A Comparison between Preprocessing Techniques for Sentiment Analysis in Twitter

Giulio Angiani, Laura Ferrari, Tomaso Fontanini, Paolo Fornacciari, Eleonora Iotti, Federico Magliani, and Stefano Manicardi

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma
Parco Area delle Scienze 181/A, 43124 Parma, Italy
{giulio.angiani,laura.ferrari,tomaso.fontanini,paolo.fornacciari,
eleonora.iotti,federico.magliani,stefano.manicardi}@studenti.unipr.it

Abstract. In recent years, *Sentiment Analysis* has become one of the most interesting topics in AI research due to its promising commercial benefits. An important step in a Sentiment Analysis system for text mining is the preprocessing phase, but it is often underestimated and not extensively covered in literature. In this work, our aim is to highlight the importance of preprocessing techniques and show how they can improve system accuracy. In particular, some different preprocessing methods are presented and the accuracy of each of them is compared with the others. The purpose of this comparison is to evaluate which techniques are effective. In this paper, we also present the reasons why the accuracy improves, by means of a precise analysis of each method.

Keywords: Sentiment Analysis, Preprocessing, Naive-Bayes Multinomial.

1 Introduction

The subjective analysis of a text is the main task of *Sentiment Analysis* (SA), also called *Opinion Mining*. One of the basic tasks in SA is to predict the polarity of a given sentence, to find out if it expresses a positive, negative or neutral feeling about a certain topic [16]. Furthermore, in recent research works, SA goes beyond the concept of polarity, trying to identify the emotional status of a sentence, such as anger, sadness, happiness, etc., according to various classifications of affective knowledge [4, 6, 13, 17, 21]. The application of SA ranges over several domains, from movie reviews to social networks, which also are proliferating in both usage and architectures [9, 10]. The demand for new techniques of SA is continuously growing, due to their inherent capacity of automatic evaluation, from both the academic and industrial points of view. In the last few years, *Opinion Mining* has become a popular research field, which brings together several different areas. Due to its heterogeneity, many different techniques were analyzed and implemented, in order to get increasingly accurate systems for a certain problem statement. Most of such techniques involve the use of *Machine Learning* (ML) classification algorithms—in particular *Supervised Learning Algorithms*—, i.e., methods that are used to train a classifier, whose aim is the association of an input with its related class, chosen from a certain set of classes. The training is done by providing the classifier with several examples of inputs and their related classes. Then, the system extracts a set of *features* (or *attributes*) from each of them, in order to become capable of recognizing the class of generic data, which can be of different types [14]. The performance of a classifier could be evaluated by different metrics, such as the accuracy, which is a measure of the correctness of a classifier. Similarly, Weka provides the confusion matrix of a simulation, which is useful for the identification of the errors in the model classification. As detailed in Figure 1, these methods often include some preprocessing corpora, which make assumptions and choices on the inclusion of features in text representations, and that are used for training a classifier and also for evaluating its performance. In fact, Machine Learning algorithms need to work on data, appropriately processed by a set of operations which make assumptions and choices on the inclusion of features in text representations. This phase is a fundamental step in order for the whole system to obtain good results. Normally it includes methods for data cleaning and feature extraction and selection. A good overview of the steps and the most known algorithms for each step is explained in [12].

Thus, given a corpus of raw data sets, the first step of SA is the preprocessing of those data. Preprocessing involves a series of techniques which should improve the next phases of elaboration, in order to achieve better performances.

As illustrated in [11], online texts usually contain lots of noise and uninformative parts, such as HTML tags. This raises the dimensionality of the problem and makes the classification process more difficult. The algorithms which are most used to polish and prepare data that comes from Twitter messages include the removal of punctuation and symbols, tokenization, stemming, and stopword as showed, for example in [5] and [18].

