Aspect-based sentiment Analysis con Recurrent Neural Networks

Sebastian Coronado*

Catholic San Pablo Univesity
sebastian.coronado@ucsp.edu.pe

January 26, 2018

This paper focus on using Twitter, the most popular microblogging platform, for the task of sentiment analysis using Recurrent Neural Networks.

I. Introduction

Microblogging today has become a very popular communication tool among Internet users. Authors write about their life, share opinions on a variety of topics.

We used a dataset provided the Workshop on Semantic Analysis at SEPLN. The dataset has thousands of tweet to learn. In our research we use the Spanish language. However, this method can be adapted to any other language.

This paper seeks to use the RNN to classify between: positive, negative, none and equivalent because we're trying to predict if this text has positive or negative sentiment.

Using an RNN rather than a feedforward network is more accurate since we can include information about the sequence of words. In this paper we use a input dataset for training with their respective labels for training.

II. Related work

The population of blogs and social networks, opinion mining and sentiment analysis became a field of interest for researchers. In [1], the authors use web-blogs to construct a corpora for sentiment analysis using emoticon icons. The authors applied SVM and CRF learners to classify sentiments at the sentence level and then investigated several strategies to determine the overall sentiment of the document. As a result, the winning strategy is defined by considering the sentiment of the last sentence of the document as the sentiment at the document level.

A different approach [2] exploits hierarchical structure and uses positional semantics to understand sentiment. The main idea is to use autoencoders that learn a reduced dimensional representation of fixed size inputs. They can be used to efficiently learn features encodings which are useful for clasification. recursive auto-associative memories (RAMMS) are simular to RNN in that they are a connectionist feedforward model. However, RAAMS learn only for fixed recursive data. The current work is related in using recursive deep learning models. However, RNNs require labeled tree structures and use a supervised score at each node. Instead, RAEs learn hierarchical structures that are trying to capture as much of the original word vectors as possible. The learned structures are not necessarily syntactically plausible but can capture more of the semantic content of the word vectors. Instead of document level sentiment classification, [2] analyze the contextual polarity of phrases and incorporate many well designed features including dependency trees.

Finally [3] represent document with

*Proyecto Final

convolutional-gated recurrent neural network, which adaptively encodes semantics of sentences and their relations. They verify the effectiveness of LSTM in text generation task.

III. Definitions

This section defines concepts to understand the algorithms used in this paper. Recurrent neural networks models compute compositional vector representations for word sequences of arbitrary length. These high level distributed representations are then used as features to classify each token in the sentence.

i. Long Short-Term Memory RNN

Long Short-Term Memory(LSTM) is specifically designed to model long term dependencies in RNNs. The recurrent layer in a standard LSTM is constituted with a special (hidden) unit called memory blocks. A memory block is composed of four elements: (i) a memory cell with a self connection, (ii) an input gate i to control the flow of input signal into the neuron, (iii) an output gate o to control the effect of the neuron activation on other neurons, and (iv) a forget gate f to allow the neuron to adaptively reset its current state through the self-connection. LSTMs are usually trained using truncated or full BPTT.

ii. Bidirectionality

Notice that the RNNs such only get information from the past. Sometimes is beneficial to know the next word. The unidirectional LSTM can be extended to bidirectional LSTM by allowing bidirectional connections in the hidden layer.

iii. Word Embeddings

Word embeddings are distributed representations of words, represented as real-valued, dense, and low dimensional vectors. Each dimension potentially describes syntactic or semantic properties of the word.

IV. Experiments

In this section we present our experimental settings and results and experimental setup.

Using a RNN is more accurate than a feed-forward network since we can include information about the sequence of the words. The architecture for this network is shown in figure 1.

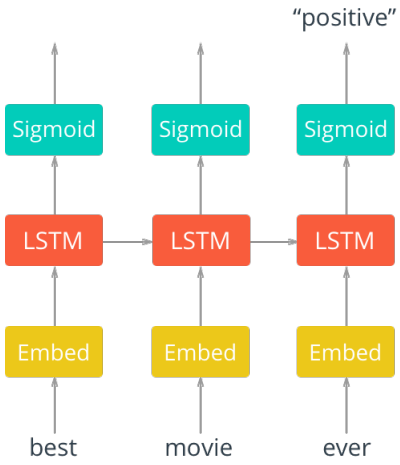

Figure 1: *Network Diagram*

The first step when building a neural network model is getting data into the proper form to feed into the network. First we remove all punctuation. Then, get all the text without the newlines and split it into individual words. After that we need to encode words and labels into numbers. Finally we need to delete tweets that have empty comments and truncate the maximum length of words. The result is a matrix of features where each feature is the encoding of a comment (hot enconding).

The second step is to pass words to the embedding layer. Here we use the word to vector embedding, that is, creating a dictionary with all the words in text and then converting tweets into number vectors. It is massively inefficient to hot encode our tweets. We create the embedding lookup matrix using Tensorflow variable.

The building of the graph relays on:

1. Number of units of the LSTM Cells. Usually larger is better.

2. Number of LSTM layers in the network. Add more if the model is underfitting.

3. Batch size is the number of reviews to feed the network in one training pass.

When creating the LSTM cells we use Tensorflow's LSTM cell and then we add a dropout, it's a really convenient way to avoid overfitting.

The accuracy we obtained was 56/100, which is good because the best accuracy obtained for this set was 70/100.

The comparison with other methods is not completely accurate since they only classify positive or negative sentiments, in our approach we classify to 4 different classes, thus the precision is less accurate. We also tried this method for classifying only for 2 classes and we got an accuracy of 76/100.

Comparison with other methods	
Our Approach	58/100
[2]	72/100
[3]	86/100
[1]	75/100

V. Conclusions

The purpose of this paper is to give a approach of how to implement recurrent neural networks for sentiment analysis. We think the embedding is key when obtaining better classification rates. The hot encoding embedding gives to many problems and it is not efficient. The dropout cell is very convenient for saving time when representing an embedding. Finally our result is 56/100.

References

[1] Cícero Nogueira dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 69–78, 2014.

[2] Cícero Nogueira dos Santos. Think positive: Towards twitter sentiment analysis from scratch. In *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014.*, pages 647–651, 2014.

[3] Bing Liu. Sentiment analysis and subjectivity. In *Handbook of Natural Language Processing, Second Edition. Taylor and Francis Group, Boca*, 2010.