

Project 2 on Machine Learning

Text classification

Team Yoor

Sergei Volodin¹, Omar Mehio¹, and Baran Nama¹

¹EPFL

{sergei.volodin,omar.mehio,baran.nama}@epfl.ch

Abstract—A classification dataset consisting of Tweets is being studied. First, the data is thoroughly explored using visual aids. Several basic Natural Language Processing methods are applied. Results are evaluated using cross-validation. Model overview is given and the best model is chosen.

I. INTRODUCTION

This paper investigates into improving the quality of sentiment analysis on Tweets dataset [7]. It consists of $N_1 = N_2 = 1250000$ positive/negative tweets, each of them representing a message in English and numerical alphabet Σ with no longer than 140 characters. This way, each of $N = N_1 + N_2 = 2500000$ tweets is assigned to one of the classes $\mathcal{C} = \{:(, :)\}$. The task is to minimize the classification error. In other words, if $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ is the dataset with tweets $x_i \in \Sigma^*$ being messages and $y_n \in \mathcal{C}$ being class labels, the goal is to train a classifier $f: \Sigma^* \rightarrow \mathcal{C}$ which minimizes the loss function $l(y, \hat{y}) = [y \neq \hat{y}]$.

The task of sentiment analysis of tweets was thoroughly studied [2], [3], [4], [5]. These solutions usually give accuracy around 80%. Several techniques were applied, mostly consisting of two steps. First, the words are converted to dense vectors using Glove, word2vec, cbow or skip-gram models. After that, the resulting word vectors are used to construct features for the whole tweet. At the end, the vector is feeded into a classifier, such as SVM or Logistic Regression. Two latter steps might be replaced with a neural network accepting variable-length input such as RNN or CNN. Moreover, the embeddings themselves might be trained using backpropagation while training the classifier.

Claim: it is possible to find a model which fits the data better than the current state-of-the-art using expert knowledge on the Tweets dataset.

Next sections describe in details our approaches and compare them to various baselines.

II. MODELS AND METHODS

This paragraph describes the data being studied. Tweets are short messages no longer than 140 characters long [1]. The table I represents a few examples from the small dataset. Being considered an informal way of communication, tweets

often contain misspelled words and letter repetitions, grammatical and other writing mistakes. Besides plain text with punctuation, tweets also contain hashtags, two types of tags, *user* and *url*, which are tokens for replaced user mentions and URL links, respectively. In addition, tweets sometimes contain emoticons. Despite the fact that objects in the training dataset mostly comply with its classes, some tweets cannot be determined as positive/negative even by a human (appear neutral). Moreover, some of the tweets are clearly mislabeled in the training data. Dataset contains repeating tweets.

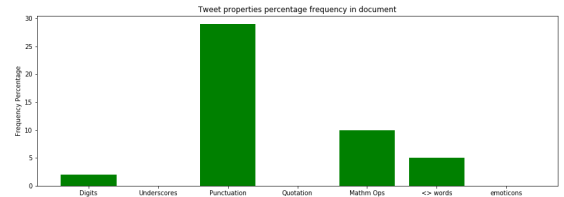


Figure 1. Distribution of Tweet properties

The first step taken was exploring special properties of tweets in the datasets. These properties include hashtags, punctuation ... and are present in huge percentages (Fig[1]). Through preprocessing the aim was to maximise the knowledge that can be extracted from the datasets in order to build word representations, which is achieved through correction of misspelled words and stemming techniques.

Usually tweeters repeat a letter in a word to stress on a certain emotion, as an example: "i am soooooo angryyyy". This problem was tackled by two different approaches. The first approach is stemming, which is defined as the process by which every word is replaced by its origin. Linguistically any word can be changed to different forms by either adding affixes or prefixes. Through stemming, the aim is to minimise the presence of these different forms and replacing them with the original stem. The stemming algorithm presented in 1 solves the mentioned problem.

The second approach adopted is misspelling correction.

