SentiME++ at SemEval-2017 Task 4A: Stacking State-of-the-Art Classifiers to Enhance Sentiment Classification

Enrico Palumbo[**,*], Efstratios Sygkounas[*], Raphaël Troncy[*] and Giuseppe Rizzo[**]

[*]EURECOM, Sophia Antipolis, France
[**]ISMB, Turin, Italy

Abstract

In this paper, we describe the participation of the SentiME++ system to the SemEval 2017 Task 4A "Sentiment Analysis in Twitter" that aims to classify whether English tweets are of positive, neutral or negative sentiment. SentiME++ is an ensemble approach to sentiment analysis that leverages stacked generalization to automatically combine the predictions of five state-of-the-art sentiment classifiers. SentiME++ achieved officially 61.30% F1-score, ranking 12^{th} out of 38 participants.

1 Introduction

The SemEval-2017 Task 4 (Rosenthal et al., 2017) focuses on the classification of tweets into positive, neutral and negative sentiment classes. In 2015, the Webis system (Hagen et al., 2015) showed the effectiveness of ensemble methods for sentiment classification by winning the SemEval-2015 Task 10 "polarity detection" challenge through the combination of four classifiers that had participated to previous editions of SemEval. In 2016, we have combined the original public release of the Webis system with the Stanford Sentiment System (Socher et al., 2013) using bagging, creating the SentiME system (Sygkounas et al., 2016b,a) which won the ESWC2016 Semantic Sentiment Analysis challenge. In bagging, the predictions of the classifiers trained on different bootstrap samples (bags) are simply averaged to obtained a final prediction. In this paper, we propose SentiME++, an enhanced version of the SentiME system that combines the predictions of the base classifiers through stacked generalization. In Section 2, we detail our approach to stack a meta-learner on top of five state-of-the-art sentiment classifiers to combine their predictions. In Section 3, we describe the experimental setup of our participation to SemEval and we report the results we obtained in Section 4. Finally, we conclude the paper in Section 5.

2 Approach

2.1 Preliminaries

SentiME++ is based on the predictions of five state-of-the-art sentiment classifiers:
NRC-Canada: winner of SemEval 2013, trains a linear kernel SVM classifier on a set of linguistic and semantic features to extract sentiment from tweets (Mohammad et al., 2013);
GU-MLT-LT: 2^{nd} ranked at SemEval 2013, uses a linear classifier trained by stochastic gradient descent with hinge loss and elastic net regularization for their predictions on a set of linguistic and semantic features (Günther and Furrer, 2013);
KLUE: 5^{th} ranked at SemEval 2013, feeds a simple bag-of-words model into popular machine learning classifiers such as Naive Bayes, Linear SVM and Maximum Entropy (Proisl et al., 2013);
TeamX: winner of SemEval 2014, uses a variety of pre-processors and features, fed into a supervised machine learning algorithm which utilizes Logistic Regression (Miura et al., 2014);
Stanford Sentiment System: one of the subsystems of the Stanford NLP Core toolkit[1], contains the Stanford Tree Parser, a machine-learning model that can parse the input text into Stanford Tree format and the Stanford Sentiment Classifier, which takes as input Stanford Trees and outputs the classification results. The output of the Stanford Sentiment System belongs to one of five classes (very positive, positive, neutral, negative, very negative) which differs from the three classes defined in SemEval. In a previous work (Sygkounas et al., 2016b), we have tested different con-

[1] http://stanfordnlp.github.io/CoreNLP/

figurations for mapping the Stanford Sentiment System classification to the three classes of the SemEval competition and finally decided to use the following strategy: very positive and positive are mapped to positive, neutral is mapped to neutral and negative and very negative are mapped to negative. The Stanford Sentiment System is used as an off-the-self classifier and is not trained with SemEval data.

2.2 Bootstrap samples

The first step in the SentiME++ approach consists in training separately the first four classifiers, using a uniform random sampling with replacement (bootstrap sampling) to generate four different training sets T_i for each of the four subclassifiers from the initial training set T. In Section 3, we report the results of the experiments that we have conducted to determine the optimal size of the samples T_i. Note that these samples are also called 'bags'. At this point, the SentiME system combines the predictions on the models trained on these bags using a simple average, while SentiME++ uses stacked generalization, as described in the next section.

