Tweet Sentiment Classification

Saibo Geng, David Yeung, Sahil Shah
EPFL, Switzerland

***Abstract*—In this report, we present a comprehensive study of sentiment classification on Twitter data, where the task is to predict whether a positive and negative smiley used to present in the Tweets, given the Tweet message. Several pre-processing techniques and their adaptation on the text data is discussed. The feature generation methods for machine learning sentiment classification models are implemented and compared. Various machine learning and deep learning model are implemented and their performance and features is also discussed. The results show that the deep learning method based on LSTM (87% accuracy+87% F1) outperforms the other machine learning models. In addition, we present an ensemble method that combines the logistic regression and Recurrent Neural Network, which proved to produce more accurate predictions(87% accuracy+87% F1)**

***Keywords*-Sentiment Analysis, Tweet Classification, Word2Vec, Glove, Machine Learning, Universal Sentence Encoder, LSTM, Ensemble Learning**

I. Introduction

Sentiment analysis is an area of research that try to identify emotions within text using techniques from natural language processing. Twitter the most popular micro-blog platforms that produce millions of short messages (280 characters max) everyday. The popularity of Twitter makes it a barometer of public opinion towards different matters: products, events, politics. Thus Tweet Sentiment Analysis has been widely used in areas like customer service, marketing etc.

In this report, we present a step-by-step approach for two main methods of sentiment analysis: machine learning and deep learning [SPW+13]. From the machine learning perspective, this problem can be formally transformed into a binary classification problem given a number of features of each tweet. The output is of two categories: 1 and -1, representing positive smiley and negative smiley. In this project, several commonly used machine learning and deep learning models as well as word/sentence embedding techniques will be built and compared. Different preprocessing and parameter tuning will be tested in order to improvement the performance of our classifier. While accuracy is the most direct indicator of whether a classifier is successful or not, other model evaluation criteria such as training time and model building complexity will also be taken into account to compare different models.

II. Tweet data set

We have three tweet data sets provided by EPFL CS433, respectively:

- positive training data: 1.25M tweets message which used to contain a positive smiley ':)'
- negative training data: 1.25M tweets message which used to contain a negative smiley ':('
- sample positive training data: 0.1M tweets message which used to contain a positive smiley ':)'
- sample negative training data: 0.1M tweets message which used to contain a negative smiley ':('
- test tweets data: 10K tweet message which used to contain a positive or a negative smiley

We have collected from internet an extra tweet sentiment analysis data set–Sentiment 140:

- positive training data:0.8M tweets message judged with positive sentiment
- negative training data:0.8M tweets message judged with negative sentiment

III. Data Pre-processing

Tweet messages are highly different from general text such as books or articles. Concretely, the following characteristics can be observed from tweet messages:

1) **Short Length**. Twitter limited Tweet length to 280 characters since October 2018. The average length in our data set:
2) **Highly Colloquial Expressions**, such as : 'y'all'
3) **Tweeter specific extra info**, such as 'url', hashtags, user referencing in '<>'
4) **Abbreviations**, such as 'lol', 'lmao' and 'ty'
5) **Informal or Incorrect Syntax Usage**, such as not spacing out words, extra duplicated characters like 'hiiiiii' or 'thankssss'

Based on these characteristics, implementing some data pre-processing before the feature extraction can potentially produce higher quality data, reduce the computational complexity and remove irrelevant information. The following pre-processing methods are investigated using untuned logistic regression with 1-gram TF-IDF :

Stop-words Removal :

Stop words are words which perform a connecting function in a sentence and do not carry a determinant effect on the meaning. These are in general words having a high frequency of occurrence in a text. Remove stop words by using pre-compiled stop word lists or more sophisticated methods are common techniques to reduce the noise on the text without affecting the content. Some research observations claimed stop word removal in tweet analysis has negative effect on the classification accuracy [SFHA14]. To verify the effect of stop-word removal, the accuracy variation before and after removing stop-words is compared in table I:

stop word removal	Accuracy	F1 Score
No	0.788	0.788
Yes	0.773	0.774

Table I: Effect of stop words removal

Stemming / Lemmatisation :

Stemming or Lemmatisation are conventional text pre-processing procedures, which replaces words with their stems or roots. Words, such as 'read', 'reader' and 'reading' are mapped into root word 'read'. However, they can also introduce new biases as 'read' and 'reader' carries different contextual meaning. Several research have claimed that stemming is more useful in morphological rich languages (e.g. German, Finnish), but not as useful in English. Also, under the length limit of the Twitter, short and more colloquial words are used instead of longer and more literary expressions.

