Lacam&Int@UNIBA at the EVALITA 2016-SENTIPOLC Task

Vito Vincenzo, Covella,

Berardina De Carolis

Department of Computer Science

University of Bari "Aldo Moro", Italy

covinc93@gmail.com, berardina.decarolis@uniba.it

Stefano Ferilli,

Domenico Redavid

Department of Computer Science

University of Bari "Aldo Moro", Italy

stefano.ferilli@uniba.it, domenico.redavid@uniba.it

Abstract

English. This paper describes our first experience of participation at the EVALITA challenge. We participated only to the SENTIPOLC Sentiment Polarity subtask and, to this purpose we tested two systems, both developed for a generic Text Categorization task, in the context of the sentiment analysis: SentimentWS and SentiPy. Both were developed according to the same pipeline, but using different feature sets and classification algorithms. The first system does not use any resource specifically developed for the sentiment analysis task. The second one, which had a slightly better performance in the polarity detection subtask, was enriched with an emoticon classifier in order to fit better the purpose of the challenge.

Italiano. *Questo articolo descrive la nostra prima esperienza di partecipazione ad EVALITA. Il nostro team ha partecipato solo al subtask inerente il riconoscimento della Sentiment Polarity, In questo contesot abbiamo testato due sistemi sviluppati genericamente per la Text Categorization applicandoli a questo specifico task: SentimentWS e SentiPy. Entrambi i sistemi usano la stessa pipeline ma con set di feature e algoritmi di classificazione differenti. Il primo sistema non usa alcuna risorsa specifiche per la sentment analysis, mentre il secondo, che si è classifcato meglio, pur mantendendo la sua genericità nella classificazione del testo, è stato arricchito con un classificatore per le emoticon per cercare di renderlo più adatto allo scopo della challenge.*

1 Introduction

We tested two systems to analyze the Sentiment Polarity for Italian. They were designed and created to be generic Text Categorization (TC) systems without any specific feature and resource to support Sentiment Analysis. We used them in various domains (movie reviews, opinion about public administration services, mood detection, Facebook posts, polarity expressed in the linguistic content of speech interaction, etc.).

Both systems were applied to the EVALITA 2016 SENTIPOLC Sentiment Polarity detection subtask (Barbieri et al., 2016) in order to understand whether, notwithstanding their "general-purpose" and context-independent setting, they were flexible enough to reach a good accuracy. If so, this would mean that the Sentiment Analysis task could be approached without creating special resources for this purpose, which is known to be a costly and critical activity, or that, if available, these resources may improve their performance.

We present here only the results of the constrained runs in which only the provided training data were used to develop the systems.

The first system was entirely developed by the LACAM research group (all the classes used in the pipeline). After studying the effect of different combinations of features and algorithms on the automatic learning of sentiment polarity classifiers in Italian based on the EVALITA SENTIPOLC 2014 dataset, we applied the best one to the training set of EVALITA 2016 in order to participate to the challenge.

The second system was developed using the scikit-learn (Pedregosa et al., 2011) and NLTK libraries (Bird et al., 2009) for building the pipeline and in order to optimize the performance on the provided training set, classifications algorithms and feature sets, different from those used in SentimentWS, were tested.

Even if initially they have been conceived as a generic TC system, with the aim of tuning it for the SENTIPOLC task, they considered also the emoticons present in the tweets. In the first system this was made by including them in the set of features, while in the second one emoticons were handled by building a classifier whose result was considered to influence the sentiment polarity detection. The results obtained by the two systems are comparable even if the second one shows a better overall accuracy and ranked higher than the first one in the challenge.

2 Systems Description

2.1 SentimentWS

In a previous work (Ferilli et al., 2015) we developed a system for Sentiment Analysis/Opinion Italian. It was called SentimentWS, since it has been initially developed to run as a web-service in the context of opinion coming from web-based applications. SentimentWS casts the Sentiment Classification problem as a TC task, where the categories represent the polarity. To be general and context-independent, it relies on supervised Machine Learning approaches. To learn a classifier, one must first choose what features to consider to describe the documents, and what is the learning method to be exploited. An analysis of the state-of-the-art suggested that no single approach can be considered as the absolute winner, and that different approaches, based on different perspectives, may reach interesting results on different features. As regards the features, for the sake of flexibility, it allows to select different combinations of features to be used for learning the predictive models. As regards the approaches, our proposal is to select a set of approaches that are sufficiently complementary to mutually provide strengths and support weaknesses.