2.3 Stacking

Stacked Generalization (or simply stacking) (Wolpert, 1992) is based on the idea of creating an ensemble of base classifiers and then combining them by means of a supervised classifier, also called 'meta-learner'. Stacking typically leverages the complementarity among the base classifiers to obtain a better global performance than any of the individual models. The base classifiers are trained separately and, for each input, output their prediction. The meta-learner, which is 'stacked' on top of the base classifiers, is trained on the base classifiers' predictions and aims to correct the prediction errors of the base classifiers. SentiME++ trains separately four models, uses the Stanford Sentiment System without training and uses these five outputs as a feature vector for a stacked supervised learner (Fig. 1). In detail, the SentiME++ approach works can be divided in a training and a testing phase:

Training phase: (1) generate four bootstrap samples T_i by sampling n tweets from the original training set T, where $n = s * |T|$ and s is a parameter that has to be fixed experimentally (2) train separately NRC-CANADA, GU-MLT-LT, KLUE, TeamX classifiers on the samples T_i and store the trained models; (3) use the four trained models and the Stanford Sentiment System to predict the sentiment of each tweet $t \in T$, producing a training set for the stacking layer T_{stack}; (4) Train the meta-learner on T_{stack}.

Testing phase: (1) use the four trained models and the Stanford Sentiment System to predict the sentiment of each tweet $t \in T_{test}$ producing a test set for the stacking layer T_{test}; (2) test the trained meta-learner on T_{test}.

Note that the described approach is slightly different from the standard procedure of Stacked Generalization described in (Wolpert, 1992), which is normally not based on bootstrap samples, but rather on disjoint splits of the training set. This variation is mainly due to the will of building SentiME++ as an incremental enhancement of the existing SentiMe system, without disrupting its base training mechanism. The meta-learner that is used as default in SentiME++ is a Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel (Scholkopf et al., 1997). Different choices are possible, but Support Vector Machines are well-studied methods in machine learning, able to be trained efficiently and to limit over-fitting. This method depends on two hyper-parameters, i.e. parameters that are not automatically learnt and that constitutes parameters of the algorithm itself: the regularization constant C and the parameter of the radial basis function γ. In order to optimize the performance of the stacking layer, we have chosen these parameters using a grid-search cross validation approach (Hsu et al., 2003). The process works as follows: (1) define a range for hyper-parameters $C \in [C_1...C_m]$ and $\gamma \in [\gamma_1...\gamma_n]$; (2) train the model with all possible pairs (C_i, γ_j); (3) compute scores with k-fold cross validation for (C_i, γ_j) pair; (4) find the best pair (C_i, γ_j) according to k-fold cross validation score.

SentiME is implemented in Java and the stacking process that characterizes SentiME++ is performed by a python script working on top of the results obtained by the SentiME system. The source code is available on github[2]. It uses a variety of lexicons (Table 1).

[2] https://github.com/MultimediaSemantics/sentime

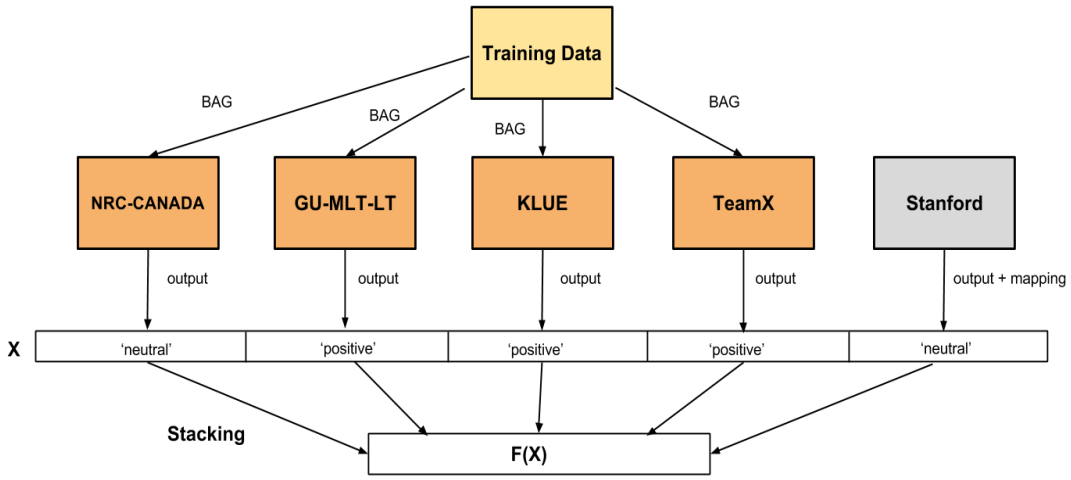

Figure 1: Illustration of the SentiME++ approach: bootstrap samples (bags) are generated to train four state-of-the-art sentiment classifiers, the Stanford System is used without training and their predictions are used as a feature vector for a meta-learner.

