UNIVERSITÄT PASSAU

Fakultät für Informatik und Mathematik

Text mining project 5981P

Learning Sentiment-Specific Word embedding for Twitter Sentiment Classification

Lavanya Bayyapu	82229
Shiva Shankar Maddila	82859
Chaitanya Gogineni	86623

CONTENTS

Abstract

Many existing algorithms model the syntactic context of words for learning continuous word representations ignoring the sentiment of the text. The problem often arises when dealing with the sentiment analysis as they map the similar syntactic context words to neighboring word vectors ignoring the opposite sentiment polarity. This issue can be addressed by implementing the sentiment specific word embedding (SSWE), which models the sentiment information in the continuous word representation. We develop the Convolutional Neural Network with pre-trained embedding (Word2vec) to effectively model the sentiment polarity of the text. For training and evaluating the model, the large corpora of data is collected from the twitter API using the positive and negative emotions. We test this model on benchmark, the result shows that the accuracy performance is further improved than some of traditional methods such as the SVM and Naive Bayes[1].

Key words: sentiment specific word embedding, Twitter Sentiment analysis; Text classification

1.Introduction

As we know opinions are central to almost all human activities and are key influence on our behaviors. Opinions, sentiments, evaluations, attitudes and emotions are the subjects of study of sentiment analysis. Sentiment analysis has grown to be one of the most active research in natural language processing. The objective is to classify the sentiment polarity of a tweet as either positive or negative.
Twitter opinion grouping is one of the well-known and appealing trends that is generally utilized as a part of request to examine inputs, surveys and suppositions towards items, people, occasions, and so forth. Emotions are the subjective feelings and thoughts of human beings. The primary emotions include love, joy, surprise, anger, sadness and fear. The concept of emotion is closely related to sentiment. For example, the strength of a sentiment can be linked to the intensity of certain emotion like joy and anger. Thus, many deep learning models are also applied to emotion analysis which are vital in sentiment analysis.
However, the existing approaches for twitter analysis are generally based on training a machine learning model and using it to predict the new tweet polarity. The features used to train the machine learning classifier are typically extracted by the traditional NLP methods such as Bag of words and TF-IDF. These traditional features did not yield to a robust model and fail to predict unseen data so that researchers tried to find another feature representation to improve the model learning performance. In this context, the Word Embedding comes as a promising alternative and becomes the most used technique to construct a vector representation of word. Word2Vec and Cw models are two words embedding models that map word to vector representation which integrates the context and the syntactic relationship between words but ignore the sentiment polarity of words. Specific-Sentiment word Embedding (SSWE) is an embedding algorithm that encodes the sentiment information in the vector representation. In this project, we will explore this technique to learn continuous representations of tweet words to train a twitter sentiment classifier. Our approach is based on training a SSWE model and then used to extract features from tweets and use them to train twitter sentiment classifier.[2]

2.Twitter streaming

To open the data stream to have Tweets delivered, we need to send a connection request to the API. In the streaming model, this connection opens up the pipeline for data to be delivered. Once the connection is established, the stream sends new tweets through the open connection. And client app will need to recognize and handle various types of messages, Inevitably, at some point the connection to the stream will close. Twitter provides samples of tweets these tweets are used for classification.

3. Data set

For the twitter sentiment analysis model, we have chosen a dataset of tweets labelled either positive or negative. This area depicts the datasets that we have utilized for this task. In this section, we will look at 3 things:
1. Separation of data into training and test sets.
2. Loading and cleaning the data to remove punctuation and numbers.
3. Defining a vocabulary of preferred words.

3.1 Separation of data into Training And Test Sets

We developed a system which can predict the sentiment of a tweet as either positive or negative. This will require the twitter data containing the emoticons. We defined the methods to classify the tweets based on the search queries for positive and negative emoticons. We used data preprocessing steps (briefly described in section 4) for cleansing the data, classified them as positive and negative tweets and placed accordingly. We divided the data into two sets namely train dataset and a test dataset. A train dataset to fit and tune our model and a test dataset to create predictions and evaluate the model at the very end.

4.Tweet Pre-processing

Preprocessing is the fundamental step in all the Natural Language Processing tasks. It strongly depends on the targeted requirements and differs from an application to another. When using the data from the twitter, it is important to format the posts containing Hyperlinks, Hash tags, Retweets, Punctuations, digits, extra spaces and so on. This will naturally make the information extraction process more complex and for that reason we applied the following preprocessing steps present in the subsections of this part of the report.

4.1 Removing Links/URLs

The URLs and Links are perhaps the most broadly utilized examples which appear to be the minimum helpful for our task. As URLs carry no information for the sentiment of a tweet, we remove all instances of URLs.

4.2 Removing user names

User mentions are used to refer to a person name. These names begin with "@" symbol followed by the user-name. User-names are used as a unique reference to a person's account and add no information for sentiment. We will remove all these twitter handles from the data as they don't convey much information.

