The SentiME System at the SSA Challenge

Efstratios Sygkounas[1], Xianglei Li[1], Giuseppe Rizzo[2], Raphaël Troncy[1]

[1] EURECOM, Sophia Antipolis, France,
{efstratios.sygkounas,raphael.troncy}@eurecom.fr
[2] ISMB, Turin, Italy,
giuseppe.rizzo@ismb.it

Abstract. We describe SentiME, an ensemble system composed of 5 state-of-the-art sentiment classifiers that have proved to perform well on short-texts. SentiME first trains the different classifiers using the Bootstrap Aggregating Algorithm. The generated models are used by the classifiers separately, while the results are then aggregated using a linear function averaging the different classification distributions. SentiME has been initially tested over the SemEval2015 test set, properly trained with the SemEval2015 train test. It would outperform the best ranked system of the challenge. It has also showed robust performance over the SSA Task 1 training set in a 4 cross-fold experimental setup.

1 Introduction

Nowadays, people frequently purchase goods on the Web. One of the most popular web site where goods are bought is Amazon. After a purchase, the buyer is generally invited to publish a review about the product. These reviews are useful for future customers that seek opinions and sentiments to support them in their decision buying process. The first task of the ESWC 2016 Fine-Grained Sentiment Analysis challenge[3] is about binary polarity detection of customer Amazon reviews. A participant system should take as input an XML document containing the textual comment enclosed in the textual tag, and should generate another XML document that adds, for each sentence, the result of the classification enclosed in a polarity tag. Precision, recall and F-measure of the detected polarity values (positive or negative) are the metrics used for evaluating the participant system on each review of the evaluation dataset. Table 1 reports some statistics about the dataset provided by the challenge organizers.

In this paper, we describe SentiME, a system that implements an ensemble of five state-of-the-art sentiment classifiers. In the reminder of the paper, we present the SentiME system workflow and the experimental settings. We then conclude with some lessons learned.

2 The SentiME System

SentiME is an ensemble system which is inspired by and built upon the Webis system [2]. We extend it using the Boostrap Aggregating Algorithm [1] (referred as bag-

[3] https://github.com/diegoref/SSA2016/wiki#task-1-polarity-detection

	Number of categories	Number of sentences	Average number of characters per sentence	Average number of word per sentence
positive	20	500000	466	86,04
negative	20	500000	513	94,75

Table 1. Some statistics of the SSA Task 1 dataset

ging) for training, and adding a fifth classifier, namely the Stanford Sentiment System [4]. SentiME first trains the 4 classifiers of the Webis system using the Bootstrap Aggregating Algorithm that is an ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid over-fitting. To perform bagging, the first step is to generate new training set for each of the 4 classifiers using uniform sampling with replacement. We then add the results of the Stanford Sentiment System, used as an off-the-shelf classifier to the model.

The final step of the approach is to aggregate the individual results from each sub-classifier. When we aggregate the classification results of five sub-classifiers, we use a linear function which averages their classification distributions and classifies the sentence according to the label that holds the maximum value in the average classification distribution. Figure 1 summarizes the workflow used in our approach.

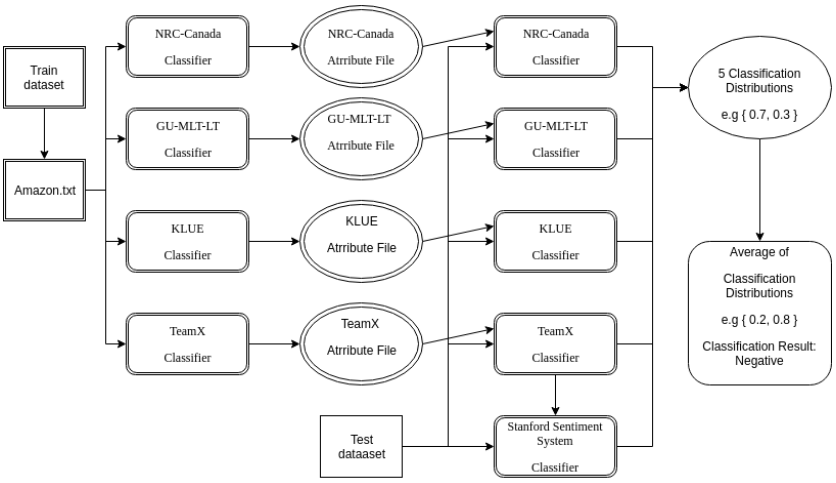

Fig. 1. Diagram of the SentiME System workflow.