Lexicon	# of Words and phrases	Classifiers
AFINN-111	2477	TeamX
Bingliu	6800	NRC-Canada,TeamX
Hashtag	16,862 unigrams	NRC-Canada
NRC-emotion-lexicon-v0.92	14,182 unigrams	NRC-Canada,TeamX
Sentiment140	62,468 unigrams	NRC-Canada,TeamX
SentiWordNet_3.0.0	155.287	NRC-Canada,GU-MLT-LT,TeamX
SentiStrength	16,000 social web texts	KLUE

Table 1: Lexicons used by each sub-classifier included into SentiME++

3 Experimental Setup

In this section, we describe the experimental setup of the SentiME++ system for the participation to the SemEval2017 Task4A challenge.

3.1 Bootstrap samples size

One of the parameters of the SentiME++ model is the size of the bootstrap samples T_i. Different sampling sizes have been experimented, ranging from 33% to 175% of the size of the initial training set T. In order to determine an optimal size, we have tested the SentiME bagging approach, which simply averages the predictions of the base classifiers, on the SemEval2013-test-B dataset training the models with different random extractions of the SemEval2013-train+dev-B dataset. The experiment was repeated three times to mitigate the randomness due to the random extractions and we observed that a 150% size[3] led to the best performance on SemEval2013-test-B dataset (Sygkounas et al., 2016a).

3.2 Encoding categorical features

In order to use the predicted sentiment classes as features for a meta-learner in the stacking layer, it is necessary to specify an encoding scheme, which allows the system to interpret the class values 'Positive', 'Neutral' and 'Negative'. These values could be simply mapped to integers $0, 1, 2$, but the meta-learner, expecting continuous or binary inputs, would interpret it as an ordered sequence of real values. To avoid this, we use a one-hot encoding scheme, i.e. m categorical values are turned into a m dimensional binary vector where only one element at the time is active. In this specific case, the encoding that we have used is: 'positive'=$[0, 0, 1]$, 'neutral'=$[0, 1, 0]$, 'negative'=$[1, 0, 0]$.

3.3 Hyper-parameters optimization

In order to optimize the performance of the SVM meta-learner, we have performed the grid-search

[3]Note that this implies that there are duplicates among the training examples

cross validation described in Section 2 on the SemEval2013-train+dev-B dataset using 10-folds. The experiment has been performed using as a range an array of 30 logarithmically spaced values for γ from 10^{-9} to 10^{3} and for C from 10^{-2} to 10^{10}. The best obtained (C, γ) pair, i.e. the pair producing the best prediction score, which has been used for the participation to the challenge is: $(C, \gamma) = (0.174, 0.028)$. The implementation of the SVM classifier and of the grid-search cross validation procedure has been carried out using the python library scikit-learn[4].

4 Results

We started our experiments by training the system on different combinations of SemEval datasets, thus producing different trained models: **model 1**: SemEval2013-train+dev; **model 2**: model 1 + SemEval2013-test + SemEval2014-test + Twitter2014-sarcasm + SemEval2015-train + SemEval2015-test; **model 3**: model 2 + SemEval2016-dev + SemEval2016-test.
In order to compare the performance of these different trained models, we have chosen as a test set the SemEval2016-test dataset, as it is the largest in size (33k tweets) and the most recent of SemEval test sets. The results obtained from this experiment are illustrated in Table 2. We observe

	Model 1	**Model 2**	**Model 3**
SentiME++	65.69	71.24	94.80
SentiME	64.35	70.23	86.87