As regards the internal representation of text, most NLP approaches and applications focus on the lexical/grammatical level as a good tradeoff for expressiveness and complexity, effectiveness and efficiency. Accordingly, we have decided to take into account the following kinds of descriptors:

- single normalized words (ignoring dates, numbers and the like), that we believe convey most of informational content in the text;
- abbreviations, acronyms, and colloquial expressions, especially those that are often found in informal texts such as blog posts on the Internet and SMS';
- n-grams (groups of *n* consecutive terms) whose frequency of occurrence in the corpus is above a pre-defined threshold, that sometimes may be particularly meaningful;
- PoS tags, that are intuitively discriminant for subjectivity;
- expressive punctuation (dots, exclamation and question marks), that may be indicative of subjectivity and emotional involvement.

In order to test the system in the context of Sentiment Analysis we added emoticons in the set of features to be considered, due to their direct and explicit relationship to emotions and moods.

As regards NLP pre-processing, we used the TreeTagger (Schmid, 1994) for PoS-tagging and the Snowball suite (Porter, 2001) for stemming. All the selected features are collectively represented in a single vector space based on the real-valued weighting scheme of Term Frequency - Inverse Document Frequency (TF-IDF) (Robertson, 2004). To have values into [0, 1] we use cosine normalization.

To reduce the dimensionality of the vector space, Document Frequency (i.e., removing terms that do not pass a predefined frequency threshold) was used as a good tradeoff between simplicity and effectiveness. To build the classification model we focused on two complementary approaches that have been proved effective in the literature: a similarity-based one (Rocchio) and a probabilistic one (Naive Bayes). SentimentWS combines the above approaches in a committee, where each classifier (i = 1,2) plays the role of a different domain expert that assigns a score sik to category ck for each document to be classified. The final prediction is obtained as class c = arg maxk Sk, considering a function Sk = f (s1k; s2k). There is a wide range of options for function f (Tulyakov et al., 2008). In our case we use a weighted sum, which requires that the values returned by the single approaches are comparable, i.e. they refer to the same scale. In fact, while the Naive Bayes approach returns probability values, Rocchio's classifier returns similarity values, both in [0; 1].

2.2 SentiPy

SentiPy has been developed using the scikit-learn and NLTK libraries for building the pipeline and, in order to optimize the performance on the provided training set, classifications algorithms and feature sets, different from those used in SentimentWS, were tested. It uses a committee of two classifiers, one for the text component of the message and the other for the emoticons. For the first classifier we use a very simple set of features, any string made at least of two chars, and linear SVC as classification algorithm.

Even if this might seem too simple, we made some experiments in which we tested other configurations of features taking advantage of i) lemmatization, ii) lemmatization followed by POS-tagging, iii) stemming, iv) stemming followed by POS-tagging. All of them were tested with and without removing italian's stopwords (taken from nltk.corpus.stopwords.words("italian")).

We tested also other classification algorithms (Passive Aggressive Classifier, SGDClassifier, Multinomial Naive Bayes), but their performance was less accurate than the one of linear SVC, that we selected.

Before fitting the classifier text preprocessing was performed according to the following steps:

- Twitter's "mentions" (identified by the character '@' followed by the username) and http links are removed;
- retweets special characters ("RT" and "rt") are removed;
- hashtags are "purged", removing the character '#' followed by the string, which is then left unmodified;
- non-BMP utf8 characters (characters outside the Basic Multilingual Plane), usually used to encode special emoticons and emojis used in tweets, are handled by replacing them with their hexadecimal encoding; this is done to avoid errors while reading the files.

After doing the aforementioned experiments using the training and testing sets provided by sentipolc2014, which was also used to fine-tune the parameters used by the LinearSVC algorithm, we compared the most successful approaches: tokenization done using nltk.tokenize.TweetTokenizer followed by stemming and feature extraction simply done by using the default tokenizer provided by *scikit* (it tokenizes the string by extracting words of at least 2 letters).

The best configurations are those shown in Table 1 and Table 2.

Tokenization	Scikit-Learn default tokenizer
Maximum document frequency CountVectorizer parameter	0.5
Maximum number of terms for the vocabulary	unlimited
n-grams	Unigrams and bigrams
Term weights	tf-idf
Vector's normalization	l2
fit_intercept classifier parameter	False
dual classifier parameter	True
Number of iterations over training data	1000
Class balancing	automatic

Table 1: SentiPy - positive vs all best configuration based on Sentipolc 2014.