Some of these techniques are exposed in the work of A. Balahur [2,3], which concerns the problem of classification of Twitter posts, that is, short sentences which refer to one topic. She uses a series of interesting preprocessing modules (such as emoticon replacement, tokenization, punctuation marks, word normalization, etc.) and she shows these methods in detail. However, such methods are collected together before data classification, and the emphasis of her work is not on why or how each of these modules helps improve the classifier accuracy. In fact, her work focuses on the classification of many types of sentiments, from positive, negative and neutral, to anger, disgust, fear, joy, sadness and surprise, rather than on the effectiveness of the presented preprocessing techniques. In our research, we have also collected data sets from Twitter and we have implemented some of Balahur's preprocessing ideas, finding them useful when applied to such a problem statement. However, our work focuses on their effectiveness and their performance in terms of accuracy, and such techniques are evaluated by analysing each one separately.

The work of A. Agarwal et al. [1], also based on Twitter data sets, proposes the use of emoticons as features and uses a dictionary of 8000 words associated with a pleasantness score from 1 (negative) to 3 (positive). Emoticons are divided into five categories (extremely-positive, positive, neutral, negative and extremely-negative) and they gain a score, like other words. Then, all scores are added up per sentence and divided by 3. If the result is less than 0.5, then the sentence is classified as negative. If, on the contrary, it is greater than 0.8, then the sentence belongs to the positive class. In all other cases, a neutral class is used. Basic cleaner, slang conversion and negation replacement are also used. In particular, we analyse emoticon replacement and other techniques, but we avoid giving a score to each word, leaving the task of assigning weights to features to the ML algorithm.

In the context of SemEval (Task 4)[1], for SA in Twitter, N. F. Silva et al. [20] analyse how much the accuracy of classification changes, using various algorithms: Naive-Bayes Multinomial (NBM), Support Vector Machine (SVM), AdaBoost with SVM, and AdaBoost with NBM. In this paper, we focus on preprocessing methods for a fixed classification algorithm: in fact, the only one utilized is NBM.

In [7] and [8] Twitter was analyzed as a communication medium in which it is possible to recognize certain features that identify a sort of Twitter culture. The techniques in the polish phase were chosen while taking into account this peculiar nature of tweets.

The main goal of the present work is to analyse and compare different preprocessing steps found in literature, and to actually define the best combination of the considered methods. In fact, preprocessing is often seen as a fundamental step for SA, but rarely is it carefully evaluated, thus leaving the open question of why and to what extent does it increase the accuracy of the classifier.

The data set used in this work is the one provided by SemEval [19] for Sentiment Analysis in Twitter, which is often used in many other works, such as [2, 15, 20], in order to make our results comparable with the others. The tool used to test the accuracy of the classification is Weka[2].

The paper is structured as follows. In Section 2, a brief introduction to Machine Learning steps is provided. In addition, the section describes the techniques and algorithms used for classification and features selection. In Section 3, the considered preprocessing techniques are presented in detail. Section 4 shows the performance of the obtained classifier for each preprocessing method, on two different data sets, and discusses the effectiveness of each of such techniques. In Section 5, a brief discussion of the proposed work and an analysis of the achieved results conclude the paper.

[1] http://alt.qcri.org/semeval2016/task4/

[2] http://www.cs.waikato.ac.nz/ml/weka

2 Algorithms and Techniques

This section describes the different preprocessing modules that have been used in this work[3]. All of them are built in Python[4] and they work with version 2.7.

The pipeline of the project is organized in the following way. Firstly, we obtain the 2015 and 2016 data sets (both training and test) of Twitter Sentiment Analysis from SemEval. The training sets are subjected to the various preprocessing techniques analysed in this work. After the text of each instance of a set has been preprocessed, the resulting sentences (the cleaned tweets) become the instances of a new training set. Then, such a data set is used for training a classifier and the corresponding test set is classified by Weka. Finally, the accuracies of the classifiers obtained from different preprocessing modules are compared with each other, in order to evaluate the efficiency and effectiveness of each technique.

