Comments on Purchases: NLP and Classification

Carlos Pulido Hernández[1], Ismael Pérez Nieves[1] and Sergio Jiménez Fernández[1†]

[1*]Machine Learning Techniques, Escuela Superior de Informática, Ciudad Real, Castilla-La Mancha, Spain.

Contributing authors: carlos.pulido@alu.uclm.es; ismael.perez3@alu.uclm.es; sergio.jimenez19@alu.uclm.es;
[†]These authors contributed equally to this work.

Abstract

For the practical part of the Machine Learning Techniques subject, about Natural Language Processing, the task consists on applying different NLP techniques that we have seen in class on the collection of opinions about drinkable and edible products and classify them in 5 different levels depending on the degree of satisfaction. For this purpose, two classification algorithms were tried for several input data schemas.

Keywords: Machine Learning, Natural Language Processing, Artificial Intelligence

1 Problem description

For the laboratory part of Machine Learning Techniques subject, we were asked to obtain the best classification model that, given an opinion with a brief summary of it, can predict the satisfaction score that the user has assigned to it (from one to five).

Aiming to solve this task, we were told to use Natural Language Processing (NLP) techniques and training data configuration, as well as at least two different classification algorithms.

2 Methods and materials

For our approach, we first did several preprocessing steps. In this process, we included many NLP issues: remove useless characters (e.g. '[', '(', '$', ...), remove capital letters, useless tokens (specifically HTML tags and their content), expand contractions, remove possible emojis, remove numbers and remove stop-words.

It is important to remark that, we also tried to correct some spell errors or *typos* but due to its high time cost, it is not used for the final version.

When the first iteration for the preprocessing was done, then we divided the sentences by means of blank spaces or punctuation marks, and then, lemmatize all the remaining vocabulary.

Once the data is correctly preprocessed following the NLP techniques, we proceeded to divide all the available documents into training and testing sets. For the training set, we left a 70% of the document dataset for training purposes and the remaining part for testing.

At this point, with a fully preprocessed compilation of documents, we tried four different strategies for designing the input data:

- **1:** *TF-IDF vectorization*: The very basic version.
- **2:** *TF-IDF vectorization and n-grams*: More specifically bi-grams, since with higher n-grams and even joininig bi-grams with other data, we ran out of resources and was impossible to move on with the models.
- **3:** *TF-IDF vectorization and POS tagging information*: For the POS tagging, we decided to use the number of adjectives in each document, since in an opinion, we consider that these words are quite representative.
- **4:** *TF-IDF vectorization, POS tagging and extra information*: Basically, the same as in the second point but adding some extra features that we extracted from the opinions, i.e., the number of words and the number of sentences.

After the vectorization we filtered the features obtained from this step, getting a 30% of the total set of features as final feature selection.

Finally, we gave those final vectorization tables obtained from the previous step as input to three different models: (i) Naive Bayes with a Multinomial approach; (ii) Decision Tree; (iii) Voting Classifier, with kNN, Decision Tree and Naive Bayes given as weak estimators.

3 Experiments and results

The final results for the models of each strategy explained before are detailed in Table 1. In this table, we indicate the F1 Score for each model of each strategy. All models have been executed with undersampling to polish the consequences of the imbalanced data, excepting the models of strategy S2, due to memory shortage reasons.

As we can see, according to our approach, the best model is achieved with strategy S1 (just TF-IDF vectorization. In order to expose the results with another point of view, check Figure 1 for a collage with the confusion matrices

Table 1 F1 Scores evaluation per strategy and model.

F1 Score	Naive Bayes	Decision Tree	Voting Classifier
S1	0.6486	0.6067	0.6218
S2	0.6154	0.6271	Not Possible
S3	0.6390	0.4725	0.4833
S4	0.6212	0.4759	0.4738

of the best models of each strategy (marked in red in Table 1). Note that all of them were obtained by applying undersampling, excepting the strategy S2, like Table 1.