2.1 Training

The training workflow consists of four steps: *i)* parse the training set that is released in XML format; *ii)* extract all the information needed such as **sentence id**, **textual**

content, and **polarity**; *iii)* compute the features and create a TSV file and, finally, *iv)* launch the bagging process. The output of the bagging will generate different models that are used as input to train separately each of the four classifiers.

The random selection process generates $(1 - \frac{1}{e})$ unique sentences, while the remaining ones are duplicated sentences. Figure 2 illustrates how the bagging process is implemented and how it affects the learning of all sub-classifiers.

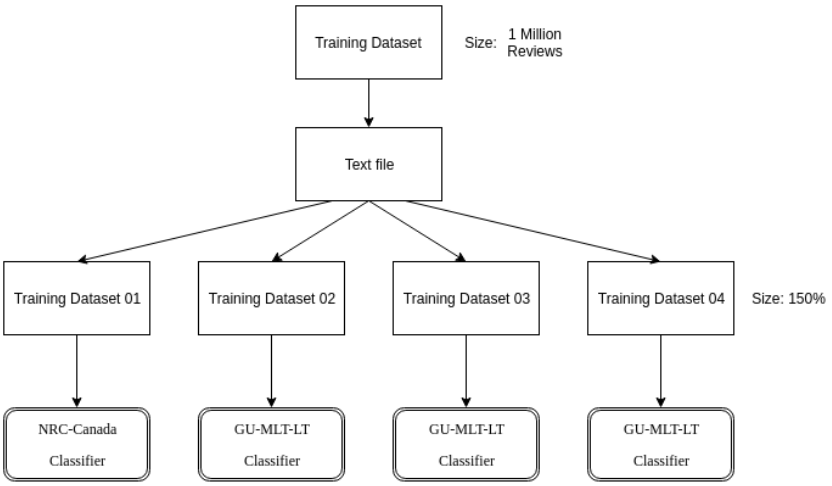

Fig. 2. Our Bootstrap Aggregating Implementation.

As shown in Figure 2, we generate a model for each of the four trained sub-classifiers. The classifier outputs are aggregated via a linear function. Due to the fact that the bagging introduces some randomness into the training process and the size of bootstrap samples is not fixed, we decide to perform multiple experiments with different size repeated multiple times for each size. From the experimental results, we have observed a convergence of the performance and thus, we can conservatively claim that a random generation is representative enough to carry out this task. A comprehensive list of features used by the sub-classifiers is shown in Table 2.

We used a variety of polarity dictionaries since we have observed numerous emoticons such as *:)* and *:-(* in the XML files of the training set. We use emoticons dictionaries to categorize a sentence according to the presence of an emoticon. An additional point worth mentioning is that punctuation marks are also a common way to express a sentiment. Consequently, we consider punctuation marks dictionaries. Finally, we also make use of POS tags either provided by the Stanford POS tagger, which is for formal language, or by the CMU ARK POS tagger, which is used for more informal language.

The fifth sub-classifier of our approach, namely the Stanford Sentiment System, is used as an off-the-shelf tool. Consequently, it is not specifically trained using the Amazon reviews dataset but instead, makes use of the model developed in [4].