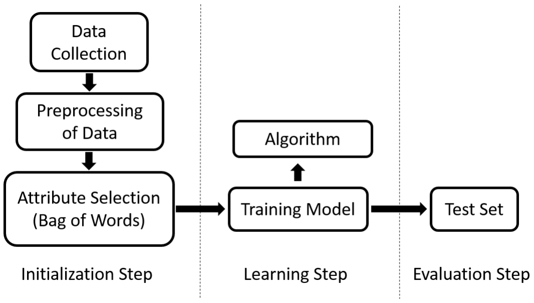

Fig. 1. Steps for training a classifier for sentiment analysis. Firstly, data have to be prepared in order to obtain a data set – namely, the training set – by means of preprocessing and feature selection methods. Then, such a data set is involved in the learning step, which uses ML algorithms and yields a trained classifier. Finally, the classifier has to be tested on a different data set – namely, the test set.

The classifier is made by using *Naive-Bayes Multinomial* (NBM) method, i.e., a ML algorithm that gives rise to a probabilistic classifier, which works on the basis of the Bayes Theorem, with the strong assumption that features are mutually independent. Let $X = (x_1, \ldots, x_n)$ be the feature vector of an instance in the data set, that is, a binary vector that takes into account the presence of a feature in that instance, and let $C_1, \ldots, C_K$ be the possible outputs (classes). The problem is to gain the posterior probability of having the class C_k as output, given the feature vector X, and given the prior probability $p(C_k)$ for each class. Thanks to the Bayes Theorem and the independence between features, the probability that needs to be estimated is the conditional $p(X|C_k)$, and then a classifier is trained with a decision rule, such as the Maximum a

[3] https://github.com/fmaglia/SA_cleaners

[4] http://www.python.org

Posteriori (MAP) rule. In summary, the probabilistic model of NBM can be expressed in terms of the following formula:

$$p(X|C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{ki}^{x_i}$$

where $X = (x_1, \ldots, x_n)$ is the feature vector, p_i is the probability that the feature i appears, C_k is a class and p_{ki} is the probability that feature i occurs in the class C_k. Then, Information Gain (IG) is the algorithm used for feature selection [REF]. It evaluates the presence or absence of a feature in a document by measuring its probability of belonging to a class. The amount of information needed to exactly classify an instance D is defined recursively as follows.

$$\mathit{Info}_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D_i|} \cdot \mathit{Info}(D_j)$$

when the instance D is divided by some feature attribute $A = \{a_1, \ldots, a_v\}$ into sub-instances $D_1, \ldots, D_v$.

3 Preprocessing Phases

3.1 Basic Operation and Cleaning

This first module manages basic cleaning operations, which consist in removing unimportant or disturbing elements for the next phases of analysis and in the normalization of some misspelled words. In order to provide only significant information, in general a clean tweet should not contain URLs, hashtags (i.e. #happy) or mentions (i.e. @BarackObama). Furthermore, tabs and line breaks should be replaced with a blank and quotation marks with apexes. This is useful in order to obtain a correct elaboration by Weka (i.e. not closing a quotation mark causes a wrong read by the data mining software causing a fatal error in the elaboration). After this step, all the punctuation is removed, except for apexes, because they are part of grammar constructs such as the genitive.

The next operation is to remove the vowels repeated in sequence at least three times, because by doing so the words are normalized: for example, two words written in a different way (i.e. *cooooool* and *cool*) will become equals. Another substitution is executed on the laughs, which are normally sequences of "a" and "h". These are replaced with a "laugh" tag.

The last step is to convert many types of emoticons into tags that express their sentiment (i.e. *:)* → *smile_happy*). The list of emoticons is taken from Wikipedia[5].

Finally, all the text is converted to lower case, and extra blank spaces are removed.

[5] http://en.wikipedia.org/wiki/List_of_emoticons

All the operations in this module are executed to try to make the text uniform. This is important because during the classification process, features are chosen only when they exceed a certain frequency in the data set. Therefore, after the basic preprocessing operations, having different words written in the same way helps the classification.