Class	Row	Message	Comment
:(171	<user> 5k i could cry i'm so unfit !	User tag
:(99804	if i feel like this tomorrow i'm going to the er	Contraction, missed comma
:)	524	its the weekend ! ! <user> s coming home andddd <user> s baby shower , excitedddd	Letter repetitions
:)	99615	<user> aren't you just an adorable granny <url>	URL tag
:)	443	<user> :d :d :d :d :d wish jocelyn had a twitter . #kudos for her too .	Hashtag, emoticons
:)	468	retweet if you was born in the 90 ' s ! #90's babies	Grammar mistake
:(14	<user> i'm white . #aw	Appear neutral to human
:(99594	<user> <user> <user> they're there tonight ! ! !	Does appear positive or negative to human

Table I
EXAMPLES OF TWEETS IN THE SMALL DATASET

Algorithm 1 Stemming(token)

```

if token in dictionary then
    return token
else
     $X \leftarrow$  token with repeating character replaced by two
    instances of that character
    if  $X ==$  token then
        return token
    else
        return Stemming( $X$ )
    end if
end if

```

The process goes as follows : a token in the tweet is proposed to an english dictionary, the dictionary responds by either indicating that the word is valid or not. This dictionary provides the most relevant suggestions to correct a wrong token. The levenshtein distance was used to determine wether a word should be converted to the suggestion or not. Levenshtein distance is defined as : "the distance between two strings is the minimal number of insertions, deletions, and substitutions of one character for another that will transform one string into the other". Thus to explore how many misspelled words are in the dataset, we plotted the distribution in Fig [2]. Most of the data can be covered by setting our levenshtein threshold to 4.

The metric used to compare both methods was prediction accuracy of logistic regression. As a first step the combined dataset of both the positive and negative tweets was exposed to stemming and correction. Then for each method a vocabulary of words is constructed. Finally, the occurrence matrix is built from both the tweets and the dictionary. K-fold is used to split the matrix and then logistic regression is run using the occurrence matrix as the data matrix, labels

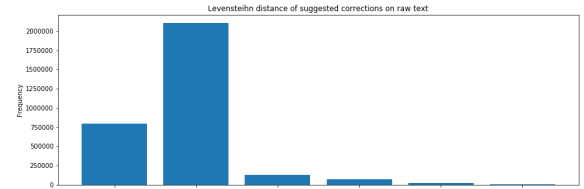


Figure 2. Levenshtein distance distribution on raw tweets

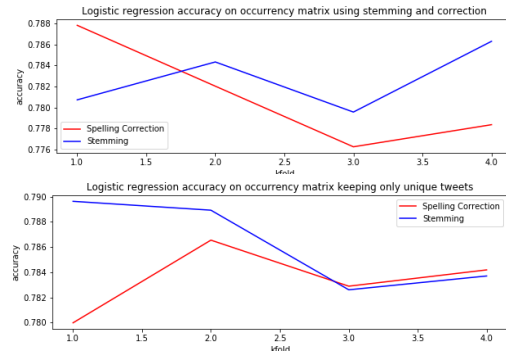


Figure 3. Logistic Regression accuracy on correct and stemmed tweets

are split into 0 denoting positive sentiment and 1 denoting a negative sentiment. As observed in Fig[3], the spelling correction method is superficial to the stemming method with reaching a maximum of 78.8 % prediction accuracy.

Another property of the dataset is the presence of redundant tweets. To view their impact, repeated instances of a tweet are eliminated and the newly formed unique dataset is exposed to the previous methods. The stemming technique overpowers the misspelling correction technique. Although with very small improvement in accuracy prediction (79%).

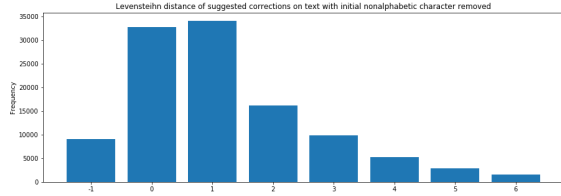


Figure 4. Levenshtein distance distribution on tweets with non-alphabetic starting character stripped

Tokens in our datasets starting with non alphabetic characters were eliminated from previous experiments as all dictionaries can't recognize non alphanumeric words. Thus a new set of tweets was constructed having all these tokens stripped of the initial nonalphabetic character or characters. As before, the distribution of levenshtein distances (Fig[4]) for suggested words was plotted, which results in a more uniform distribution.

