Twitter Sentimental Analysis using Deep Learning

Sreeja Kamishetty
201402200
kamishetty.sreeja@research.iiit.ac.in

Hemanth Kumar Veeranki
201401145
hemanth.veeranki@students.iiit.ac.in

***Abstract*— Micro-blogging today has become a very popular communication tool among Internet users. Millions of users share opinions on different aspects of life everyday. Therefore micro-blogging web-sites are rich sources of data for opinion mining and sentiment analysis. Because Micro-blogging has appeared relatively recently, there are a few research works that were devoted to this topic. In our paper, we focus on using Twitter, the most popular micro-blogging platform, for the task of sentiment analysis. The purpose of this project is to build an algorithm that can accurately classify Twitter messages as positive or negative, with respect to a query term. Our hypothesis is that we can obtain high accuracy on classifying sentiment in Tweets using machine learning techniques.**

***Index Terms*— Sentiment Analysis, Deep learning, NLP, Data Mining**

I. INTRODUCTION

Sentiment Analysis is the process of computationally determining whether a piece of writing is positive, negative or neutral. Its also known as opinion mining, deriving the opinion or attitude of a speaker. Sentimental Analysis is useful in many areas like Business, Politics, Public Actions where we need to understand customers and opinions towards products, government, public decisions.

Millions of messages are appearing daily in popular web-sites that provide services for micro-blogging such as Twitter, Tumblr, Facebook. Authors of those messages write about their life, share opinions on variety of topics and discuss current issues. Because of a free format of messages and an easy accessibility of micro-blogging platforms, Internet users tend to shift from traditional communication tools (such as traditional blogs or mailing lists) to micro-blogging services. As more and more users post about products and services they use, or express their political and religious views, micro-blogging web-sites become valuable sources of peoples opinions and sentiments. Such data can be efficiently used for marketing or social studies.

We use a dataset formed of collected messages from Twitter. Twitter contains a very large number of very short messages created by the users of this micro-blogging platform. Data from these sources can be used in opinion mining and sentiment analysis tasks.

In this project we aimed at measuring the sentiment of tweet as positive or negative using different algorithms.

II. DATASETS

We have used the Twitter Sentiment Analysis Dataset[1] which is made of about 1,400,000 labeled tweets. The dataset is quite noisy and the overall validation accuracy of many standard algorithms is always about 70%.

III. PRE PROCESING DETAILS

Pre-Processing is essential for efficient Feature Extraction. Also preprocessing helps in removing redundant or use less data which would reduce the noise in the data. The main steps involved in the Pre-Processing Stage of the Pipeline are :

- Hash-tags and Mentions removal
- URL removal

A. Hash-tags and Mentions removal

Hash-tags and mentions are generally of no use in determining the sentiment of tweets in most of the cases. Because under same hashtag or mentioning same person, tweets can be made negatively and positively in many of the cases. So they are stripped off from the tweets before the start training on them.

B. URL Removal

URLs can have minimal say in the emotion of tweet that too in very rare cases. So they are striped of from data to make data and the trained language model more robust.

IV. METHODS

We used Keras [1] to develop both the models of our neural networks.

A. LSTM on word embeddings

Recurrent Neural networks (RNN) are generally used for NLP related deep learning applications. In a traditional RNN, during the gradient back-propagation phase, the gradient signal can end up being multiplied a large number of times (as many as the number of time steps) by the weight matrix associated with the connections between the neurons of the recurrent hidden layer. This means that, the magnitude of weights in the transition matrix can have a strong impact on the learning process. This can lead to vanishing or exploding gradients problem. These issues are the main motivation behind using the LSTM [2] model. We used the standard tokenizer from keras [1]. This splits the sentence

[1] http://thinknook.com/wp-content/uploads/2012/09/Sentiment-Analysis-Dataset.zip