Table 2: Comparison among trained models on SemEval2016-test dataset for SentiME and SentiME++ according to F1 scores

that for all models, SentiME++ performs better than SentiME, proving the efficiency of stacked generalization with respect to bagging for combining the predictions of classifiers. For Model 1 and Model 2 the difference is around 1% and for Model 3 around 8%. Model 2 performs about 6% better than Model 1: this can be explained by the size of the training set which is bigger. Model 3 achieves the highest performance but the test dataset is part of the training dataset. While being aware that this introduces a bias in this evaluation, we also see that using more training data enhances the performance of the system and thus we opt for using SentiME++ with Model 3 for the final submission. Being the process of bootstrap sampling stochastic, we ran the system four times and computed the F1-score on the SemEval2017 development dataset and, after the release of the gold standard for the test set, on the SemEval2017 test dataset (Tab. 3). Run 4 has been submitted as

SentiME++	**Run 1**	**Run 2**	**Run 3**	**Run 4**
Dev	84.63	86.15	85.13	**86.16**
Test	60.70	60.90	**63.40**	61.30

Table 3: Comparison among runs on SemEval2017-dev and SemEval2017-test dataset for SentiME++ according to F1 scores.

it was the best performing on the development set, but, a posteriori, we can observe that Run 3 performs better on the test set. We also observe a significant performance drop from the development to the test set. We believe that this might be due to the marked difference in the category distributions of the tweets in the two datasets (see Tab.4 in (Rosenthal et al., 2017)). The best SentiME++ run at SemEval2017 Task 4 Sub-Task A would rank 8^{th} out of 38 participants.

5 Conclusion

In this paper, we have presented SentiME++, a sentiment classifier that combines the predictions of five state-of-the-art systems through stacking. SentiME++ achieved officially 61.30% F1-score, ranking 12^{th} out of 38 participants. We have shown how stacking can improve the combination of the classifiers with respect to bagging, implemented in the previous version of SentiME, evaluating it on SemEval2017 Challenge datasets. We have described an experimental procedure to determine an appropriate size of the bootstrap samples and optimize hyper-parameters of the meta-learner. In general, we provide a further evidence of the power of the ensemble approach applied to sentiment analysis. As a future work, we plan to improve the bootstrap sampling process by taking into account the class distributions of the tweets, to determine the bag sizes directly using SentiME++, to include more base classifiers and experiment different meta-learners.

Acknowledgments

This work was partially supported by the innovation activities 3cixty (14523) and PasTime (17164)

[4] http://scikit-learn.org/stable/

of EIT Digital.

References

Tobias Günther and Lenz Furrer. 2013. GU-MLT-LT: Sentiment Analysis of Short Messages using Linguistic Features and Stochastic Gradient Descent. In *7^{th} International Workshop on Semantic Evaluation (SemEval)*.

Matthias Hagen, Martin Potthast, Michel Büchner, and Benno Stein. 2015. Webis: An Ensemble for Twitter Sentiment Detection. In *9^{th} International Workshop on Semantic Evaluation (SemEval)*.

Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. 2003. A practical guide to support vector classification. http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

Yasuhide Miura, Shigeyuki Sakaki, Keigo Hattori, and Tomoko Ohkuma. 2014. TeamX: A Sentiment Analyzer with Enhanced Lexicon Mapping and Weighting Scheme for Unbalanced Data. In *8^{th} International Workshop on Semantic Evaluation (SemEval)*.

Saif Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. In *7^{th} International Workshop on Semantic Evaluation (SemEval)*.

Thomas Proisl, Paul Greiner, Stefan Evert, and Besim Kabashi. 2013. KLUE: Simple and robust methods for polarity classification. In *7^{th} International Workshop on Semantic Evaluation (SemEval)*.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 Task 4: Sentiment Analysis in Twitter. In *11^{th} International Workshop on Semantic Evaluation (SemEval)*. Vancouver, Canada.

Bernhard Scholkopf, Kah-Kay Sung, Christopher JC Burges, Federico Girosi, Partha Niyogi, Tomaso Poggio, and Vladimir Vapnik. 1997. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE transactions on Signal Processing* 45(11):2758–2765.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Efstratios Sygkounas, Giuseppe Rizzo, and Raphaël Troncy. 2016a. A Replication Study of the Top Performing Systems in SemEval Twitter Sentiment Analysis. In *15^{th} International Semantic Web Conference (ISWC)*.

Efstratios Sygkounas, Giuseppe Rizzo, and Raphaël Troncy. 2016b. Sentiment Polarity Detection From Amazon Reviews: An Experimental Study. In *13^{th} European Semantic Web Conference (ESWC), Semantic Sentiment Analysis Challenge*. pages 108–120.

David H. Wolpert. 1992. Stacked generalization. *Neural networks* 5(2):241–259.