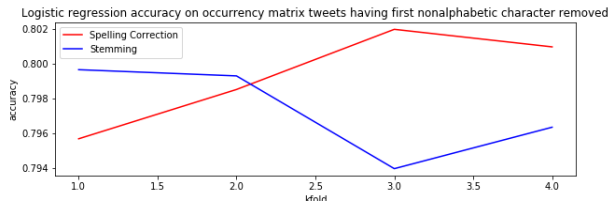


Figure 5. Logistic Regression accuracy on tweets with non-alphabetic starting character stripped

The newly constructed dataset is subject to previous methods. Elimination of redundant tweets, stemming and misspelling correction. Accuracy prediction then improves to 80.2 %, the best score reached so far.

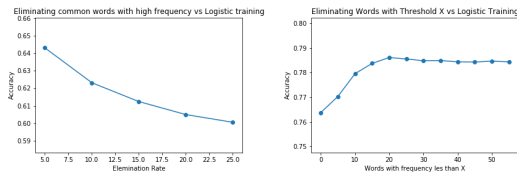


Figure 6. Elimination of common frequent terms vs separate nonfrequent terms

Now with the optimal model having initial nonalphabetic characters removed, redundant tweets eliminated and misspelling correction applied with levenshtein threshold set to 5. The dataset is tackled from a different angle, the effect of the frequency of which one token appears in the document is explored. Thus two approaches are considered: removing most frequent tokens that are common in both datasets and removing nonfrequent words from the combined vocabulary

set. The results of the first experiment came as a surprise, since we suspected that elimination of common words would make it easier for the classifier to distinguish between a positive sentiment tweet and a negative one. A reasoning contributed for such results is the limited number of characters provided for users to express themselves and thus most of the tweet content gets distributed among tagging other users, using punctuation and stopwords. The second approach was of more relevance, where elimination of non frequent tokens from the vocabulary has helped improving accuracy to 78 %, as observed in Fig[6].

A several models were considered. First, the baseline was the GloVe [6] model, which considers training embeddings on the co-occurrence matrix. Unfortunately, the model did train quite slow on the dataset gave meaningless embeddings (closest words for a given word did not correspond to it in meaning). Therefore, it was not considered.

Secondly, a word count model was considered as a replacement. Here, a tweet is represented by a high-dimension sparse vector, each item corresponds to a number of occurrences of this word in the tweet. This vector was further classified using several techniques, such as SVM and Neural Network. This method gave higher accuracy and allowed for tweaking its parameters with relatively small training time (a Google Cloud instance with 8 CPU was used)

III. RESULTS

The following models were considered: *aba*, *caba*, the best one is *aba*. This (does not) correspond to already conducted experiments [99, 98, 97]. Our contribution consists of running *method* with *xxx* modified with *yyy* and this does (not) give an improvement of 0.01231%

IV. DISCUSSION

Our experiments lack *zzz*, which can be improved by doing also *ttt*

V. SUMMARY

We have shown that it is possible to predict tweets using *aba* better than state-of-the-art.

REFERENCES

- [1] Twitter
- [2] Go, Alec, Lei Huang, and Richa Bhayani. "Twitter sentiment analysis." *Entropy* 17 (2009): 252.
- [3] Kouloumpis, Efthymios, Theresa Wilson, and Johanna D. Moore. "Twitter sentiment analysis: The good the bad and the omg!" *ICWSM* 11.538-541 (2011): 164.
- [4] Tapan Sahni, Chinmay Chandak, Naveen Reddy, Manish Singh. "Efficient Twitter Sentiment Classification using Subjective Distant Supervision"
- [5] Alec Go, Richa Bhayani, Lei Huang. "Twitter Sentiment Classification using Distant Supervision"

[6] Jeffrey Pennington, Richard Socher, Christopher D. Manning.
"GloVe: Global Vectors for Word Representation"

[7] Competition and data downloads on [kaggle.com](https://www.kaggle.com)