Stemming	Accuracy	F1 Score
No	0.788	0.788
Yes	0.776	0.773

Table II: Effect of stemming (snowball)

Negation Processing :

In order to fully utilize the semantic sense of negation words in the tweets, an unconventional pre-processing step can be adopted as follows [PLV02]: first transform contraction form of negation into standard one, such as "can't" into 'can not', then add the tag **NOT_** to the first word following the negation word, e.g. "I don't like hot dogs" will be transformed to 'I do **NOT_like** hot dogs'

Negation processing	Accuracy	F1 Score
No	0.788	0.788
Yes	0.790	0.792

Table III: Effect of negation processing

Replace Repeating Characters :

Repeating letters to stress emotion is a common phenomenon in tweets as mentioned above. regular expression based detector is used to remove repeating letters.

Negation processing	Accuracy	F1 Score
No	0.788	0.788
Yes	0.775	0.773

Table IV: Effect of replacing repeating characters

Tokenize Wrongly-joint Tokens :

Wrongly-joint Tokens are words which should be separated syntactically with white-space but are joined together. Examples found in the data set includes: 'happybirthdaynathan', 'littlethingscount'. A total of 304 such tokens in test data set, roughly representing 1% of tweets. These super long tokens usually appear only once in the whole lexicon, and will normally be ignored in the feature extraction part. To avoid discarding the information contained in such tokens, we adopt the Viterbi algorithm [VIT] to segment these long tokens, e.g. **'myhomeawayfromhom'** into **'my home away from ho m'**.

Auto-correction :

Typos are very common in tweets and can account for up to 32% of words(97140/305831 in a test-set of 10K tweets). A Python package *SymSpellCompound* is introduced which supports automatic spelling correction. Working examples of the auto-correction: **'thiink'–'think', 'happi'–'happy'**. However, additional biases is also introduced: **'hahahahha'–'yamaha cha';'lool'–'look'**

IV. Feature Generation

There are different ways to encode the word string into number representation, aka. embedding. It would be ideal if the feature representation can capture important characteristics of the original sentence. Different feature generation methods are tested and discussed in the following session.

Term Frequency (TF), Phrase Frequency / N-gram: Either each word (TF) or N consecutive words (phrase / n-gram) is represented as a one hot vector. The resulting sentence embedding is the sum of the one-hot vectors.

Term Frequency-Inverse Document Frequency(tf-idf): Words that occurs in a lot of the documents are likely to carry less classification power than the others. The TF (term / phrase frequency) matrix can be multiplied by the inverse of the document frequency of words to include this intuition.

Word Embedding: Word embedding does not simply consider the existence of words but also the context of the words. Similar words will have similar vector representation using proper word embedding. The resulting sentence vector is the combination of the word embedding. Commonly used word embedding library includes Glove, Word2Vec and Bert.

Sentence Embedding: The same training can be done to sentence level. The sentence embedding is trained together with word embedding and directly output the embedding for a document. The library tried is gensim.doc2vec and also google universal sentence encoder. (refers to google universal sentence encoder documentation on tfhub)

Comparison of feature generation methods

Simple method like **bag of words** does not give a decent result. A reasonable interpretation will be that they does not capture the semantics of the words. **Tf-idf** is able to reach over 85% accuracy using fine-tuned logistic regression.

On the other hand, word embedding (**word2vec**) dose not work as well as expected (80% with logistic regression) in this context

Doc2vec from gensim does not work well with logistic regression. However, the **Universal Sentence Encoder** from Google can give a 84% accuracy trained with Neural Network(will be discussed later).

V. Machine Learning Approaches

The most popular machine learning algorithms for text sentiment classification are Support Vector Machine (SVM), Naïve Bayes and Decision Trees. After feature generation step, the text is represented in a vector of certain dimension that can be used as training data for the algorithms. 7 supervised classification algorithms in Sickit-Learn package is tested with reduced TF-IDF feature (PCA is used to reduce the dimension of the sparse TF-IDF vector into 100 dimensions). The results in the table below are obtained with 4-fold cross validation.

The results indicates that the logistic regression and non-linear SVM give the best performance in terms of accuracy. But the training time of RBF SVM is also longest compared to other algorithms. Thus, logistic regression seem to be the most promising method among all these and the hyper-parameters can be fine-tuned to obtained better performance.

Algorithm	Baseline Accuracy	Training Time (s)
Nearest Neighbors(K=5)	0.67	744.8
Linear SVM	0.74	1146.4
RBF SVM	**0.78**	**6510,9**
Decision Tree	0.681	7.3
Random Forest	0.631	0.7
Naive Bayes	0.719	210.2
Logistic Regression	**0.802**	**491.7**

Table V: Accuracy and training time of different ML algorithms

Tuning Logistic Regression: The Accuracy of is at its peak when 4-gram is used. However, the relative improvement from 3-gram to 4-gram is not very significant (only 0.1%) but the pre-computation time for feature generation as well as the feature dimension is significantly increased. Therefore, the more balance TF-IDF matrix will be the one using 3-gram.