3.2 Emoticon

This module reduces the number of emoticons to only two categories: *smile_positive* and *smile_negative*, as shown in Table 1.

Table 1. List of substituted Emoticons

smile_positive	smile_negative
0:-)	>:(
:)	;(
:D	>:)
:*	D:<
:o	:(
:P	:|
;)	>:/

Table 2. Likelihood of some Emoticon

Features	$P(X \mid C_{pos})$	$P(X \mid C_{neg})$
:)	0.005107	0.000296
:(	0.000084	0.001653
:*	0.001055	0.000084
;)	0.000970	0.000084

Table 3. Likelihood of Smile_Positive and Smile_Negative

Features	$P(X \mid C_{pos})$	$P(X \mid C_{neg})$
smile_positive	0.007320	0.000718
smile_negative	0.000336	0.002283

This is done to increase the weight of these features in the classification phase and to reduce the complexity of the model. In Table 2 and Table 3, it is possible to notice how much the likelihood of a feature changes.

3.3 Negation

Dealing with negations (like “not good”) is a critical step in Sentiment Analysis. A negation word can influence the tone of all the words around it, and ignoring negations is one of the main causes of misclassification.

In this phase, all negative constructs (*can’t, don’t, isn’t, never etc*) are replaced with *“not”*.

This technique allows the classifier model to be enriched with a lot of negation bigram constructs that would otherwise be excluded due to their low frequency.

Table 4 shows some examples of extracted features and their *likelihood*.

Table 4. Example of Features Extracted

Features	$p(X\|C_{pos})$	$p(X\|C_{neg})$
not wait	0.002345	0.000304
not miss	0.000651	0.000043
not like	0.000004	0.000391

3.4 Dictionary

This module uses the external python library PyEnchant[6], which provides a set of functions for the detection and correction of misspelled words using a dictionary.

As an extension, this module allows us to substitute slang with its formal meaning (i.e., *l8* → *late*), using a list. It also allows us to replace insults with the tag “bad_word”.

The motivation for the use of these functions is the same as for the basic preprocessing operation, i.e., to reduce the noise in text and improve the overall classification performances.

3.5 Stemming

Stemming techniques put word variations like “great”, “greatly”, “greatest”, and “greater” all into one bucket, effectively decreasing entropy and increasing the relevance of the concept of “great”. In other words, Stemming allows us to consider in the same way nouns, verbs and adverbs that have the same radix.

[6] `http://pythonhosted.org/pyenchant`

This method is already implemented in Weka and the algorithm in use is *IteratedLovinsStemmer*[7].

As in the case of emoticons, with the use of this technique it is possible to combine features with the same meaning and reduce the entropy of the model.

3.6 Stopwords

Stop words are words which are filtered out in the preprocessing step. These words are, for example, pronouns, articles, etc. It is important to avoid having these words within the classifier model, because they can lead to a less accurate classification.

4 Results

The data set is composed of the training and test sets.

Table 5. Data set table

Data set	Positive	Negative	Total
Training set	1339	1339	2678
Test set 2016	623	169	792
Test set 2015	343	174	517

The training sets are those provided by SemEval, with a little revision: neutral sentences are removed, in order to focus only on positive and negative ones. Furthermore, in the training set there are more positive sentences than negative ones. Excess positive ones have been eliminated, because they distort the Bayes model.

In the executed tests, the features collected have a minimum presence in the text that is greater than or equal to 5. The Ngrams used are only one-grams and bi-grams. Before starting the simulation with the test set, a 10-fold cross-validation is carried out. In particular, we searched for the optimal length of N-grams to potentially consider as features. In Figure 2, it can be observed that accuracy nearly peaks at N-gram = 2. Longer sequences increase the complexity of the training phase, without giving a significant improvement of the result. Also, we analysed the total number of features to consider. This parameter does not provide a monotonic improvement to the classifier quality. Instead, it peaks out at around 1500 features.

