Comparative Study of Combination of Convolutional and Recurrent Neural Network for Natural Language Processing

Afshin Shirbandi and Babak Moradi

September 2, 2019

Comparative Study of Combination of Convolutional and Recurrent Neural Network for Natural Language Processing

Afshin Shirbandi

Robotic Research Center, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran

E-mail: Afshin_shirbandi@aut.ac.ir

Babak Moradi

Department of Computer (Computer engineering), Kermanshah Branch, Islamic Azad University, Kermanshah, Iran

Email: babak.moradi1988@gmail.com

Abstract

Nowadays social networks are very inclusive and there is a lot of raw information. Facebook has the most users. Most of the information on Facebook is comments. Because of this reason we choose Facebook for this research. The aim of this research is to comment mining in the Facebook social network. In this research, at the first for removing noise and data cleaning, we apply 10 preprocessing methods on the dataset. Then the data classified by using CNN, LTMS, CNN-LTMS and LTMS-CNN methods. The result showed that the most accuracy belongs to combined LTMS-CNN method and fastest approach is CNN method. We can use these methods for getting useful data on social networks.

Keywords: Text mining, Comment mining, NLP, CNN, LSTM

Introduction

Most of the people have their account on social networks (e.g. Facebook, Vkontakte) where they express their attitude to different situations and events. Facebook provides only the positive mark as a like button and share button [1].

According to the article published by zephoria.com on August 2019, nowadays Facebook has more than 2.41 billion monthly active users [21]. These users write more 510 000 comments every minute and this is a source of large information on the Internet. Usually, these textual comments are the results of the reaction of people regarding recent news or happened events. Understanding of users attitude helps to know how a certain person or groups respond to the particular topic, and it serves to draw relevant conclusions or make efficient decisions based on feedback [1-3].

All of the comments and posts on this social network are text. We can extract lots of data from those comments by Text Mining.

Text Mining [4] is one of the data mining techniques that attempts to discover new, previously unknown information by applying techniques from Natural Language Processing. Text mining is different from what are familiar with in web search. Mostly user looks into already existing data which is being written by others. The problem is pushing aside all the material that currently is not relevant to your needs in order to discover the relevant information. Text mining has different names i.e., Intelligent Text Analysis, Text Data Mining or Knowledge-Discovery in Text (KDT) [5], generally defined as the process of extracting interesting and non-trivial information and knowledge from non-structured text [6].

Text mining is used to extract interesting information, knowledge or pattern from the unstructured documents that are from different sources. It converts the words and phrases in unstructured information into numerical values which may be linked with structured information in a database and analyzed with ancient data mining techniques [7, 8].

Words are often considered as the basic constituents of texts for many languages, including English. The first module in an NLP pipeline is a tokenizer which transforms texts to sequences of words. However, in practice, other preprocessing techniques can be (and are) further used together with tokenization [9].

A CNN is basically a neural-based approach which represents a feature function that is applied to constituting words or n-grams to extract higher-level features. The resulting abstract features have been effectively used for sentiment analysis, machine translation, and question answering, among other tasks. Collobert and Weston were among the first researchers to apply CNN-based frameworks to NLP tasks. The goal of their method was to

transform words into a vector representation via a look-up table, which resulted in a primitive word embedding approach that learns weights during the training of the network [9].
In other hands, LSTM is used for learning long-distance dependency between word sequences in short texts. The final output from the last point of time is used as the prediction result [10].
We decided to use CNN method and LSTM method for Text Mining in Twitter data. We tried to improve the accuracy of our job, so we used two combined method of CNN and LSTM.

2. Method

This part has two steps. The First step is preprocessing and the second part is Text Mining by CNN method and LSTM Method in two different ways: Separate methods and combined methods.

2.1. Preprocessing

Preprocessing is an important task and essential in Text Mining, Natural Language Processing (NLP). In Text Mining, data preprocessing used for extracting interesting and non-trivial and knowledge from unstructured text data. If we want to achieve an acceptable output, preprocessing is one of the most important steps to be taken on the data. In the preprocessing, the deletion phrases that have no effect on the efficiency of the algorithm are eliminated. Sometimes one of the goals of preprocessing on data is reduces the size of them.In the following, the pre-processing on the data is applied to increase the efficiency of the algorithm Are describing.