Tokenization	Scikit-Learn default tokenizer
Maximum document frequency CountVectorizer parameter	0.5
Maximum number of terms for the vocabulary	unlimited
n-grams	Unigrams and bigrams
Term weights	tf-idf
Vector's normalization	l2

fit_intercept classifier parameter	True
dual classifier parameter	True
Number of iterations over training data	1000
Class balancing	automatic

Table 2: SentiPy - negative vs all best configuration based on Sentipolc 2014.

These two configurations, which had the same fine-tuned LinearSVC's parameters, were compared observing the evaluation data obtained testing them on sentipolc2016 training set, taking advantage of a standard common technique: 10-fold cross validation, whose results are shown in Table 3.

The obtained results were comparable therefore we selected the configuration shown in the first two rows of Table 3 combined with the emoticon classifier since it was not presented in the SentimentWS system.

Configuration	**F1-score macro averaging**	**Accuracy**
VotingClassifier default tokenization – *positive vs all*	0,70388	0,77383
VotingClassifier default tokenization – *negative vs all*	0,70162	0,70648
VotingClassifier stemming – *positive vs all*	0,70654	0,75424
VotingClassifier stemming – *negative vs all*	0,6952	0,70351

Table 3: 10-fold on Sentipolc 2016 training set.

As far as the emoticons and emojis are concerned, in this system we decided to exclude them from the features set, solution adopted in SentimentWS, and train a classifier according to the valence with whom the tweet was labeled. This approach may be useful to detect irony or for recognizing valence in particular domains in which emoticons are used with a different meaning. Emoticons and emojis were retrieved using a dictionary of strings and some regular expressions. The emoticons and emojis retrieved are replaced with identifiers, removing all other terms not related to the emoticons, thus obtaining a matrix emoticons-classes. The underlying classifier takes this matrix as input and creates the model that will be used in the classification phase. The algorithm used in the experiments is the Multinomial Naive Bayes.

The committee of classifiers was built using the VotingClassifier class, which is provided by the Scikit-Learn framework. The chosen voting technique is the so called "hard voting": it is based on the majority voting rule; in case of a tie, the classifier will select the class based on the ascending sort order (classifier 1 → class 2; classifier 2 → class 1; class 1 will be the selected class).

3 Experiments

Both systems were tested on other domains before applying them to the SENTIPOLC subtask.

In the results tables, for each class (*positive* and *negative*) 0 represents the value "False" used in the dataset annotations for the specific tweet and class, 1 represents "True", following the task guidelines of Sentipolc 2016. Thus the cell identified by the row *positive* and the column *prec.0* shows the precision related to the tweets with positive polarity annotations set to False. The meaning of the other cells can be obtained analogously.

3.1 SentimentWS Results

SentimentWS was tested initially on a dataset of 2000 reviews in Italian language, concerning 558 movies, taken from http://filmup.leonardo.it/. In this case, classification performance was evaluated on 17 different feature settings using a 5-fold cross-validation procedure. Equal weight was assigned to all classifiers in the SentimentWS committee. Overall accuracy reported in (Ferilli et al., 2015) was always above 81%. When Rocchio outperformed Naive Bayes, accuracy of the committee was greater than that of the components; in the other cases, corresponding to settings that used n-grams, Naive Bayes alone was the winner.

Before tackling the EVALITA 2016 SENTIPOLC task, in order to tune the system on a (hopefully) similar environment, we tested our system on the EVALITA 2014 dataset and determined in this way the combination of features that had a better accuracy on this dataset.

We tested the system using a subset of ~900 tweet (taken from the dataset provided in Sentipolc 2014), in order to find the best configuration of parameters, which resulted to be the following one:

- term normalization: lemmatization;
- minimum number of occurrences for a term to be considered: 3
- POS-tags used: NOUN-WH-CLI-ADV-NEG-CON-CHE-DET-NPR-PRE-ART-INTADJ-VER-PRO-AUX
- n-grams: unigrams

With the configuration described above, the system SentimentWS was able to classify the whole test set of Sentipolc 2014 (1935 tweet) obtaining a combined F-score of 0.6285.