At first, the executed simulations compare *no preprocessed file* vs. *basic cleaned file*. As shown in Table 5, the resulting accuracy is strongly increased. Given the importance of the basic cleaner, we decided to use it in every case,

[7] `weka.sourceforge.net/doc.dev/weka/core/stemmers/IteratedLovinsStemmer.html`

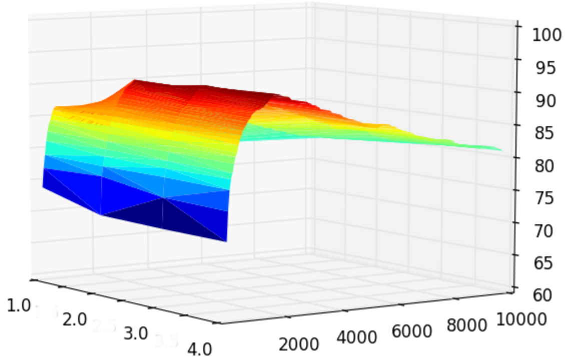

Fig. 2. Optimization of system parameters.

together with another preprocessing module, in order to evaluate their contribution together.

Stemming increases the performance, because it groups words reduced to their root form. It allows many words to be selected as useful features for the classification phase. In fact, it modifies the weight of a feature, usually increasing it.

Stopword removal enhances the system because it removes words which are useless for the classification phase. As a common example, an article does not express a sentiment but it is very present in the sentences.

Table 6. Result classification table

Technique	# Max Features	CV folds 10 [%]	Test 2016 [%]	Test 2015 [%]
No preprocess	1800	78,08	65,65	69,05
Basic	2000	80,05	65,40	74,08
Basic + Stemming	2200	**80,84**	**68,68**	**76,40**
Basic + Stopwords	1800	80,32	65,27	74,85
Basic + Negation	2000	80,40	65,65	75,04
Basic + Emoticon	2000	80,13	65,98	74,66
Basic + Dictionary	2000	78,00	64,39	75,82
All	2000	80,40	64,89	75,82
All Without Dictionary	2100	80,76	65,78	75,04

As a notable result, it is interesting that using a dictionary did not enhance the performance in our tests, but it increased the elaboration-time needed for cleaning raw data.

There is also an improvement in the accuracy of the classifier between the two SemEval test-sets: 2016 and 2015. However, this is only due to there being fewer of sentences in the last test-set, with a corresponding lower probability for the classifier to make mistakes.

5 Conclusions

Text preprocessing is an important phase in all relevant applications of data mining. In Sentiment Analysis, in particular, it is cited in virtually all available research works. However, few works have been specifically dedicated to understanding the role of each one of the basic preprocessing techniques, which are often applied to textual data.

Arguably, having a more precise measure of the impact of these basic techniques can improve the knowledge of the whole data mining process. This work adopts a straightforward methodology: it basically applies each one of the most known filters, independently, to the raw data. However, given the importance of the basic cleaner, we decided to use the basic cleaner in every case, together with another single preprocessing module, and then evaluate their joint contribution.

As an interesting result, it is worth noting that using a dictionary did not enhance the performances in our tests, but it increased the elaboration-time needed for cleaning raw data. All other techniques, however, provided significant improvements to the classifier performances. Some of the techniques simply removed useless noise in the raw data, while others increased the relevance of some concepts, reducing similar terms and expression forms to their most basic meaning.

This research has been conducted over data which originated from Twitter. In our opinion, a similar analytical work should be performed on different kinds of data sets, to have a more comprehensive understanding of the different preprocessing filters. The decision to mix some of these filters together is often correct. However, it should be better motivated by empirical data and result evaluations for various application domains and the peculiar nature of their textual data.