2. 1. 1. Tokenizing

The first step involves the choice of the unit of text to analyze and the separation of the text based on the unit of analysis. This unit could be a word; however, in other cases, it may be a grouping of words or a phrase. Single words are the simplest choice and make a good starting point. It is difficult for a computer to know where to split the text. Fortunately, most text mining software contains functions to split text, because computers do not naturally sense when punctuation designates the end of a word or sentence. For example, apostrophes could indicate the end of a token, or not, depending on the use [11, 12].

2. 1. 2. Remove stop word

We want to drop frequently used filler words, or stop words, which add no value to the analysis. According to the Oxford English Dictionary, and, the, be, to, and of are the most common words in the English language. In the case of text analysis, we remove common terms because, although common terms such as these serve a grammatical purpose, they provide little information in terms of content [12, 13].

2. 1. 3. Noise removal

Noise removal is about removing characters, digits, and pieces of text that can interfere with your text analysis. Noise removal is one of the most essential text preprocessing steps. It is also highly domain dependent. For example, in comments, noise could be all special characters except hash tags as it signifies concepts that can characterize a comment. The problem with noise is that it can produce results that are inconsistent in your downstream tasks.

2. 1. 4. Lower Casting

This is the simplest preprocessing technique which consists of lowercasing every single token of the input text. Due to its simplicity, lowercasing has been a popular practice in modules of deep learning libraries and word embedding packages. Despite its desirable property of reducing sparsity and vocabulary size [9].

2. 1. 5. Stemming

Stemming is the process of ruling back a word to its root form from its derived form. It ensures that all the tokens are in its root form [6]. Stemming refers to the process of mapping each token which is generated from the previous step into its own root form. The stemming rules which are the association rules of tokens with their own root form are required for implementing it. Stemming is usually applicable to nouns, verbs, and adjectives. The list of root forms is generated as the output of this step [14].

2. 1. 6. Lemmatization

One difficulty encountered with stemming (and text analytics in general) is that a single word could have multiple meanings depending on the word’s context or part of speech. Lemmatization deals with this problem by including the part of speech in the rules grouping word roots. This inclusion allows for separate rules for words with multiple meanings depending on the part of speech. This method helps improve the algorithm by correctly grouping tokens at the cost of added complexity [12].

2. 1. 7. Normalization

Consists of matching each entity to an identifier belonging to a knowledge base that unequivocally represents its concept. For example, a protein may be mentioned by its full name or by an acronym; in this case, the normalization process should assign the same identifier to both occurrences. The identifiers can be provided by an external database or ontology [15]. Related tasks include named entity disambiguation [16], entity linking, and harmonization [17].

2. 1. 8. POS-tagging

POS tags are used to annotate words and depict their POS, which is really helpful when we need to use the same annotated text later in NLP-based applications because we can filter by specific parts of speech and utilize that information to perform specific analysis. We can narrow down nouns and determine which ones are the most prominent [12].

2. 1. 9. NP-chunking

Chunking, also called shallow parsing or light parsing, is a task that divides a sentence into non-recursive structures. The primary aim is to specify chunk boundaries and classes. Although chunking generally refers to simple chunks, it is possible to customize the concept. A simple chunk is a small structure, such as a noun phrase (NP), a verb phrase (VP), or a prepositional phrase (PP), while a constituent chunk is a structure that functions as a single unit in a sentence, such as a subject, or an object. Some natural language processing (NLP) tasks utilize the chunking because it is easier than full parsing [18].

2. 1. 10. Word2vec

Word2vec is a two-layer neural network that processes text. Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus. While Word2vec is not a deep neural network, it turns text into a numerical form that deep nets can understand [19].

2. 2. Models

We decided to use several approaches and compare the results of them. We tried to show this point, that combined methods have more performance, but they take more time to train. This section gives a brief introduction of CNN, LSTM, CNN-LSTM, LSTM-CNN methods that we used in this work.

