Twitter Sentiment Analysis using Blending on various Embedding-Based Classifiers

Robert Dadashi-Tazehozi
rd2669
Department of Computer Science
Columbia University

Flavien Prost
fp2316
Department of Industrial Engineering
& Operations Research
Columbia University

Abstract

This paper introduces a new classifier for sentiment analysis on tweets. The complexity of tweets compared to other corpus relies in its short length and its spelling errors. The tweet dataset come from the Sentiment140 corpus and is composed of 1.6 million tweets automatically annotated using emoticons. Different embedding are used and compared in this report. We used both bag of word representation associated to feature selection and Doc2Vec which is based on Google recent research. Furthermore we came up with a new polarity grained based representation. We built various classifiers on these embeddings and compared their results. Eventually we combined them through ensemble learning methods. We reached a 80.1% accuracy.

1 Introduction

Twitter has been a center of interest for research over the past few years and one of the still open issue is the automatic sentiment extraction on tweets.
As Twitter is one of the most popular social media, such a tool offers a wide range of applications mostly in quantifying public opinion for politicians, events, brands, new products, ideas or sport teams.
Sentiment analysis has yield efforts in computational linguistics. But traditionally, the task is made on larger texts such as movie reviews ([1], [2]). Indeed the unconsistent nature of tweets is by itself a challenge for classification.

2 Data Processing

We used the Sentiment140 dataset for Twitter Sentiment Analysis [3]. The dataset includes 1,600,000 tweets in the training set and 498 tweets in the test set.
While the testing data was manually anotated, the labels on the training set were automatically assigned through emoticon analysis: a tweet containting ':)' was assigned to be positive. After that, the emoticons were stripped off the text of the tweet. This makes our work very different from the paper [3], which takes emoticons into account.

We basically followed the same preprocessing pipeline as in [3] so as to have comparable results. The tweets were cleaned off their unconsistencies:

- Letter repetition: *Huuungry, Huuuuuungry, Huuuuuuuungryyyyyyy*
- Hashtags
- URLs
- Usernames

3 Method: embeddings and classifiers

3.1 Bag of words Representation and feature Selection

This section introduces a traditional representation for text: Bag of words.
From the pre-processed training sentences, we

define a vocabulary V containing all the words. Then each sentence is represented by a vector of size $|V|$ and the dimension i represents the number of occurences of the word number i. Due to the size of the vocabulary (about 300,000 words), this representation suffers from a memory size problem. We indeed need to save a vector of size $|V|$ for each training sentence. Therefore we selected the features which carry the most information.

Indeed, feature selection plays an important role in text categorization. It is a good bet that many words carry low information, and will lead to bad performance when aggregated. The target is to remove the noisy data as much as decreasing the dimensionality.

Yan Xu et al. [4] analyze different ways to select the best features. They put emphasis on a document frequency threshold in order to consider only the frequent words. Then they study different selection criteria such as Information Gain or Information Theoretic Mutual Information.

We decided to use the Gini information criteria which gives a neasure of how common a word is in a positive tweet compared to a negative one. It can be easily computing by going through all the training tweets.

$G(w) = p_1(w) * (1 - p_1(w)) + p_2(w) * (1 - p_2(w))$

To make sure that we do not select any rare words, we add the constraint that the word must appear at least 40 times in the training set. We identified then the m words with the lowest Gini information that meet the constraint.

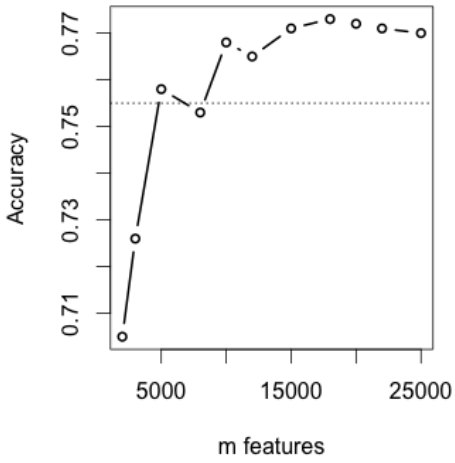

Figure 1: Bag of words representation depending on the number of best features: the dotted line is the proportion of tweets in the validation set which have no best features. The solid line is the number of tweets wich have 0 or 1 best feature

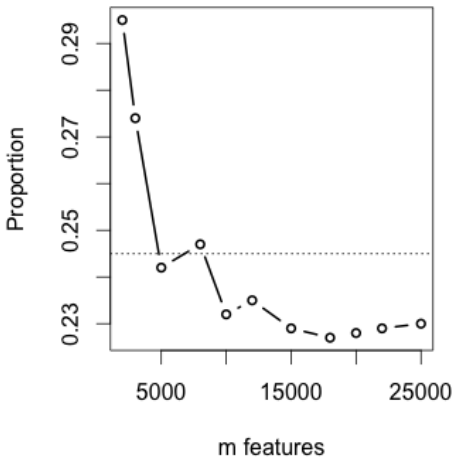

Figure 2: Naive Bayes classifier accuracy as a function of the number of best features used, the dotted line represents the accuracy when all the features are selected.

