Natural Language Processing: Reviews satisfaction

Marina Prieto Pech[1*†]

[1*]Machine Learning, UCLM, Paseo de la universidad, 3, Ciudad Real, 13002, Ciudad Real, España.

Corresponding author(s). E-mail(s): Marina.Prieto1@alu.uclm.es;
[†]These authors contributed equally to this work.

Abstract

This work studies the use of nautral language processing as a way of applying supervised learning algorithms for predictions.

Keywords: Natural Language, Processing, Machine Learning

1 Introduction

For this task, it was required to review and evaluate the scores given to certain products in a platform of e-commerce. Objective was to successfully predict the given scores (in between the values 1 to 5) by taking into account the text description the users gave to the products.

2 Dataset Description

On this dataset we can observe the following variables:

- **ID** (type: int): Identifier of the review
- **ProductId** (type: string): Product stringified identificator, which was a code including numbers and letters.
- **UserID** (type: string): ID of the user writing said review.
- **ProfileName** (type: string): Name with which the user gets identified by in the platform.

- **HelpfulnessNumerator** (type: int): Number of users that have found the review useful.
- **HelpfulnessDenominator** (type: int): Total number of users that have reviewed the review by classifying it into useful or not useful.
- **Score** (type: int, ranged): Score given by the reviewer from 1 to 5.
- **Time** (type: int): Time at which the review was posted.
- **Summary** (type: string): Summary of the review, short description.
- **Text** (type: string): Complete review given by the user.

3 Starting steps

First of all, we need to import and load the needed dataset, this being the dataset **products.csv**. We will install the basic libraries and tools and include some source code to select and import the dataset properly.

We also need to take note about this file not being properly formated, having to remove some extra characters on the starts of certain lines, as well as removing semicolons and getting rid of the "ghost columns".

To finalize this part, we create a products dataframe including only the necessary values: **Summary, Text and Score.**

4 Preprocessing

For this, we need to be aware this dataset contains mostly string values, which means the pre-processing is going to need different ways of working with it. This are the steps taken:

4.1 Removing useless characters

First of all we are going to take away all the characters that are considered unnecessary for the review's semantic value. We do this, with a function that detects those characters and applies the removal on both the summary and text, but in separate ways.

We will see this is the overall idea for the rest of the reprocessing steps.

4.2 Replacing capital letters

For this we just need to check every data frame entry and use a native pythons function: .lower(). Doesn't require more processing for it to work properly.

4.3 Removing contractions

This can be done by creating a list of patterns, following the contraction and how it would be not contracted.

For reference, we use the following contractions as base of our processing:

After clearing that up, we just create two methods to go word by word replacing the instances of those words with their equivalent.

4.4 Removing repeated words

This method handles the words one by one, removing in case it encounters an exact instance on the entry.

4.5 Removing emoticons

For this, we create a code capable of removing the characters on the entry we are evaluating that counts as emoticons. This is done thanks to the method emoji.is_emoji(char)

4.6 Lemmatizing terms

As last step, we need to lemmatize all the terms we have on the data frame. For this we can use the method itself, followerd by a splitter to keep the words together after the processing.

5 Vectorization

Since all the data has been processed with the previous steps. The next step is the vectorization of the data. For this, we're going to divide the proccess in 4 steps:

5.1 TFIDF

Term Frequency - Inverse Document Frequency measures the relevance that a word has in a document.

For this, we take into account the number of times a word appears in a document and the total number of words in the document.

For this processing, the algorithm available in the colab file should be followed.

5.2 TFIDF + N-grams

Next step is including N-grams to our dataframe results.

They are continuous sequences of words, symbols or tokens in a document.

So now, we establish the TFIDF for a range of N-grams with terms of different sizes.

5.3 TFIDF + N-grams + POS tagging

We assign the grammatical category to each word after the process. POS tagging bases this category on the definition and context of the word

5.4 Other features

Lastly, to go a step further, we take into account the number of words and number of sentences each review has, this gives us off more precision on the vectorization.

6 Feature selection

For the features selection, we're going to apply the selectKBest method and removing the 70% of our dataset values.

We repeat the process for every Vectorization step in order to see the progress of it.

7 Classification algorithm

To finalize this project, whats left to do is training the algorithm, so it can do the score predictions and overall the results.

For each of the steps done in the vectorization (TFIDF, N-grams, etc...), we apply the SVM (Support Vector Machines) training method.

The results of each confusion matrix generated by the algorithm, relating the predicted values to the real ones are available throught the colab file.