2. 2. 1. CNNs

CNN is a class of deep, feed-forward artificial neural networks where connections between nodes do not form a cycle. CNNs are generally used in computer vision, however they’ve shown promising results when applied to various NLP tasks as well.
CNN’s are good at extracting local and position-invariant features. For tasks where feature detection in the text is more important, for example, searching for angry terms, sadness, abuses, named entities etc., CNN’s work well**.** The model used in this work has shown in figure 1.

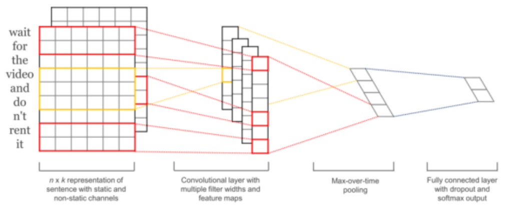
wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 1: CNN used for Sentence Classification[20]

2. 2. 2. LSTMs

Long-Term Short Term Memory (LSTMs) is a type of network that has a memory that "remembers" previous data from the input and makes decisions based on that knowledge. These networks are more directly suited for written data inputs, since each word in a sentence has meaning based on the surrounding words (previous and upcoming words).

In our particular case, it is possible that an LSTM could allow us to capture changing sentiment in a comment. For example, a sentence such as: *At first I loved it, but then I ended up hating it.* Have words with conflicting sentiments that would end-up confusing a simple Feed-Forward network. The LSTM, on the other hand, could learn that sentiments expressed towards the end of a sentence mean more than those expressed at the start.

We use LSTM models, It models word sequence x as follows:

[21] $i_t = \sigma(x_t U^i + h_{t-1} W^i + b_i)$ (1)

[21] $f_t = \sigma\left(x_t U^i + h_{t-1} W^f + b_f\right)$ (2)

[21] $o_t = \sigma(x_t U^o + h_{t-1} W^o + b_o)$ (3)

[21] $q_t = tanh\left(x_t U^q + h_{t-1} W^q + b_q\right)$ (4)

[21] $p_t = f_t \times p_{t-1} + i_t \times q_t$ (5)

[21] $h_t = o_t \times \tanh(p_t)$ (6)

2. 2. 3. CNN-LSTM Model

The first model I tried was the CNN-LSTM Model. Our CNN-LSTM model combination consists of an initial convolution. a layer which will receive word embedding as input. Its output will then be pooled to a smaller dimension which is then fed into an LSTM layer. The intuition behind this model is that the convolution layer will extract local features and the LSTM layer will then be able to use the ordering of said features to learn about the input's text ordering. In practice, this model is not as powerful as our other LSTM-CNN model proposed. This model has shown in figure 2.

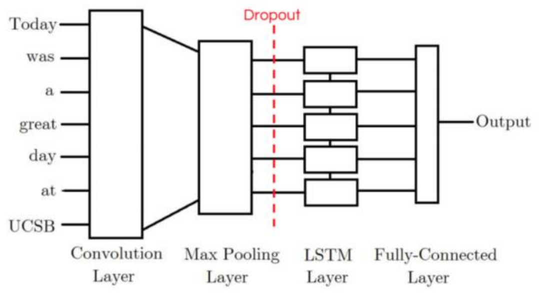

Figure 2: CNN-LSTM Model [22]

In this approach, we use DropOut technique For regularization.We Apply DropOut On Model To Prevent Co-Adaptation and OverFitting. In Our Model We Apply DropOut To Max Pooling layer Before Feeding the output of it into the LSTM layer.

2. 2. 4. LSTM-CNN Model

Our CNN-LSTM model consists of an initial LSTM layer which will receive word embedding for each token in the Comment as inputs. The intuition is that its output tokens will store information not only of the initial token, but also any previous tokens; In other words, the LSTM layer is generating a new encoding for the original input. The output of the LSTM layer is then fed into a convolution layer which we expect will extract local features. Finally, the convolution layer's output will be pooled to a smaller dimension and ultimately output as either a positive or negative label. This model has shown in figure 3.

