General Sentiment Decomposition: opinion mining based on raw NL text

Keywords: *GSD, Naïve Bayes*, NLP, Twitter.com, TripAdvisor.com, Booking.com*

1 Introduction

Word of Mouth political and marketing importance is growing day by day. Those phenomena can be directly observed in everyday life, e.g.: the rise of influencers and social media managers. If people talk about a specific product, then more people are encouraged to buy it and vice versa. This effect is amplified proportionally to how high the consideration or close the relationship between the potential customer and the reviewer is. Furthermore, considering the negative reporting bias, it is easy to understand how customer satisfaction is of absolute interest for any field.
We propose an algorithm to extract the sentiment from a natural language text corpus. The combined approach of Neural Networks, characterized by high predictive power but at the cost of harder interpretation, with more straightforward and informative models, allows not only to predict how much a sentence is positive (negative) but also to quantify a sentiment with a numeric value. The assessment of an objective quantity improves the interpretation of the results in many fields. For example, it is possible to identify specific critical sectors that require intervention to improve the offered services, to find the strengths of the company (useful for advertising campaigns), and, if time information is present, to analyze trends on macro/micro topics.
After showing how to properly reduce the dimensionality of the textual data with a data-cleaning phase, we shown how to combine: WordEmbedding, K-Means clustering, SentiWordNet, and the Naïve Bayes* model. We shown then how to use such a model for unlabelled data while preserving the interpretability and the good performance.

2 Methods

After a data cleaning phase that use a dimensionality reduction technique based on a combination of WordEmbedding (Mikolov et al. [1]) and K-Means clustering, a temporally [-1,1] sentiment score is calculated with the help of SentiWordNet (Baccianella et al. [2]). This score permits the creation of a temporally label that will be the input for the Naïve Bayes* model, allowing to use it for unlabelled data while preserving the interpretability and the good performance of the model itself.

2.1 Data Cleaning

The raw data that is downloaded is certainly not fit for the analysis. It has a lot of unnecessary words like stop words that do not contribute to explain the meaning of the sentence and acronyms whose meaning is difficult to decipher and hence tend to confuse the algorithm. Moreover, it contains emojis which have useful information, so they have to be converted into meaningful text. This phase is the first step for pre-process the data and make it useful for the analysis. Below mentioned are the details of all the processes used for the data cleaning phase in a subsequent way for each single observation in the dataset.

pre-processing: some basic - but necessary - filtration. It is important to remove links (expecially if just a partial one), acronyms (as their meaning is difficult to decipher) or recurrent and meaningless keywords (like: RT (re-tweets), @username, uninterested #hashtags etc.)

emoticons conversion & emoji replacement: valuable information is contained inside emoticons (like :-) or :-( ) and in emojis (like ☺ or ☹). We replace emoticons and emojis in sentences by their corresponding meaning. In that way all meaningful symbols are converted into text

stop words & alphanumeric characters: the incoming text is first tokenized into separate words. Cases of all alphabets are normalized to lowercase. Next, stop words like “a”, “the”, “do”, “to”, and punctuation symbols are removed from the data. However, the negative words like “not" are kept

stemming: the tokens are stemmed; they are reduced to its root or base form. For example, “fishing”, “fished”, “fisher” are all reduced to the stem “fish”. In that way we merge words that are related to the same topic by their root or base form

2.2 SentiWordNet & WordEmbedding combination

The main goal of Word Embedding is to reduce the dimensionality of text data. In order to achieve this goal, we go thought the hypernyms and lemmas phases. Other fundamental concepts and terminologies to better understand the next steps are the following: a) synsets: a collection of words that have a similar meaning, b) hypernyms: more abstract terms concerning the name of particular synsets, c) lemmas: a WordNet’s version of an entry in a dictionary: A word in canonical form, with a single meaning, d) merging words by their meaning: The Word Embedding function iterates through every word of the received text, and, for every word, it fetches the synset which it belongs to. Using the synset name, it fetches the hypernym related to that word. Finally, the hypernym name is used to find the most similar word, replacing the actual word in the text.

To obtain the temporally sentiment score, we use ad hoc function that uses the newspapers pre-trained Word Embedding produced by Google, with a K-Means clustering technique. A number of clusters, say λ, is produced, and the centroid-word as the word that replaces all the other words present in that cluster is identified. The default λ-value can be estimated by cross validation, calculating the best accuracy (or other performance metrics) within a labelled dataset (e.g. Booking.com or TripAdvisor data).
Once data is correctly cleaned and all the words with the same meaning are merged in a single one, it is possible to calculate the total sentiment score of each observation. The sentiment score ($pos_{score} - neg_{score}$) of the word in SentiWordNet determines the polarity of each word. Then all the scores of all the words present in the parsed observation are calculated and they are averaged to obtain the overall score.