VI. Neural Network

A. Dense Feed-forward Neural Network

A 4-layer feed-forward network is built using the keras framework. Two different set of input has been tested (TF-IDF / universal sentence encoder) and the accuracy and training time is observed.

Input features	Accuracy	Training Time (per epoch, average in s)
TF-IDF	81.9%	5373
universal sentence encoder (512-dim)	84%	207

Table VI: Comparison using neural network

B. LSTM

LSTM was proved to be very successful to address various tasks in NLP as it takes into account the time order of words. Here we implement a five layer Long short-term memory network. We tried different configurations in order to fine tune the model:

- **Add one drop-out layer** We obersve that drop-out layer does put off overfitting but also leads to a decrease of 0.1% on accuracy

- **Add one batch normalization layer** The batch normalization brings 0.1% of improvement in accuracy
- **Change different optimizer** Adam is slightly faster than the other common options
- **Effect of extra dataset** The result confirms that enhancing the training dataset helps increase the accuracy of classification. Our extra dataset is Sentiment 140, which was not built on the exactly the same criterion than original training data. But the classification accuracy does improve after including the extra dataset. Our explanation is: 1. two dataset were built on similar frame even though not identical 2. Slight difference between the tweets in two dataset creates a 'noise' which helps to prevent overfitting.

C. BERT

As the state-of-the-art in NLP domaine, BERT is a method of pre-training language representation.BERT was first pre-trained on a large corpus (Wikipedia + BookCorpus) for considerable number iterations (1M). One can build their models directly upon the pre-trained model and simply fine-tune the model to be adapted to the individual case.

D. Learning and Self-Correction of NN

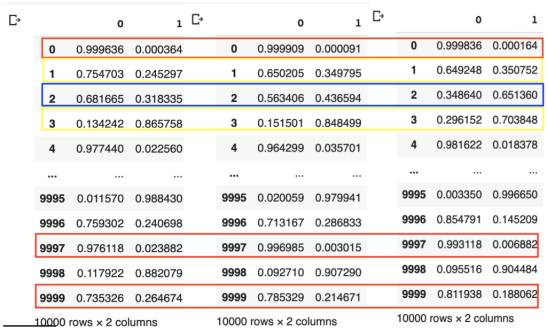

Figure 1: NN's confidence evolution

In the figure above, the three DataFrames are the results after 1,2,3 epochs. Each row represents a test sample. We can observe that NeuroNetwork becomes more and more confident about their classification over samples in red(the probability gap between 0 and 1 increases), but less and less confident over samples in yellow(the probability gap decreases) and finally, it may reverse(or correct) its previous judgement(samples in blue). The red samples helps to reduce loss but doesn't contribute to the accuracy. Blue samples are the origin of performance improvement after each epoch.

VII. Final Model

. Our results for the best 3 classifiers are presented in Table. The logistic Regression was based on 3 full gram TF-IDF with 100 iter. 5-layer LSTM was trained for 3 epocs and we got the best result after 2 epochs. 7

Algorithm	Final Accuracy	Training Time
Logistic Regression	0.868	**48mins**
5-layer LSTM	0.87	6h(GPU)
BERT	**0.885**	6h40mins(GPU)

Table VII: Accuracy and training time of different ML algorithms

Our final model is achieved with BERT training on sample training data

Auc	Accuracy	F1 Score	Precision	Recall
0.897	0.897	0.897	0.891	0.900

Table VIII: Performace of BERT on cross validation

VIII. Discussion of Results

As our data is Twitter posts, the sentence and the style of the data is quite loosely formatted and often unstructured. But as our experiment results suggest, one should be very careful while doing data preprocessing, because a preprocessing could easily be too aggressive and makes the classifier perform worse. In our LSTM and BERT models, we only proceeded the most basic preprocessing such as tokenization so that we could keep as much information as possible.

It's difficult to conclude which classifier is the absolute best.The best accuracy and F1 score were obtained with BERT, but the training time was considerably long. Logistic Regression could achieve our third best result and the training time is much shorter. However, fine-tuning the hyper-parameters of logistic regression(n-gram, regularisation,TF-IDF vocabulary size etc.) is quite time-consuming and the the time accumulated may be longer than the training time of BERT.

Limit of our model: We checked the False Positive and False Negative of our BERT classifier, 2 False Positive examples:

1) so justin is in london and i am off to school
2) guess who texted me again and wants us back

The above instance suggests our classifier perform doesn't well when the sentence contains a hidden emotion. This is the subtlety of human emotion.

References

[PLV02] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. *EMNLP*, 10, 06 2002.

[SFHA14] Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. On stopwords, filtering and data sparsity for sentiment analysis of twitter. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 810–817, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).

[SPW+13] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. 2013.

[VIT]