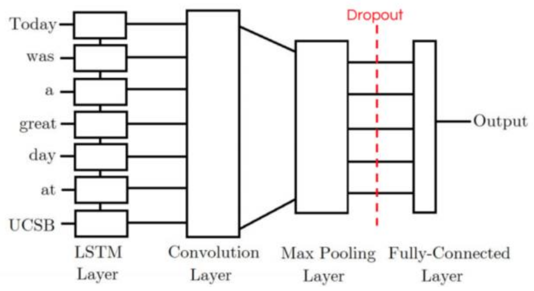

Figure 3: LSTM-CNN Model [22]

In this approach, we use DropOut technique For regularization.We Apply DropOut On Model To Prevent Co-Adaptation and OverFitting. In CNN model We Apply DropOut To Max Pooling layer Before Feeding the output of it into Fully-Connected layer.

2. 3. Facebook Data

Implementation of batch data processing makes sense in the case of high volumes of data. Firstly, we choose a topic, which is popular recently. For each post, using the Facebook Graph API, all comments have been collected during the first 12858s. Data is stored in flat table format (e.g. CSV file) which is easy to save in a distributed file system. The header of CSV file contains the following columns: [Datetime] [Topic] [Post] [Comment] [Positive] [Negative].

3. Results

After preprocessing and use all of four approaches, we compared results of approaches. Two important feathers were accuracy and time because by, both of these feathers, you can have a better choice to better results.

In figure 4 we compared the accuracy of four approaches and LSTM-CNN has best result. the Final accuracy of all approaches has shown in table 1.

Table 1: Final accuracy of approaches

Approach	CNN	LSTM	CNN-LSTM	LSTM-CNN
Accuracy	0.75	0.79	0.82	0.86

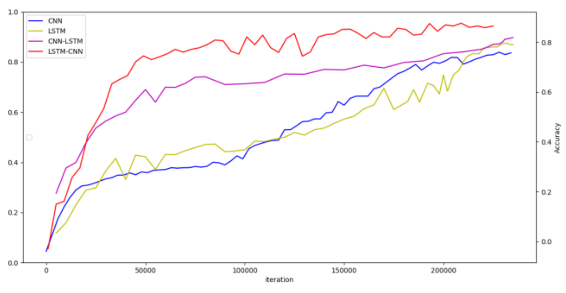

Figure 4: compare among Accuracy of CNN, LSTM, CNN-LSTM, LSTM-CNN approaches

In table 2 we showed the result of times that spends to train approaches. And in figure 5 we compared these times together. In this feature, the CNN approach has the best result.

Table 1: Time for train approaches

Approach	CNN	LSTM	CNN-LSTM	LSTM-CNN
Time (min)	27	35	150	190

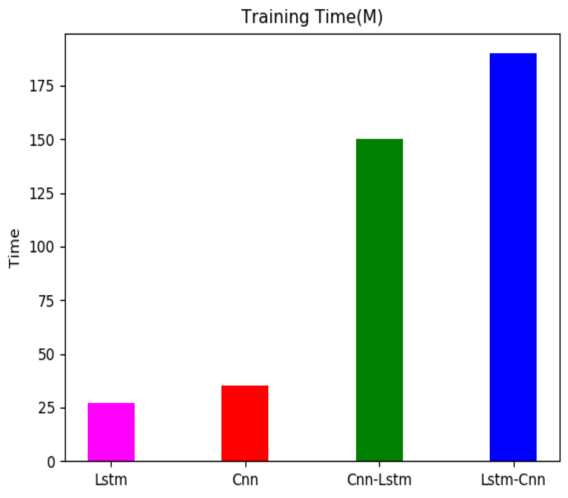

Figure 5: Comparison of network training times

Conclusions

In this paper at the first, we preprocessed all comments in 10 steps. Then we used four approaches for processing comments. Due processing, we calculated the Accuracy and Time of training of each approach. Finally, we compared the results of those features. In Accuracy, LSTM-CNN method has the best result and in Time of training CNN method has the best result. we found out that combined CNN and LSTM approach had more efficient than a single method. About these results, we can say, when you need more Accuracy you must spend more time and it is better to use combination methods, and when you need a faster method you will have less accuracy, in this case, it is better to use a single method.

