Project Report: Twitter Text Sentiment Prediction

Fereshte Mozafari, Mohammad Tohidi Vahdat, Ehsan Mohammadpour
fereshte.mozafari@epfl.ch, mohammad.vahdat@epfl.ch, ehsan.mohammadpour@epfl.ch,
EPFL, Switzerland

***Abstract*—Twitter sentiment Classification (whether they are positive or negative) has been studied in recent years. In this paper, we provide some of the most-known challenges of pre-processing and representing Twitter data. Moreover, we evaluate some of the state-of-the-art neural architectures to classify the sentiment of twitter messages. We use GloVe embedding to obtain feature representation for tweets. We show that our pre-processing techniques as well as a proper representation of tweets provide a good performance for neural models. We compared different neural models and found that a long short-term memory (LSTM) recurrent neural network with 4 layers is the best suited model for twitter sentiment classification. The results show that our proposed neural model outperforms baseline Logistic Regression and Support Vector Machine up to 8.2% and 8.8%, respectively. Submitting on crowdAI, we managed to get the classification accuracy of 86.6% and the F1-score of 0.871.**

I. Introduction

Nowadays we can observe billions of messages being carried daily on social networks such as Twitter, Instagram, Facebook, and other sites. These messages include many information about people's opinion and feelings. Hence, developing tools for automatic sentiment classification of these messages is attracted. In this paper we focus on Twitter messages, and we work on the problem of Twitter Sentiment Classification.

Twitter sentiment classification has attracted research interest in recent years [1], [2]. The goal is to predict whether a Twitter message, called *tweet*, is expressing a positive or negative feeling; therefore, it is categorized as the binary classification problems.

The text data, and the text pre-processing is a crucial step to remove the words that are neutral or meaningless for the classification. This filtering improves the efficiency of training phase and accuracy of the classification. Next, it is mandatory to represent the input data into a set of numerical values in order to be used for the training phase. For the text input data, a good representation is the one that reflects their semantics.

Various approaches have been proposed in order to perform feature representation. Among them, we selected term-frequency and word embedding. The former is used for the baseline methods and the later is used to fed into the deep learning neural approaches.

Finally, the input data that are transformed into a numerical feature matrix is used to train the model. For the baseline, we chose Regularized Logistic Regression and Support Vector Machine. Then, we built deep learning architectures including convolutional and recurrent neural networks. At the end, we showed that the long short-term memory (LSTM), which is a type of recurrent neural networks, performs better than the other models.

The rest of the paper is organized as follows. In Section II, we explain the data pre-processing in order to eliminate neutral and meaningless data. Section III describes the embeddings that are used in this paper. The baseline methods are discussed in Section IV. Section V describes our proposed deep learning architectures. The results of our implementations are reported in Section VI. Finally, we conclude the paper in Section VII.

II. Data pre-processing

An important step to have a decent classification is to eliminate invaluable data from the given input data-set. We are given two input files that contain positive and negative tweets. To refine the input files, we did the pre-processing steps that are explained as follows.

- **Replacing emojis with appropriate words:** Emojis are commonly used in the messaging applications to express feeling or to shorten the tweet, e.g., one can replace 'it was funny' by putting a 'smiley' emoji. The idea is to replace the emojis with the corresponding main word, e.g., in the aforementioned example, the emoji is replaced with 'smile'.
- **Removing numbers:** Numbers do not express the absolute sentiment of a tweet. While, numbers sometimes can express positive or negative feeling in tweeter messages. For instance, consider these two sentences: 'I bought 1 car' and 'I have 1 CHF', where both are using the number 1. The former is expressing a positive feeling from a user informing they bought a car, while the later is considered as a negative one showing sadness of having little money. Here, for simplicity, we decided to remove the numbers since they do not really express the absolute sentiment of a tweet.
- **Removing tags:** We have noticed that tweets contain two tags: <user> and <url>, expressing the Twitter ID of a user and a URL[1] of an external resource. They

[1]Uniform Resource Locator

are not informative in the term of emotions; therefore, we deleted them from the input files.

- **Replacing heart sign:** The GloVe database includes a special token 'hear' for the heart sign < 3. As this sign can show a positive sentiment, we replace it with token 'heart'.
- **Removing elongated endings:** As GloVe database does not include elongated endings words, we remove those endings and keep the normal word.
- **Splitting concatenated words:** We GloVe does not provide concatenated words with special sign of '/', we split them in order to use their representations. For example, we can split 'happy/sad' as 'happy' '/' 'sad' words.

Moreover, we examined some other pre-processing techniques which reduced the accuracy of our models, so we did not consider them in our final model. These consist of removing stop-words, stemming, and lemmatization.