Approach	Machine learning	Features
NRC-Canada	Support Vector Machine	n-grams, alcaps, POS, polarity dictionaries, punctuation marks, emoticons, word lengthening, clusters of words and negation
GU-MLT-LT	Linear Regression	normalized uni-grams, stems, clusters of words, and negation
KLUE	Maximum Entropy	unigrams, bigrams, and an extended unigram model of negations
TeamX	Logistic Regression	word n-grams, character n-grams, clusters of words, and word senses
Stanford Sentiment System	Recursive Neural Networks	-

Table 2. Machine learning algorithms and features being used by each sub-classifier

2.2 Testing

The testing process of our system contains the following steps. First, we load all the parameters into the feature extractor and the classifier. Then, we pre-process the sentences as what we do in the training process. In a next step, we extract the feature vectors from the cleansed sentence texts and pass them to the classifier. The classifier will give the classification result which will be used to generate the final classification. When we evaluate our ensemble system, we use the linear function that averages the classification distributions provided by the five sub-classifiers and produce the final classification according to the maximum value of the labels in the average classification distribution.

Concerning the Stanford Sentiment System, the main challenge in using a corpus not labeled at tree level is that the sentences often contains misspelling, poor grammatical structure, emoticons, acronyms, and slang which is quite out of the range of Stanford Sentiment System because it cannot robustly compute the tree structure. To reduce partially this difficulty, we perform a pre-processing of the sentences where we filter out all URLs, trim the sentences and lowercase all the capitalized characters. We give the output to the Stanford Tree Parser which is a machine-learning model that parses the input text into Stanford Tree format and this output is given to Stanford Sentiment Classifier which outputs the classification results for Stanford Trees. Stanford Sentiment Classifier provides us with a rich result format: classification label and classification distribution on all the nodes in the Stanford Tree. We only extract the root classification distribution because it represents the classification distribution of the entire sentence text. Finally, it produces the classification result which has been mapped in two labels (positive and negative).

3 Preliminary Results and Lessons Learned

Table 3 reports the F-measure performance of the 4-fold experiment we conducted on the training dataset. Due to scalability issue, we have further down sampled each fold using 100K reviews as training and 10K reviews as test.

Fold 1	Fold 2	Fold 3	Fold 4
0.9042	0.9065	0.9095	0.9113

Table 3. Classification results in a 4-fold cross validation experiment setup

The ensemble approach shows consistent results in the experimental setup and it results to be agnostic to textual variations as it is the case in the sentences collected from twenty different categories. The key-strength of the approach relies on the bagging process allowing stable results preserving the model to overfit. Being a data-driven approach, the variation of the training set makes the approach ready to process different text-genres such as microposts and reviews. Finally, the addition of the Stanford Sentiment System in our ensemble improves the performance on sarcasm sentences, that are quite common when conveying sentiments. This happens because the native Stanford Sentiment System has a great strength to classify sentences whose golden standard is negative as it has been tested in our previous experiments over the SemEval2015 test set [3, 5]. This means that we can use Stanford Sentiment System to help our system to classify the sarcasm sentences. On the other hand, Stanford Sentiment System is heavily skew towards negative. We address this problem in using the Bootstrap Aggregation Algorithm (bagging) with the size of 150% of the original training set.

Acknowledgments

The authors would like to thank Xianglei Li for his earlier work on the SentiME system. This work was partially supported by the innovation activity 3cixty (14523) of EIT Digital and by the European Union's H2020 Framework Programme via the FREME Project (644771).

References

1. Breiman, L.: Bagging Predictors. Machine Learning 24(2), 123–140 (1996)
2. Hagen, M., Potthast, M., Büchner, M., Stein, B.: Webis: An Ensemble for Twitter Sentiment Detection. In: International Workshop on Semantic Evaluation (SemEval) (2015)
3. Rosenthal, S., Nakov, P., Kiritchenko, S., Mohammad, S., Ritter, A., Stoyanovm, V.: SemEval-2015 Task 10: Sentiment Analysis in Twitter. In: International Workshop on Semantic Evaluation (SemEval) (2015)
4. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A., Potts, C.: Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In: Conference on Empirical Methods in Natural Language Processing (EMNLP) (2013)
5. Sygkounas, E., Rizzo, G., Troncy, R.: A Replication Study of the Top Performing Systems in SemEval Twitter Sentiment Analysis. In: 15^{th} International Semantic Web Conference (ISWC) (2016)