The figure 1 and 2 show that m has great influence on the results. We see in figure 2 that the naive Bayes classifier results with selected

features are above the naive Bayes classifier accuracy when trained on all the dataset, meaning that features selection makes sense. We can see that if it is not large enough the Bag of Word representation does not carry enough information, as it is a 0 vector for many tweets (Figure [**?**]). However, for memory size purpose, there is a trade-off between the number of sentences that we use to train our model and the size m of our vocabulary *V'*. This is why we chose the value of m when we trained the models (see next section).

These m words form our new dictionary *V'*.
After this analysis, our representation of the tweets is a bag ow words with the dictionary *V'*

3.2 Classifier using bag of words representation

Using the Bag of words representation where the best features have been selected, the "classic" classifiers can be used.

Our problem suffers the curse of dimensionality: we need to keep many features to represent the tweets correctly and therefore can not use many training sentences. To tackle the risk of overfitting, we came up with two different kind of models:

First, we trained a logistic regression, which is a simple but usually efficient classifier We add a regularization term (using $|| \, . \, ||_2$).

- Decision Tree Based models. This seems quite intuitive in this problem.
To control overfitting, we used the Random Forest Classifier. Indeed, by gererating decorrelated trees and computing the average prediction, the random forest perfects very well and often outperforms logistic regression.
We also trained an Adaboost classifier. This latter recursively improves its predictions by looking at its error.

We trained the model on a part ofthe training set and used the validation set to select the parameters. This implies choosing for each classifier the number of features ,m, the number of training sentences we use, n, and the parameter of the classifier itself, such as the number of estimator for Adaboost.

Table 1: Test errors of different classifiers

Classifier	Test error
Logistic Regression	79,2
Random Forest	75,5
Adaboost	82,9

3.3 A topic-based representation: Doc2Vec

Doc2vec was developed by Google and is inspired from their previous word on Word2Vec [5]. Word2Vec [5] is a topic-based representation and aims at matching a word with a vector. This vector is a way to estimate the distribution of the neighboring words:

$$p(w_{i+1}|w_i) = \frac{exp(v_{w_i}.v^T_{w_i+1})}{\sum'_w exp(v_{w'}.v^T_{w_i})} \text{ ,where}$$

v_w is the representation of a word

The representations of the words, as well as their size, are trained to maximize the log-likelihood on a training corpus.
This method has reached impressive results as the representations can be easily combined: by computing $Queen + man - woman$, we are close to $King$! Doc2Vec relies on this method and matches a vector to a whole sentence.

Topic-based representations are adapted for text classification and have shown efficiency ([6], [7], [8], [9]). These representations have also been used for sentiment analysis, reaching 91% accuracy using the Doc2Vec representation on the IMDB reviews dataset [9].
On the erratic twitter dataset, we computed a doc2vec representation using [10] with different number of dimensions (50, 100, 400) of the representation.
The results for several classifiers (summarized in Table 2) are way below the ones expected. We explained this gap with the fact that tweets are shorter then movie reviews, more erratic and simple. Thus a context for a positive word can be extremely close to a context for a negative word (as for *love* and *hate* in *'I love you'* against *'I hate you'*).

Table 2: Accuracy on testing data using 400 dimensions in doc2vec. The parameters of the classifiers were chosen using cross validation.

Method	Accuracy
SVM	60.0
Logistic Regression	65.1
Adaboost	53.4
Random forest	55.7

3.4 A new polarity-grained representation

In bag of words and doc2vec representations, the order of the words does not matter. We had the intuition that it represents a substancial loss of information as in the following examples: *make love not war* and *make war not love*.
We came up with a new representation for the sentences to catch the sentence order. As the tweets are in average composed of 7 words, we decided to represent each tweet as the 20 first words or as the tweet completed with 'null' words. This 20-word based representation has proven to be more efficient then a string representation having the advantage of being independant on the string size on which we applied String Subsequence Kernels [11].
To prevent dimension explosion, each word is represented as its global polarity on the dataset in a range of 1 to 5 (0 being for the null words), which was inspired inspired by [1] where the grained polarity is used on sentences. Indeed, words only associated with positive sentences will get a 5 polarity whereas words appearing equally in each kind of sentences will get a 3 polarity.
For the previous examples, *make love not war* and *make war not love* would be represented as $[3, 5, 2, 1, 0, .., 0]$ and $[3, 1, 2, 5, 0, .., 0]$ respectively.

Again from this representation, we built a set of classifiers with results detailed in Table 3. One interesting thing is that we get similar result using 10%, 50% and 100% of the dataset.

Table 3: Accuracy of different classifiers on the polarity grained representation, with different amount of data. For each classifiers, the hyperparameters were optimized using cross-validation.