2.3 Naïve Bayes* Classifier

In Romano et al. [3] it has been implemented an ad-hoc classifier able to predict, as accurately as possible, a comment as negative or positive based on the words included in its content. This Naïve Bayes* classifier derives from a modification of the original classifier having the same name and resulted as the best performing one in terms

of generalizability among several of the most commonly used classifiers. The basic features of Naïve Bayes* applied to reviews' content are as follows. For a specific review r and for each word w ($w \in BoW$), we consider the log-odds ratio of w,

$$\begin{aligned} LOR(w) &= \log\left[\frac{P(c_{neg}|w)}{P(c_{pos}|w)}\right] \approx \\ &\approx \log\left[\frac{P(w|c_{neg})}{P(\bar{w}|c_{neg})} \cdot \frac{P(w|c_{pos})}{P(\bar{w}|c_{pos})} \cdot \frac{P(c_{neg})}{P(c_{pos})}\right] = \ldots = \\ &\approx pres_w + abs_w \end{aligned} \tag{1}$$

where $c_{pos}(c_{neg})$ are the proportions of observed positive (negative) comments whilst $pres_w$ and abs_w are the log-likelihood ratios of the events $(w \in r)$ and $(w \notin r)$, respectively. Likewise, for the set of J words included in a comment c, the log-odds ratio of c is defined as:

$$\begin{aligned} LOR(c) &= \sum_{w_i \in J} pres_{w_i} + \sum_{w_{i'} \notin J} abs_{w_{i'}} = \\ &= \sum_{w_i \in \mathcal{J}} (\log P(w_i|c_{neg}) - \log P(w_i|c_{pos})) + \\ &\quad + \sum_{w_{i'} \notin \mathcal{J}} (\log P(\bar{w_{i'}}|c_{neg}) - \log P(\bar{w_{i'}}|c_{pos})) + \\ &\quad + \log P(c_{neg}) - \log P(c_{pos}) \end{aligned} \tag{2}$$

While calculating those values for all the w ($w \in BoW$) words, it is possible to obtain the values of c_{pos}, c_{neg}, $pres_w$ and abs_w for each comment. We have then used cross-validation to estimate a parameter τ such that: c has been classified as "negative" if $LOR(c) > \tau$ or as "positive" if $LOR(c) \leq \tau$.The selected value of τ is that minimizing simultaneously both the Type I and the Type II errors.
The same approach is used for the set of K words composing a review r, thus computing $pres_{w_i}$ and $abs_{w_{i'}}$ for all the words appearing and not appearing in a review, and comparing $LOR(r)$ with the value of τ obtained from the classification of the comments into "positive" and "negative".

3 Results

It is essential to assess the performance of the model, especially for an unlabelled context. To achieve that we train the model with some labelled data, but without providing the real label to the model. For this purpose we have used TripAdvisor text data while considering the associated number of stars [1-5] to be the negative [1-2] (or positive [3-5]) real label.
We could just use directly the estimated label, created with the sentiment score described in Section 2.2. But, we demonstrate that, training the Naïve Bayes* model with such an estimated label, allows to achieve more reliable results.
For this purpose, we have calculated the most used performance indicators using the real label that was not given to the model: Misclassification Error, Accuracy, True Positive Rate, True Negative Rate, F1-score, Matthews Correlation Coefficient, Bookmaker Informedness, MarKedness.
As it is possible to notice in Table 1 and Table 2, the Naïve Bayes* classifier presents a good performance while classifying a comment into positive or negative with a numerical value. What actually it is even more interesting, compared to other kind of approaches, are the LOR values, that the model must estimate in order to being able to produce the classification.

Table 1: estimated label performance on predicting the real label.
Notice that no label was used for the training phase, but the text data instead.
Performance estimated with 10-fold CV

ME	ACC	TPR	TNR	F1	MCC	BM	MK
0.092	0.908	0.936	0.398	0.951	0.268	0.334	0.215

Table 2: Naïve Bayes performance on predicting the real label.
Trained with the estimated label. Performance estimated with 10-fold CV

ME	ACC	TPR	TNR	F1	MCC	BM	MK
0.055	**0.945**	**0.973**	**0.503**	**0.973**	**0.475**	**0.476**	**0.474**

4 Conclusions

According to what we have shown before, those values have a "versatile nature", in fact they can be summed together matching a certain criteria. The criteria that the model use for merging words' values is, for a given set c ($c \in BoW$), to check which word belongs (or not) to c. While using the same LOR values with the same approach, but a different criteria, it is possible to apply a dimensionality reduction technique for produce some valuable and interesting plots.
Natural language text data that are taken form the web, usually comes with temporal information (data only or time too), so a time-series of the LOR words values will take full advantage of that.
In fact, using a semi-supervised clustering methods approach over a Word Embedding representation of the words considered in our BoW, it is possible to create some categories of interest. We consider every category to be a centroid of a fixed number of clusters representation in that Word Embedding space, where the number of clusters is the number of categories. To conclude, all the words inside a certain cluster-category will be mapped with the corresponding category. It is then possible to calculate how the LOR for a certain category change in time. In fact a single category can be considered to be a comment composed by all the words that are mapped to such a category.

References

[1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. pages 3111–3119.

[2] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. European Language Resources Association (ELRA).

[3] Maurizio Romano, Luca Frigau, Giulia Contu, Francesco Mola, and Claudio Conversano. Customer satisfaction from booking. In *Selected paper GARR_18 Data (R)evolution*, pages 111–118. Consortium GAAR. ISBN 978-88-905077-8-6. DOI: 10.26314/GARR-Conf18-proceedings-21.