III. Feature representation

In the previous section, we captured the useful words in each tweet. The goal of this section is to find a numerical feature representation for each tweet. We mainly focused on two methods namely, tf-idf vectors and word embedding. We describe them in details as follow.

A. *tf-idf vectors*

tf-idf is a document-term matrix in which each row represents a document (tweet in this project) and each column represent a term (word). For each row, only the terms in the corresponding document have a non-zero value; the value is a product of the term frequency and the inverse document frequency. The term frequency shows how frequently a term is in a document and the inverse document frequency indicates in how many documents the term is used.

B. *Word embedding*

Word embedding is a well-known method in the context of Natural Language Processing research to numerically represent the words. In fact, it is a mathematical embedding from a space with many dimensions per word to a continuous vector space with a much lower dimension.

Training word embeddings is a cumbersome task as it should be trained on huge data-sets and hence requires a lot of training time. Therefore, we decided to use a pre-trained set of word embeddings, called GloVe[2]. There are multiple dimensions for GloVe from the range of 25 to 300. Note that 300 dimensions is used from Common Crawl. Although increasing the dimensionality of the embedding improves the accuracy of model, it also increases the time required to train a model. At the end, each tweet is represented as a sequence of word embeddings to be fed into our deep learning models.

[2]http://nlp.stanford.edu/projects/glove/

IV. Baseline

We consider regularized logistic regression and support vector machine as two baseline methods. The main features of them are simplicity of implementation and fast training. In both methods, we use tf-idf as feature representation.

A. *Regularized Logistic Regression*

Regularized Logistic Regression (RLR) is known to be used for classification purposes. We also rediscovered this in the first project of the course where RLR provide better accuracy in comparison with others including least square and regularized ridge regression. Therefore, we selected this method as one of the baseline methods. The numerical evaluation of the method is provided in Section VI.

B. *Support Vector Machine*

Support Vector Machines (SVMs) are used for both regression and classification problems; specifically, they provide a maximum-margin decision boundary in the classification problem, which in turn causes the classification to be more robust. Details of implementation is provided in Section VI.

V. Deep Learning

Deep learning allows computational models that are composed of multiple processing layers in order to learn representations of data with multiple levels of abstraction [3]. We provide three deep learning architectures namely convolutional neural network, gated recurrent neural network and long short-term memory.

In the all aforementioned architectures, we have the following general layers:

- The first layer is an embedding layer. We initialize this layer with GloVe word embeddings (Section III). It it noteworthy to mention that weights are trainable and may change during the training phase.
- There are multiple layers corresponding to each deep learning architecture described later in this section.
- The one before the last layer is the drop-out layer. Here, the goal is to avoid over-fitting by dropping randomly a defined fraction of the hidden units out in each iteration. In other words, since in deep learning methods, we are dealing with lots of parameters, our model might suffer from over-fitting problem; drop-out layer is a solution to this issue.
- The last layer is the dense layer which is a fully connected network; this layer generates the final prediction for the sentiment of the tweets.

Moreover, we use categorical cross entropy as the loss function for all the aforementioned architectures as the goal is to solve a classification problem.

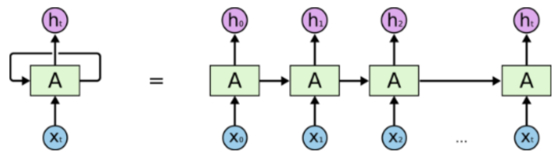

Figure 1. An unrolled recurrent neural network. 'A' is a recurrent unit (LSTM or GRU), x_t is the input content and h_t is the state output that is also used in the next computation round of the current state.

A. *Convolutional neural network*

Convolutional neural networks (CNN) use multiple layers with convolving filters that are applied to local features. In our CNN implementation, we have only one convolutional layer followed by a max pooling layer:

- The convolutional layer: it uses a filter size of 3 implying that it extracts the local features around each word window with the size of 3. The activation function of this layer is a rectified linear unit (ReLU).
- The max pooling layer: It extracts the most important features in the feature map obtained from the convolutional layer.

B. *Recurrent neural network*

Recurrent Neural Network (RNN) can effectively make use of sequential information. In traditional neural networks, we assume that all inputs (and outputs) are independent from each other; however, for many tasks, this is not the case. For example, to predict the next word in a sentence, it is important to know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence that is illustrated in Figure 1. RNNs have a 'memory' which captures information about what has been calculated so far. Therefore the output of RNNs are dependent on the previous computations.

We have implemented two common forms of recurrent layers, namely long short-term memory and gated recurrent units.