4.3 Removing Hash Tags

Sometimes the text used with hashtag can provide useful information about the tweet. It might be risky to get rid of all the text together with the hash tag. So, we decided to leave the text intact and just remove the ''. we will do this in the process of cleaning all the non-letter characters including numbers.

4.4 Correcting Apostrophe words:

Most of the words mentioned in the tweets uses apostrophe words. We implemented the preprocessing task to correct these apostrophe words into its full word sense.

4.5 Removing Repeated characters

It is very common to use tweets like "I'm veryyy happpyyy" or similar words to express their emotions. The repeated characters present in this kind of tweets may mislead the embedder which will consider these forms of words, if not preprocessed. This can be achieved following the rule that a letter can't be repeated more than three consecutive times.

4.6 Removing Stop-words

The most common words in English appear in most of the documents, and therefore rarely help in discriminating between them. Thus, it is common to remove these words, called stop words. Remove stop-words are function words with high-frequency of presence across all sentences. It is considered needless to analyze them because they do not contain much useful information. We used the standard stop-words provided by NLTK Examples of stop words may be "the", "a", "is", and "to".

4.7 Removing of extra spaces, punctuations and digits

We also eliminated some punctuation, such as periods, commas, full stops and extra spaces between the words and digits.

5 Defining a vocabulary of preferred words

It is important to define a vocabulary of known words when using a bag-of-words or embedding model. The more words, the larger the representation of documents, therefore it is important to constrain the words to only those believed to be predictive. This is difficult to know beforehand and often it is important to test different hypotheses about how to construct a useful vocabulary. This can repeat for all documents and build a set of all known words. And can develop a vocabulary as a Counter, which is a dictionary mapping of words and their counts that allow us to easily update and query.

6. Sentiment-Specific Word Embedding

In this section, we present the details of learning sentiment-specific word embedding (SSWE) for Twitter sentiment classification. We model to give the sentiment information of sentences to learn continuous representations for words and phrases. We extend the existing word embedding learning algorithm (Collobert et al., 2011) and develop the convolutional neural network to learn SSWE. We describe the use of SSWE in an unsupervised learning framework for Twitter sentiment classification. Word embeddings are computed by applying dimensionality reduction techniques to datasets of co-occurrence statistics between words in a corpus of text. This can be done via neural networks (the "word2vec" technique), or via matrix factorization. In our project we are using Word2Vec.

6.1 Word2Vec

word2vec is a group of Deep Learning models developed by Google with the aim of capturing the context of words while at the same time proposing a very efficient way of pre-processing raw text data. This model takes as input a large corpus of documents like tweets or news articles and generates a vector space of typically several hundred dimensions. Each word in the corpus is being assigned a unique vector in the vector space. The powerful concept behind word2vec is that word vectors that are close to each other in the vector space represent words that are not only of the similar meaning but of the similar context as well. On a more general level, word2vec embeds non-trivial semantic and syntactic relationships between words. This results in preserving a rich context.[8] To use word2vec model, we need to import genism library which is a natural language processing python library. It makes text mining, cleaning and modelling very easy. Besides, it provides an implementation of the word2vec model. First, we need to get the unique vocabulary from the training set (mentioned in the section 5). Second, we need to define the parameters (training set data, training set vocabulary, n dimension space, window size and minimum count) to train the word2vec model. Later we created a text file to represent the word vectors. These created vectors are used as an input to the CNN to build the model.

6.2 CNN with pretrained word embeddings

Convolutional Neural Network can be build using Keras api. Keras is a higher level library which operates over either TensorFlow or Theano, and is intended to stream-line the process of building deep learning networks. Models in Keras can come in two forms – Sequential and via the Functional API. In our project, we used the sequential model as it can easily stack the sequential layers (and even recurrent layers) of the network in order from input to output. The functional API allows us to build more complicated architectures.
Convolution neural network is a feed-forward neural network, consisting of convolutional layers, pooling layers and fully-connected layers. Convolutional layers and pooling layers connect alternately, and output of each layer is the input of the next layer. Convolutional layer has a set of filters to a sliding window over each sentence, filters are applied to every possible window of words in the sentence and a feature is generated. Max pooling layer is to aggregate the information and to reduce the representation.[5]
To use CNN model, we need to import Keras library. First, we need to get the weights of the vocabulary created earlier (mentioned in section 5.1) and then create the embedding layer. Later, we defined the model by adding 1D convolutional layer to process the data. The arguments passed in the Conv1D() layer function is the number of output channels, kernel size and the activation function. Next, we add a 1D max pooling layer which can down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. Next, we can flatten the output from these to enter our fully connected layers. Then we can declare our fully connected layers using Dense() layer in Keras by specifying the size – in line with our architecture, each activated by a "sigmoid" function. For estimating the loss and accuracy in our defined model, we choose "binary-crossentropy" for the loss, the "adam" optimizer and a metric that will be calculated when we run evaluate() on the model. Finally, we pass the validation or test data to the fit function so Keras knows what data to test the metric against when evaluate() is run on the model.