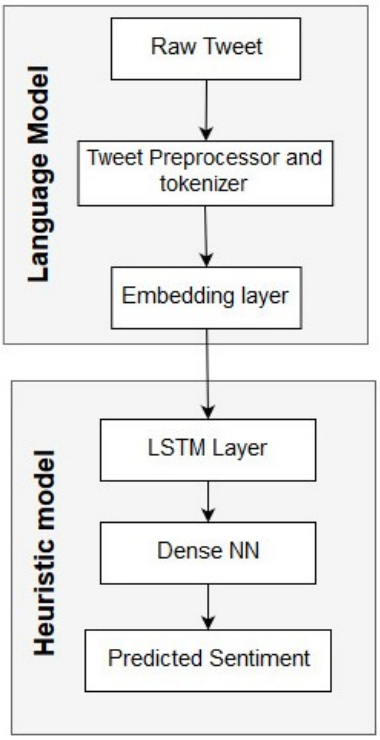

Fig. 1. Model A Architecture

with delimiter as space and tokenizes the given tweet. We then give this tokenized tweets to an embedding layer which replaces word tokens with their word embeddings. This later passes through LSTM and couple of dense layers with sizes 32 and 1 respectively to give us the final prediction. The architecture is shown in Figure 1

B. Fully connected neural networks on phrase embeddings

In this model, we first obtained phrase embeddings[3] for the given tweets. Then passed these phrase embeddings to a Fully connected neural network and obtained the prediction. In the fully connected neural network these are the specifications of the layers we used.{128,Tanh},{64,ReLu},{33,Tanh} and {1,sigmoid} respectively starting from the beginning layer (written as {Number of nodes, Activation function}). Phrase embeddings are obtained by first tokenizing the tweets using TweetTokenizer from nltk[4] library and then passing these tokenized representations of tweets to Doc2Vec training model of gensim [5] library after removing stopwords from them. The architecture is shown in Figure 2

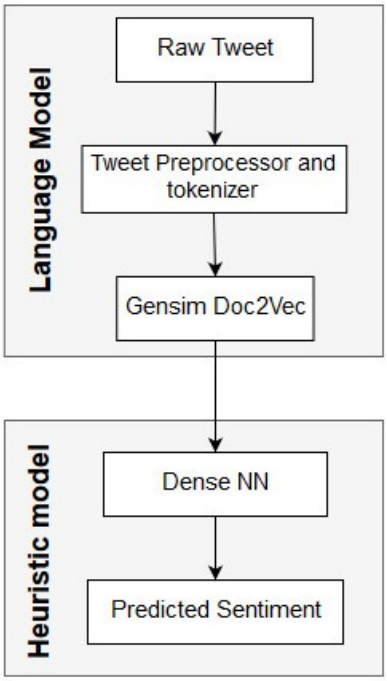

Fig. 2. Model B Architecture

V. Evaluation Measures

We used precision, recall and f1 score as evaluation measures for the models we evaluated.We used sklearn [6] library for calculating the above mentioned measures.From our experiments LSTM with embedding layer gave us the most accurate results with an accuracy of 82.6%

VI. Experiments and Results

We experimented with the LSTM model by changing the number of words we are considering for tokenization. As illustrated in table I even if we take only 50K most popular words which is around one-fourth of the total word count we were able to get similar accuracy. Using less words will have lesser parameters in the total model and hence can prevent over-fitting. Similarly we experimented with the FCNN model by changing the size of phrase embeddings. As illustrated in table II phrase embedding with size of 300 gave us the best results.

TABLE I
F1 scores for various word sizes in LSTM model

Word size	f1	Model size (parameters)
All words(2,17,961)	82.1	27.9 M
1,00,000	82.1	12.8M
50,000	82.6	6.4M

TABLE II

F1 SCORES FOR VARIOUS DOC2VEC SIZES IN FCNN MODEL

Phrase embedding size	f1
300	71.0
250	- 71.5
200	71.7

REFERENCES

[1] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[4] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[5] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[6] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.