1) Long short-term emory: LSTMs have a Nature of Remembering information for a long periods of time in their Default behaviour. Therefore they help to extract dependencies between further words within a document. They consist of three gate called forget gate, input gate and output gate. Forget gate decides how much of the past should be remembered. Input gate decides how much of the current LSTM unit should be added to the current state and output gate decides which part of the current unit makes it to the output (which will be fed to the next LSTM layer).

In addition to the aforementioned advantage, they also help to overcome the vanishing gradient issue which is a common issue with RNNs.

2) Gated recurrent neural network: GRU aims to solve the vanishing gradient problem which comes with a standard recurrent neural network. GRU can also be considered as a variation on the LSTM as they are both designed in a similar fashion. GRU consists of two main gates, namely the update gate and the reset gate. The former helps the model to determine how much of the past information (from previous time steps) needs to be passed along to the future and the latter is used to decide how much of the past information to forget.

VI. Results & Discussion

For evaluating and comparing the accuracy of our models, we did a local validation set. 10% of data is used for validation set and the remaining as train set.

Lots of classification schemes are tested throughout this project. First, to have an estimation about the baseline, simple classifiers including SVM and Logistic Regression are implemented. We selected these two models because they are simple and most commonly use in the task of classification. Table I shows the accuracy for these two methods.

Table I
Experimental results of baseline methods using TF-IDF input with 100 dimensions.

Model	Accuracy
Support Vector Machine	77.8%
Regularized Logistic Regression	78.4%

In order to improve our classification in terms of accuracy, three deep learning architectures (CNN, GRU, and LSTM) are implemented. To compare our models, we used GloVe embedding with 100 dimensions. Moreover, in all three schemes, we enable the back-propagation parameters to train our architectures more accurately. Note that all networks are trained for five epochs and the test accuracy is evaluated on the crowdAI test set. A convolutional neural network (described in Section V-A) leads to the accuracy of 82%, while with GRU neural network (see Section V-B) we are able to get the accuracy up to 85.6%. Finally, the best accuracy is obtained using LSTM model (explained in V-B) with the accuracy of 86.2%. This is due to the fact that the LSTM is able to use the global context in the sequence of words to make predictions, while CNN only uses local context and does not take into account global dependencies. Furtheremore, we combined two models LSTM and CNN to get better result, We observed that LSTM-CNN model perform almost as good as the basic LSTM. Therefore, we left it and selected LSTM as our best model to train twitter dataset over GloVe embeddings with different dimensions.

We evaluated the effect of using different feature vector dimension of GloVe. To do so, we trained our model with GloVe embeddings of 25, 50, 100, and 200 dimensions, that are trained on almost two billion tweets. Then, we

Table II
EXPERIMENTAL RESULTS OF DEEP LEARNING MODELS USING GLOVE EMBEDDING WITH 100 DIMENSIONS.

Model	Accuracy
Convolutional Neural Network	82%
Gated Recurrent Unit	85.6%
Long Short-Term Memory	86.2%

also performed training with the GloVe embeddings with 300 dimensions, which trained on Common Crawl data-set containing about 2.2 Million different vocabularies.

Table III
EXPERIMENTAL RESULTS OF LSTM USING GLOVE EMBEDDINGS WITH VARIOUS DIMENSIONS.

GloVe dimension	Accuracy
25	86%
50	86.2%
100	86.2%
200	86.4%
300	86.6%

Table III reveals that by increasing the dimensions of feature vector, more parameters need to be learned, which results in getting more powerful and accurate learning architecture for sentiment analysis. Finally, we can conclude that our final LSTM model over 300 dimensional GloVe embedding outperforms baseline logistic regression and support vector machine by 8.2% and 8.8%, respectively on the crowdAI test set.

VII. CONCLUSION

In this project, we tackled a binary classification problem to classify the sentiment of Twitter messages. We extracted the useful information out of the given Twitter messages by removing neutral and meaningless words. Then, we explored two variant of feature representation methods namely tf-idf and word embedding. At the end, we built a few deep learning architectures and compared their accuracy with two baseline methods. The results showed that using long short-term memory (LSTM) recurrent neural network outperformed in comparison with the other models with an accuracy of 86.6%.

REFERENCES

[1] L. Wang, J. Niu, and S. Yu, "Sentidiff: Combining textual information and sentiment diffusion patterns for twitter sentiment analysis," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2019.

[2] R. Othman, Y. Abdelsadek, K. Chelghoum, I. Kacem, and R. Faiz, "Improving sentiment analysis in twitter using sentiment specific word embeddings," in *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 2, Sep. 2019, pp. 854–858.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, p. 436–444, 2015.