Project 2

Bilguun Chinzorig, Monika Avila, Lkham Nyambuu
EPFL

***Abstract*—The aim of this project is to conduct a twitter sentiment analysis. Our goal is to classify untaged tweets into two groups: positive and negative. For this purpose, we begin our work by preprocessing the tweets and generate the dictiorany. Following, we create the word space, i.e. matrix of embeddings using GloVe methodology. After, we span the tweet space by retrieving features from the embedding matrix. In addition to these features, we constructed additional variables that capture the meaning and the importance of the words as well as hashtags included in each tweet. Finally, we used our futures in two types of classifiers: 1. Random Forests and 2. SVM. We conclude that random forests outperforms SVM considering both computing efficiency and classifying results.**

I. Introduction

II. The vocabulary construction: Word preprosecing

We begin our work by preprocessing the available tweets such that we can define our vocabulary. However the words in the dataset are not easily separable by whitespaces due to following reasons:

Separators: Words can be separated by multiple characters including whitespace for words, period for sentence ending, comma for clause endings, dash for connected words and : or ; to for beginning independent clauses. The problem is that the last two characters can be used as emojis.

Word contractions: word contractions can be viewed as complete new token, but in the end it is just combination of two words. Common word contractions are related to to be's and modal verbs.

Special words: Due to freedom of writing tweets, we can observe multiple emphasizes on words including hashtags, and repeated characters like (hey to heyyyy). These words must have a special treatment but for vocabulary building these variations were eliminated.

Stop words: The full list of stop words can be found here https://kb.yoast.com/kb/list-stop-words/. In our case, we are assuming pronouns like ”the” can represent some meaningful information since it emphasizes following nouns.

Word variations: In english word can take multiple forms like plural form, verb tenses, incorrect spelling etc. Hence, simple word separation is not enough. And also in english, people use ”’s” or ”s’” to represent possessions REPHRASE THIS!!

Numbers: we assumed that numbers usually conveys factual informations which is not helpful to identify the opinion of a person. Moreover, we need to treat numbers different from words. Thus, we have completely removed every numbers.

In order to overcome the problems mentioned above, we created a customized vocabulary building algorithm with the following procedure:

First we tokenized the text using white space, comma, period, new line. We excluded : and ; since they are maybe part of emojis. Following, for each token we look for word contractions. In our case, we only considered hashtags, common contraction list to separate tokens which contains multiple words. Moreover, we have removed possessions from each token. After this, we address the issue of word variation by shrinking consecutive repeated characters, e.g. we replaced ”fooooot” by ”foot”. Finally, we used their stems to build our vocabuary.

This preprocesing lead to an increase of the tokens in 80,000. Moreover, the total number of unique words in decreased significantly. Indeed, we have successfully obtained almost 32,000 unique words. This means that we ended up with just 32% of all the tokens produced after splitting. (Clarify)

Now let's look at the distribution of each words. The graph 1 shows the distribution of the words of our final vocabulary. As expected, it shows similar relationship as Zipf's law, but note that the number of words with only 1 occurence in the entire text is important. Indeed, almost 50% of the vocabulary takes only 1.2% of the text which implies that it is not worth keeping the 50% of the words for simplicity.

Finally, in order to increase efficiency of the information contained in our final co-ocurence matrix we apply term frequency- inverse document frequency (tf-idf) feature. This factor gives more importance to words that are repeated in a tweet but not highly recurrent in the whole corpora.

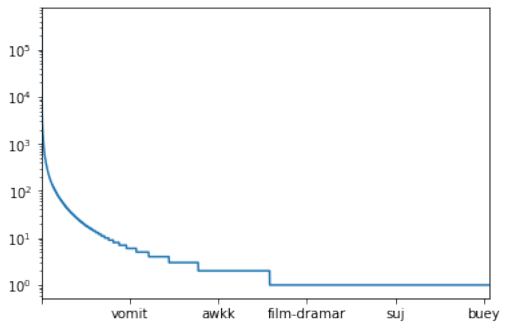

Fig. 1. Word Distribution

III. The word space: embedding matrix

In order to obtain a numerical representation of the words w_i contained in the data set as $\mathbf{w_i} \in \mathbb{R}^K$ we used GloVE model.

This model retrieves the numerical representation of the words that is closest to natural logarithm of the observed co-occurence value x_{ij}. Indeed, Pennington et al. (2014) propose to obtain the real valued vector by minimizing a weighted sum of squared errors. Where the error is the difference between the log of co-occurence value and the predicted one given by $\mathbf{w_i'w_j}$. Thus, the model minimizes the following cost function:

$$L(\mathbf{w_n}, \mathbf{w_c}) = \sum_{j=1}^{N} \sum_{i=1}^{D} f(x_{ij})(log(x_{ij}) - \mathbf{w_i'w_j})^2$$

with:

$$f(x_{ij}) = min(1, x_{ij}/x_{max})^{3/4}$$

IV. The tweet space: feature retrieval

In order to obtain a numerical representation of the tweet space we retrieved two types of features. The first class is obtained using GloVe matrix of embeddings. On contrast, we developed the second type of features with the aim to capture the syntax.

Using the GloVe matrix of embeddings, we used the mean of the word embedings.