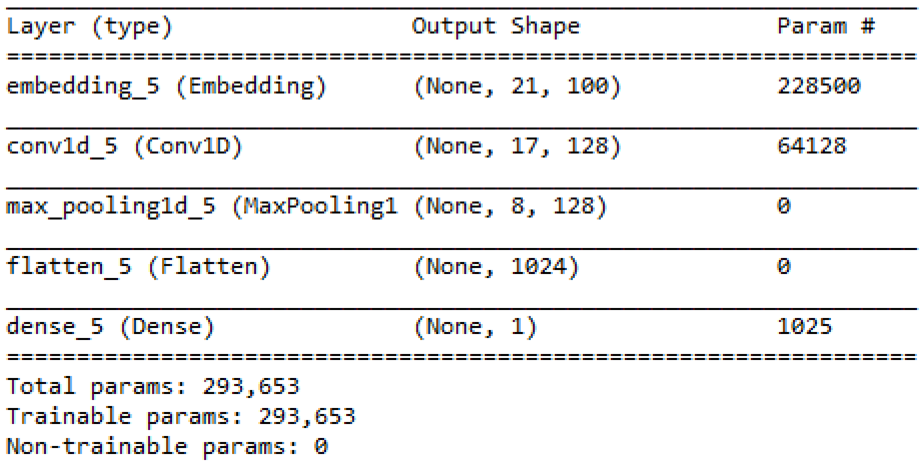
6.3 Twitter Sentiment Analysis

By using the defined model, we can predict the sentiment of any given sentence. In our project, we classified the prediction in a scale of 0 – 1 with a prediction value less than 0.5 as "Negative" and the rest value as "Positive".

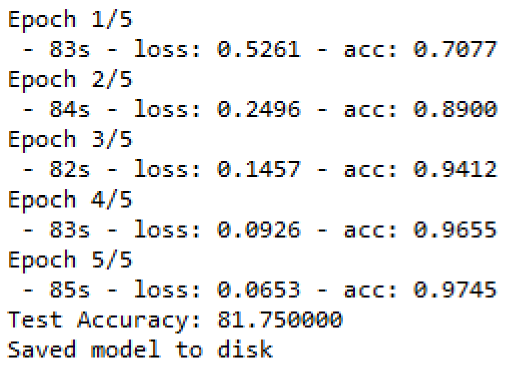
7. Results

For the training of the SSWE embedding algorithm and to evaluate our machine learning models (Word2Vec and Convolution Neural network), we have tried different parametrization metrics, namely:
1. The number of epochs of the convolutional neural network of the SSWE model.
2. The size of the embedding vector, hence the dimensionality of the embedding space.
3. Loss and accuracy parameters.

The training data set is easily learned, and the model achieved 83% accuracy on the test data set which is a better accuracy. We reached a better accuracy because of the higher quality vector trained on more data or using a slightly different training process. Our specific results may vary by the stochastic nature of neural networks.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, 21, 100)           228500
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 17, 128)           64128
_________________________________________________________________
max_pooling1d_5 (MaxPooling1 (None, 8, 128)            0
_________________________________________________________________
flatten_5 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 1025
=================================================================
Total params: 293,653
Trainable params: 293,653
Non-trainable params: 0
```



```
Epoch 1/5
 - 83s - loss: 0.5261 - acc: 0.7077
Epoch 2/5
 - 84s - loss: 0.2496 - acc: 0.8900
Epoch 3/5
 - 82s - loss: 0.1457 - acc: 0.9412
Epoch 4/5
 - 83s - loss: 0.0926 - acc: 0.9655
Epoch 5/5
 - 85s - loss: 0.0653 - acc: 0.9745
Test Accuracy: 81.750000
Saved model to disk
```

8. Conclusion

In this report, we implemented a learning continuous word representation as features for Twitter sentiment classification. We explained that the convolutional neural network combined with the predefined training algorithm boosts the performance. We trained the SSWE model with massive labeled tweets, cleansed the data, generated the vectors using Word2vec and vocabulary features to train a twitter sentiment classifier. Those features have been used later to train the CNN classifiers we have built to measure word similarity in the embedding space for sentiment lexicons. Both classifiers have yield good performance measures.

References

[1] Duyu Tang, Furu Wei , Nan Yang , Ming Zhou, Ting Liu , Bing Qin "Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification", Research Center for Social Computing and Information Retrieval

[2] D. Sai Krishna, G AkshayKulkarni, A. Mohan,"Sentiment Analysis-Time Variant Analytics",International Journal of Advanced Research in Computer Science and Software Engineering Volume 5, Issue 3, March 2015

[3] PranaliTumsare, Ashish .S. Sambare, Sachin .R. Jain."Opinion Mining In Natural Language ProcessingUsing Sentiwordnet And Fuzzy", International Journalof Emerging Trends and Technology in Computer

[4] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. Journal of machine learning research, 3(Feb):1137–1155, 2003.

[5] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning, pages 160–167. ACM, 2008.

[6] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, 1(2009):12, 2009.

[7] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. Transactions of the Association for Computational Linguistics, 3:211– 225, 2015.

[8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.