Chicago, IL.

[8] Pengfei Liu, Shafiq Joty and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP-15), pages 1433-1443.

[9] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods on Natural Language Processing (EMNLP-14), pages 1746-1751.

[10] T. Mikolov, K. Chen, G. Corrado, J. Dean, "Efficient Estimation of Word Representations in Vector Space," 2013.