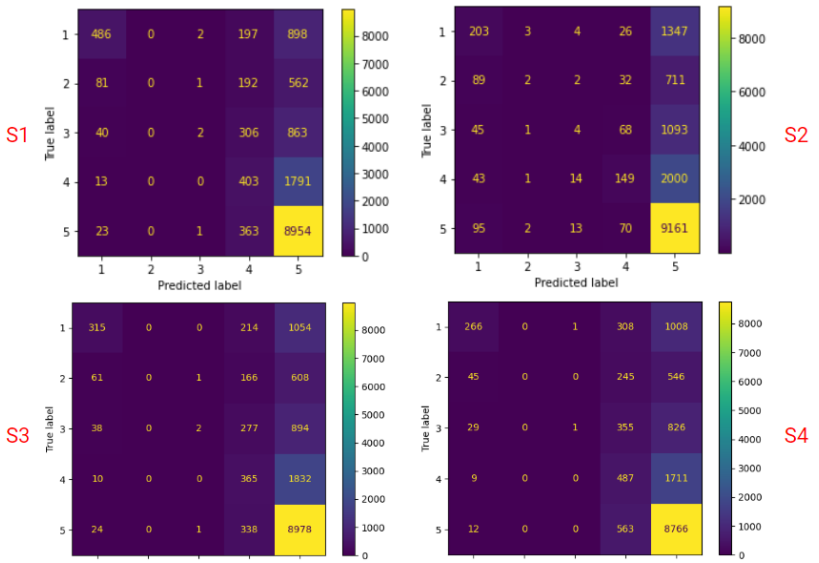

Fig. 1 Confusion matrices for best model in each strategy

4 Conclusions

For concluding the report of this project, we will analyze the results exposed in the previous section and try to give a final idea that sums up the whole development.

Taking a look to Table 1, the best models are Naive Bayes (especially) and Decision Tree. This demonstrate us that our try of improving the accuracy of our solution by using an ensembling model was not the answer. Anyway, we expose the process since we consider an interesting point to take into account.

Relevant to Table 1 too, we can also highlight that the best scores are near to a 60%. In order to improve it, we think that a solution could be trying to mitigate the effects of the imbalance in the data with more detail.

Finally, analysing the confusion matrices from Figure 1, we can outline that, in spite of our efforts in reducing the effect of the imbalance, it still has a big influence on the models. We can see this issue just by checking any column in the matrices excepting the fifth one: the number of hits is nearer to the number of errors.

Appendix A Google Colab Notebook

All this study was developed mainly in Google Colab Notebooks. The project development was divided in several notebooks following the next schema:

- NLP Preprocessing:

  https://github.com/SergioJF10/MLT-ESI-UCLM_CIS/blob/main/products/Notebooks/NLP/NLP_products.ipynb

- S1 - TF-IDF:

  https://github.com/SergioJF10/MLT-ESI-UCLM_CIS/blob/main/products/Notebooks/Models/1-TF_IDF.ipynb

- S2 - TF-IDF + N-Grams:

  https://github.com/SergioJF10/MLT-ESI-UCLM_CIS/blob/main/products/Notebooks/Models/2-TF_IDF_and_N_Grams.ipynb

- S3 - TF-IDF + POS:

  https://github.com/SergioJF10/MLT-ESI-UCLM_CIS/blob/main/products/Notebooks/Models/3-TF_IDF_and_POS.ipynb

- S4 - TF-IDF + POS + Extra:

  https://github.com/SergioJF10/MLT-ESI-UCLM_CIS/blob/main/products/Notebooks/Models/4-TF_IDF_POS_and_Extra.ipynb

  https://colab.research.google.com/drive/1_UwZwxy9QoqjKxOEQcmz_u--pMdEuadf?hl=es#scrollTo=wea_JmAU2bN0&uniqifier=3

Appendix B GitHub Repository

In the *products* folder of the GitHub Repository, you can find all the pertinent source code and considerations involved in the development of this project.
https://github.com/SergioJF10/MLT-ESI-UCLM_CIS