	10 %	50%	100%
Logistic Regression	66.1	66.4	66.4
SVM-RBF	75.0	75.1	75.0
KNN	74.5	74.5	74.7
1-Hidden-Layer MLP	73.0	74.1	74.4

We represent the accuracy of the support vector machine using radius basis function with C = 0.01 and gamma = 0.01 as a function of the data fraction used to train the classifier in figure 3. The figure that the performance of the classifier converges fast with the fraction of the dataset taken to build the classifier.

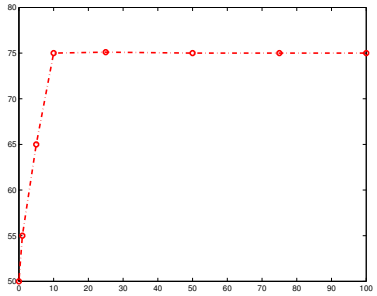

Figure 3: Accuracy of the support vector machine using radius basis function with C = 0.01 and gamma = 0.01 as a function of the data fraction taken to build the classifier

4 Emsemble Learning

By analyzing our experiments, we found out that some of our models performed well in different situations. For instance, logistic regression performs very well but faces the problem of empty bag of word representations after feature selection. On the other hand, the polarity-grained based classifiers do not suffer this problem. This led ud to look over Ensemble Learning Methods for Classifiers.

Dietterich [12] introduces ensemble methods. Compared to the standard supervised Learning problem, these methods combine the individual decision of a set of classifiers. To expect to get a higher accuracy, two main criteria are necessary: the classifiers need to be diverse and accurate.
As most of our classifiers are based on different models and different representations, we believe that they meet both criteria.

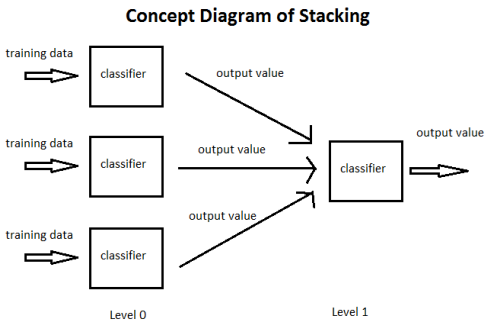

Figure 4: Principle of ensemble methods

Ting et al. [13] detail then the stacking algorithm in two different levels. The level-0 consists in fitting different classifiers on the training data.

This classifiers can be any standard classifier and it is common to diversify them by using different training set and parameters. The level-1 takes the output predictions of level-0 as features for a new classifier. The common methods used for level-1 are Majority-Voting, Bagging and Boosting.

Comparing different ensemble learning methods, we decided to use a blending method. It is a simple and intuitive way to combine different models and is commonly used when the models are fewer and more complex than for Bagging and Boosting.

For the level-0, we used the following classifiers: Naive-Bayes, Polarity-Grained, Random-Forest, Adaboost and Lotistic Regression.

For the level-1, we used a Logistic Regression and will compare it to the Majority voting result. In a first approach, in the level-1 we decided to use the predicted class instead of using the probabilities associated to each prediction. This decision was motivated by the discriminative classifiers we used on the polarity-grained representation and do not return any probability associated with the predictions.

We have then analyzed the evolution of the classification error depending on which classifiers from level-0 that we considered as a feature for the level-1.

Table 4: Classifier of level-0 and their number

Classifiers used in Level-0	Number
Naive Bayes	1
BOW + Logistic regression	2
BOW + Random forest	3
P-grained + SVM	4
P-grained + logistic regression	5
P-grained + knn	6

Table 5: Blending Method: Test error

Classifiers used	Maj Voting	Blending
1-2-3-4-5-6	65,7	63,8
1-2-3-4-6	65,7	67,7
1-2-4	63,8	58,9
1-2-3-4	61,9	80,4

This tables show the results of the blending method. The best predictions are reached by a combination of four different classifiers: Nave Bayes, Logistic regression and Random forest with a Bag of word representation and a SVM with polarity grained representation. All these reprentations performed well independantly and are quite diversified.

5 Results and Discussion

Our approach has revealed itself powerful to reach step by step an accurate model. Our preprocessing was different from [3], where the authors used emoticons in their prediction models. These emoticons have strong prediction power as Naive Bayes classifier highlighted: on their dataset, this classifier on unigrams reached 81.3% whereas ours reached 75.5%. By using diversified and innovative embedding and fitting various classifiers, we managed to reach a predictive accuracy of 79%. Our last step, which consists in blending previous classifiers, eventually gave us a 80.4% accuracy.

These results are satisfying and their impact could be even more comprehensive by adding a new class containing the neutral tweets.

References

[1] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[2] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[3] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12, 2009.

[4] Yan Xu, Gareth JF Jones, JinTao Li, Bin Wang, and ChunMing Sun. A study on mutual information-based feature selection for text categorization. *Journal of Computational Information Systems*, 3(3):1007–1012, 2007.

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[6] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

[7] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134, 2000.

[8] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[10] 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`.

[11] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.

[12] Thomas G Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.

[13] Kai Ming Ting and Ian H Witten. Stacked generalization: when does it work? 1997.