References

1. Agarwal, A., Xie, B., Vovsha, I., Rambow, O., Passoneau, R.: Sentiment Analysis of Twitter Data. Computer Science - Columbia University (New York, USA) (2011)
2. Balahur, A.: Sentiment Analysis in Social Media Texts. European Commission Joint Research Center (Varese, Italy) (2013)
3. Bao, Y., Quan, C., Wang, L., Ren, F.: The role of pre-processing in twitter sentiment analysis. In: International Conference on Intelligent Computing. pp. 615–624. Springer (2014)
4. Cambria, E., Olsher, D., Rajagopal, D.: Senticnet 3: a common and common-sense knowledge base for cognition-driven sentiment analysis. In: Twenty-eighth AAAI conference on artificial intelligence (2014)
5. Duncan, B., Zhang, Y.: Neural networks for sentiment analysis on twitter. In: Cognitive Informatics & Cognitive Computing (ICCI* CC), 2015 IEEE 14th International Conference on. pp. 275–278. IEEE (2015)
6. Esuli, A., Sebastiani, F.: Sentiwordnet: A publicly available lexical resource for opinion mining. In: Proceedings of LREC. vol. 6, pp. 417–422. Citeseer (2006)
7. Fornacciari, P., Mordonini, M., Tomaiuolo, M.: A case-study for sentiment analysis on twitter. In: Proceedings of the 16th Workshop ”From Objects to Agents”-WOA (2015)

8. Fornacciari, P., Mordonini, M., Tomaiuolo, M.: Social network and sentiment analysis on twitter: Towards a combined approach. In: Proceedings of the 1st International Workshop on Knowledge Discovery on the WEB –KDWEB 2015 (2015)
9. Franchi, E., Poggi, A., Tomaiuolo, M.: Blogracy: A peer-to-peer social network. International Journal of Distributed Systems and Technologies (IJDST) 7(2), 37–56 (2016)
10. Franchi, E., Tomaiuolo, M.: Distributed social platforms for confidentiality and resilience. Social Network Engineering for Secure Web Data and Services p. 114 (2013)
11. Haddi, E., Liu, X., Shi, Y.: The role of text pre-processing in sentiment analysis. Procedia Computer Science 17, 26–32 (2013)
12. Kotsiantis, S., Kanellopoulos, D., Pintelas, P.: Data preprocessing for supervised leaning. International Journal of Computer Science 1(2), 111–117 (2006)
13. Liu, B.: Sentiment analysis: Mining opinions, sentiments, and emotions. Cambridge University Press (2015)
14. Matrella, G., Parada, G., Mordonini, M., Cagnoni, S.: A video-based fall detector sensor well suited for a data-fusion approach. Assistive Technology from Adapted Equipment to Inclusive Environments, Assistive Technology Research Series 25, 327–331 (2009)
15. Neethu, M.S., Rajasree, R.: Sentiment Analysis in Twitter using Machine Learning Techniques. Department of Computer Science - College of Engineering (Trivandrum, India) (2013)
16. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Foundations and trends in information retrieval 2(1-2), 1–135 (2008)
17. Poria, S., Cambria, E., Winterstein, G., Huang, G.B.: Sentic patterns: Dependency-based rules for concept-level sentiment analysis. Knowledge-Based Systems 69, 45–63 (2014)
18. Saif, H., Fernández, M., He, Y., Alani, H.: On stopwords, filtering and data sparsity for sentiment analysis of twitter (2014)
19. Semeval: Semeval 2016 task 4: Sentiment analysis in twitter. http://alt.qcri.org/semeval2016/task4 (2016)
20. Silva, N.F.F., Hruschka, E.R., Hruschka, E.R.: Biocom Usp: Tweet Sentiment Analysis with Adaptive Boosting Ensemble. University of Sao Paulo and Federal University of Sao Carlos (Sao Carlos,Brasil) (2014)
21. Strapparava, C., Valitutti, A., et al.: Wordnet affect: an affective extension of wordnet. In: LREC. vol. 4, pp. 1083–1086 (2004)