References

[1] H. Tran and M. Shcherbakov, "Detection and prediction of users attitude based on real-time and batch sentiment analysis of facebook comments," in *International Conference on Computational Social Networks*, 2016, pp. 273-284.

[2] N. Godbole, M. Srinivasaiah, and S. J. I. Skiena, "Large-Scale Sentiment Analysis for News and Blogs," vol. 7, pp. 219-222, 2007.

[3] (2019). *The Top 20 Valuable Facebook Statistics – Updated July 2019*. Available: https://zephoria.com/top-15-valuable-facebook-statistics/

[4] W. J. S. o. T. M. C. Berry Michael, Classification and S. V. Retrieval", New York, LLC, "Automatic discovery of similar words," pp. 24-43, 2004.

[5] H. Karanikas and B. J. C. f. R. i. I. M. Theodoulidis, Department of Computation, "Knowledge discovery in text and text mining software," 2002.

[6] K. Prasad, S. Saritha, and D. J. I. J. o. C. A. Saxena, "A Survey Paper on Concept Mining in Text Documents," vol. 166, pp. 7-10, 2017.

[7] S. Kannan and V. Gurusamy, "Preprocessing techniques for text mining," in *Conference Paper. India*, 2014.

[8] S. Vijayarani and R. J. A. C. I. A. I. J. Janani, "Text mining: open source tokenization tools-an analysis," vol. 3, pp. 37-47, 2016.

[9] J. Camacho-Collados and M. T. J. a. p. a. Pilehvar, "On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis," 2017.

[10] J.-H. Wang, T.-W. Liu, X. Luo, and L. Wang, "An LSTM Approach to Short Text Sentiment Classification with Word Embeddings," in *Proceedings of the 30th Conference on Computational Linguistics and Speech Processing (ROCLING 2018)*, 2018, pp. 214-223.

[11] S. M. Weiss, N. Indurkhya, T. Zhang, and F. Damerau, *Text mining: predictive methods for analyzing unstructured information*: Springer Science & Business Media, 2010.

[12] M. Anandarajan, C. Hill, and T. Nolan, "Text Preprocessing," in *Practical Text Analytics*, ed: Springer, 2019, pp. 45-59.

[13] W. J. Wilbur and K. J. J. o. i. s. Sirotkin, "The automatic identification of stop words," vol. 18, pp. 45-55, 1992.

[14] C. C. Aggarwal and C. Zhai, *Mining text data*: Springer Science & Business Media, 2012.

[15] Y. Tsuruoka, J. McNaught, and S. Ananiadou, "Normalizing biomedical terms by minimizing ambiguity and variability," in *BMC bioinformatics*, 2008, p. S2.

[16] R. Bunescu and M. Paşca, "Using encyclopedic knowledge for named entity disambiguation," in *11th conference of the European Chapter of the Association for Computational Linguistics*, 2006.

[17] A. Lamurias, F. M. J. E. o. b. Couto, and c. biology, "Text mining for bioinformatics using biomedical literature," vol. 1, 2019.

[18] O. Aslan, S. Gunal, B. T. J. I. P. Dincer, and Management, "On constituent chunking for Turkish," vol. 54, pp. 1262-1276, 2018.

[19] D. Zhang, H. Xu, Z. Su, and Y. J. E. S. w. A. Xu, "Chinese comments sentiment classification based on word2vec and SVMperf," vol. 42, pp. 1857-1863, 2015.

[20] Y. J. a. p. a. Kim, "Convolutional neural networks for sentence classification," 2014.

[21] W. Yin, K. Kann, M. Yu, and H. J. a. p. a. Schütze, "Comparative study of CNN and RNN for natural language processing," 2017.

[22] (2018, Aug). *Twitter Sentiment Analysis using combined LSTM-CNN Models*. Available: http://konukoii.com/blog/2018/02/19/twitter-sentiment-analysis-using-combined-lstm-cnn-models/