The previously mentioned best configuration was also used in one of the two runs sent for Sentipolc 2016, obtaining a combined F-score of 0.6037, as shown in Table 4.

class	prec. 0	rec. 0	F-sc. 0	prec. 1	rec. 1	F-sc. 1	F-sc
positive	0.8642	0.7646	0.8113	0.2841	0.4375	0.3445	0.5779
negative	0.7087	0.7455	0.7266	0.5567	0.5104	0.5325	0.6296

Table 4: SentimentWS - Sentipolc 2016 test set - Combined F-score = 0.6037.

class	prec. 0	rec. 0	F-sc. 0	prec. 1	rec. 1	F-sc. 1	F-sc
positive	0.8792	0.7992	0.8373	0.3406	0.4858	0.4005	0.6189
negative	0.7001	0.8577	0.7709	0.6450	0.4130	0.5036	0.6372

Table 5: SentiPy@Sentipolc2016 results (LinearSVC fine-tuned + EmojiCustomClassifier) - Combined F-score = 0.6281.

3.2 SentiPy Results

With the configuration discussed above SentiPy combined F-score was 0.6281 as shown in Table 5.

We made other experiments on the Sentipolc 2016 test set after the deadline of EVALITA. Their results, even if unofficial, show significant improvements, since we managed to get 0.6403 as a combined F-score. We got it by making specific changes in the *positive vs all* classifier: we used lemmatization (without stopwords removal), unigrams (no other n-grams allowed) and the parameter fit_intercept of the LinearSVC algorithm was set to True. The other parameters remained unchanged. No changes have been made to the classifier *negative vs all*.

4 Conclusions

Looking at the results of the Sentiment Polarity detection subtask we were surprised of the overall performance of the systems presented in this paper since they were simply Text Categorization systems. The only integrations to the original systems, in order to tune their performance on the sentiment polarity detection task, concerned emoticons. In SentimentWS these were included in the feature set and SentiPy was enriched with a classifier created for handling emoticons.

Besides the experiments that were executed on the SENTIPOLC dataset, we tested both systems on a dataset of Facebook posts in Italian collected and annotated by a group of researchers in our laboratories. This experiment was important in order to understand whether their performance was comparable to the one obtained in the SENTIPOLC challenge. Results in these cases were encouraging since both systems had a combined F-score higher than 0.8.

We are currently working at the improvement of the performance of the system by tuning it on the Sentiment Analysis context. To this aim we are developing a specific module to handle negation in Italian and, in our future work we plan to integrate the two systems by creating one committee including all the classifiers, moreover we plan to include an approach based on a combination of probabilistic and lexicon (De Carolis et al., 2015).

References

Barbieri, Francesco and Basile, Valerio and Croce, Danilo and Nissim, Malvina and Novielli, Nicole and Patti, Viviana 2016. Overview of the EVALITA 2016 SENTiment POLarity Classification Task. In *Proceedings of Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016)*. Associazione Italiana di Linguistica Computazionale (AILC).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: machine learning in python. *Journal of Machine Learning Research*, 12(Oct), 2825-2830.

Steven Bird, Ewan Klein, and Edward Loper (2009). Natural Language Processing with Python. O'Reilly Media Inc.

Stefano Ferilli , Berardina De Carolis, Floriana Esposito , Domenico Redavid. 2015. Sentiment analysis as a text categorization task: A study on feature and algorithm selection for Italian language. In *Proceeding of IEEE International Conference on Data Science and Advanced Analytics (DSAA).*

H. Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, pp. 44–49.

M. F. Porter, 2001. Snowball: A language for stemming algorithms," [Online]. http://snowball.tartarus.org/texts/introduction.html

Stephen Robertson. 2004. Understanding inverse document frequency: On theoretical arguments for IDF. *Journal of documentation*, 60(5), 503-520..

S. Tulyakov, S. Jaeger, V. Govindaraju, and D. Doermann. 2008. Review of classifier combination methods," ser. *Studies in Computational Intelligence (SCI)*. Springer. vol. 90, pp. 361–386.

B. De Carolis, D. Redavid, A. Bruno. 2015. A Sentiment Polarity Analyser based on a Lexical-Probabilistic Approach . In *Proceedings of IT@LIA2015 1st AI*IA Workshop on Intelligent Techniques At LIbraries and Archives co-located with XIV Conference of the Italian Association for Artificial Intelligence (AI*IA 2015).*