NLP, Sentiment Analysis

Pablo Donís Ebri (p.donis@alumnos.upm.es)

Link to the Github Repository:
https://github.com/pablodonisebri/IntelligentSystems_NLP

1. Introduction

In this work the main objective is to implement a very basic and naïve tool for detecting the main sentiment coming from a short and simple text. For doing so, we thought that reviews and comments of Amazon products would be the best option, as they are short enough and they show the emotion of the client with respect to the product (whether it is a good product and the client likes it or the exact opposite).

After some research in the web, a dataset that suited our needs was found (the file containing the data is stored in the repository containing the code and files is linked above). Also, some tools shown in the HandsOn by Raul were used in order to achieve the goal. Just to mention that the entire work was made with R and Rstudio.

The implemented technique is as basic as counting the positive words and the negative words found in the text and seeing whether the text has more positive ones or negatives, in the first case the text would be classified as positive and in the second as negative. We will see that it works in many cases but not in all.

2. Methodology

2.1 Data Exploration

The data comes in a csv file containing 34660 observations of 21 variables. The variables containing all the information of the comments made in Amazon in relation with the product purchased, but we are just interested in the product that the comment is about, the title of the comment (will be used as the title of the document in the corpus later on) and the content of the comment itself. We stored these three variables in a Dataframe format for easier use of it. The dataframe needed to be in a specific format for using it with the lexicon of sentiments and building a corpus with it, so we renamed the columns of it to match the specifications. For the work we used getSentiments from the package tidytext in order to have a list of positive words and another list of negative words.

2.2 Implementation of the tools

We built three functions for achieving the objective, they are:

- *prepareText:* An auxiliar function for getting the text into a cleaner form for the analysis (as Raul does in the HandsOn), using tm_map.

- *get_sentiments_from_text:* This function returns the count of positive and negative sentiments from the document in a dataframe format. For this purpose, the function first calls *prepareText and then* tokenizes the text, then matches the tokens with the lexicon of positive and negative words and makes the count of both. The function contemplates a special case that occurs when no positive or negative words are found in the text. This case needed to be considered apart because otherwise the returned values were incorrect (positive count and negative count were equal although one should be zero).

- *sentiment:* This function calls *get_sentiments_from_text* in order to get the count of positive and negative and makes the difference, if the count for positive is greater, it returns "positive", in the case that the count for negative is greater it returns "negative" and in case they are the same, it returns "neutral".

3. Results

Once the tools were ready, we could start analyzing the comments. From the 34660 we got 30058 positive comments, 3303 neutral and 1299 negative ones.

In the R script there are shown examples of positive, negative and neutral comments, and there are also shown some that are misclassified. Let´s look into detail at the the comment misclassified that is shown in the script and see where the mistake is:

- We look now at the comment that has been classified as negative although the meaning of the comment tells us that the client likes the product, so it is a positive comment.
  The content is: "I bought 3 tablets and my family was not disappointed."

  If we call the function *get_sentiments_from_text* with that text we get positive 0 and negative 1. We know that the word "not" is removed because is a stopword and therefore is removed in the calling of *prepareText* and we are left with the word disappointed, which the lexicon considers as negative therefore the negative 1.
  So, the mistake comes when removing the negation of the word with negative implications. That mistake is difficult to catch by a naïve classifier like the one we implemented but should be taken in consideration when trying to implement a more refine classifier.

We could not detect in this corpus any case in where happened the opposite, that a positive word came with a negation in front so its implications are negative (for example: this device does not work properly), but our classfier will count it as positive. But if a text came presenting that, it would be misclassified.

Apart from that, if neither of those patterns presented (a positive/negative word with a negation in front) the classifier does a good enough job in classifying, specially in comments of products as they clearly show an opinion/emotion related to the product.

Please take into account that the script might take some minutes in the calling of classifying the comments as they are too many.

4. Bibliography

1. García, R. (2017). Intelligent Systems. Hands-on 1.RPubs, website: https://rpubs.com/rgcmme/IS-HO1

2. Consumer Reviews of Amazon Products. (2018). Kaggle, Website: https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products?select=1429_1.csv

3. Silge, J., & Robinson, D. (2020). Introduction to tidytext. Cran-r Project, website: https://cran.r-project.org/web/packages/tidytext/vignettes/tidytext.html