On the other hand, the features that we developed are:

1) The mean of the tf-idf.
2) The weighted sum of the tf-idf, which means that we weighted hashtaged tf-idf, positive tags tf-idf, negative tags tf-idf and emphasis tf-idf.

V. The Classification Model

After obtaining the tweet space i.e. the features that represent the data set of tweets, we trained the classification model. For this aim, we used two models: 1. Random Forest and 2. SVM.

The first model is random forests, this is a method based in an ensemble approach composed of a set of decision trees. At each node, the variables used to divide the space of the independent variables are randomly selected and used to create the decision tree (Breiman (2001)).

The second classification algorithm used is SVM. In this method, we look to maximize the geometric margin between positive and negative tweets subject to the constraint that the classified vectors must lie outside this margin[1]. This optimization problem is equivalent to the following one for each training sample:

$$min_w ||w||^2$$
$$s.t. \quad y_i(x_i'w + b) \geq 1$$

Now, setting the dual problem we can obtain the support vectors which are few training observations that lie on the margin.

[1]AndrewNg, Support Vector Machines

VI. The Data

The training data set is formed by 2 millions of tweets, half of them correspond to the positive emotions and the other half to negative ones. This data has been already tokenized, thus we start with the cleaning and the analysis of the data.

VII. Results

Among the multiple tests with different parameters, we have included the best results First, Word embeddings were experimented with all four different options, baseline, trained, pretrained and merged as they are described in Section IV. In order to understand the performance of SVM classifier in this configuration, we also tested NN classifier for the same data representations. For this purpose, we first intended to find the best possible NN configuration for the resources we have, since more powerful architectures require more computational and memory resources. Both single hidden layer and two hidden layers architectures with different number of neurons are trained. Table I gives the architectures and corresponding abbreviations, which are used in Figure 2 to give a clear picture of performances for different GloVe embeddings and SVM-NN classifiers. As can be seen from the performance chart, NN outperformed SVM for all cases. In addition, the best performance is obtained with the largest single layer NN architecture, i.e. NN3, which was the biggest that our resources enabled us. Second hidden layer, i.e., more nonlinearity never achieved better. Regarding the GloVe dictionaries, pre-trained vocabulary always lead to better results compared to trained ones, which is expected due to the amount of data used for the pre-trained vocabulary. However, merging the new words contributes to performance by 1both NN an SVM classifiers. Only interesting thing to notice is 5https://www.tensorflow.org/ Figure 2: Performance of SVM/NN classifiers using GloVe representations obtained by a variety of dictionaries the fact that utilization of TF-IDF constants as weighting coefficients for GloVe representations does not improve the classification performance. In fact, we obtained 81:50with weighted representations of pretrained vocabulary and NN3 classifier, whereas the raw representations achieved 82:14Furthermore, we investigated the performance of n-grams representation for different window sizes, where it is applied to preprocessed tweets and SVM is utilized as the classifier. Results are summarized in Figure 3a with 4-grams to seem to be the best representation. (a) n-grams: Performance vs. n (b) FastText: Performance vs. D As another test, FastText algorithm is experimented for different dimensions of representations, results of which is given in Figure 3b. For all different dimensions, preprocessed dataset is utilized and the classifier is trained with a learning rate 0:05 by 50 epochs. Dimension of 200 seems to give the best accuracy. In order to further observe the contribution of preprocessing step, we also chose the best-performing n-gram and FastText configurations to apply them to raw data. results presented in Table II clearly indicates that proposed preprocessing step adds around 2Table II: Classification performance (Algorithm w/o preprocessing w/ preprocessing Improvement 4-grams 85.28 87.20 1.92 FastText, D=200 80.36 83.04 2.68 The last but not the least, we trained several CNN architectures, where we altered the capacity of the network in order see the limits of the algorithm regarding the training data. Best performance is achieved when we use a single hidden layer with 128 neurons, where 128 convolutional filters of 2,3,4,5,6 neighborhoods are used. Regarding the computational necessities, instead of crossvalidation, we kept randomly chosen 10K samples of training dataset for the validation for all training attempts. Figure 4 illustrates the evolution of accuracy during the training. Roughly speaking, after 18K steps, we suspected model to overfit due to the divergence of accuracy values for training and validation dataset, but as can be seen on the second figure, when we kept training with another randomly chosen validation data due to non-constant seeds, validation performance was better than the training. The main reason behind this difference is related to the fact that second validation set had been used for the training at the first stage. Since we didnt have enough time for a training from scratch, we stopped the training at around 30K global steps, which gave us 87:54error, which had been 87:48(a) (b) Figure 4: Evolution of accuracy values for training(orange, smoothed) and validation(blue) datasets for 4a first 18K steps and 4b remaining 12K steps When we increased or decreased the capacity of networks by increasing/decreasing the number of filters per neighborhood sizes as well as the number neighborhoods, or adding another hidden layer before soft-max classifier, performance always got worse. If we increased the capacity, decay in the performance is related to overfitting and if we decrease the capacity, deterioration can be due to insufficient representational power.

VIII. Discussion

IX. Summary

References

Breiman, L.
2001. Random forests. *Machine learning*, 45(1):5–32.

Pennington, J., R. Socher, and C. Manning
2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, Pp. 1532–1543.