Bag-of-Authors
Text Mining Project

Ana Rita Marques, Ana Sofia José, Pedro Alves
R2016700, R2016718, R2016734

1. Introduction

The main goal of the project is to develop a machine learning algorithm capable of **identifying the author of a given text excerpt with 500 or 1000 words applying NLP techniques**. For this purpose, we were given a non-balanced corpora containing several books from different portuguese authors - Almada Negreiros, Camilo Castelo Branco, Eça de Queirós, José Rodrigues dos Santos, José Saramago e Luísa Marques Silva -, and text excerpts with no information on the author that should be correctly predicted at the end.

2. Method/Approach

2.1. Pre-Processing

A step we consider to be an informed dimensionality reduction since parts of the provided texts aren't relevant for our analysis, was to remove by hand directly on the txt files the final notes, editor and scan notes, the word "fim", the prefaces and the index that appeared across the train texts as they are not exactly part of the text, they are information on the book itself which will not appear in our test samples. Then, we continued to pre-process our data following the steps below:

i. Lowercase our text;
ii. Remove tags, line breaks, multiple spaces and accent marks from letters;
iii. Replace numbers with a token "NUMBER";
iv. Replace every word that has an apostrophe with the respective word plus a token "APOSTROPHE";
v. Replace the special characters "!" and "?" with a token "EXPRESSION";
vi. Remove all the other punctuation and/or stopwords (optional);
vii. Apply stemming (optional).

The reason why we perform some of the steps are evident, but others might not be so clear. We decided to replace numbers with the token "NUMBER" as we believe that the reference to digits along the text can be a characteristic of some of the authors. The same happens with the apostrophes and the special characters "!" and "?" that can be important features to consider as they are an evidence of different writing styles. From our point of view, apostrophes can be a reference to older writing or relaxed and practical writing while the specified characters may be a reference to an enthusiastic writing and can be a distinctive characteristic that contrasts with authors like José Saramago.

This process was developed inside a function in order to allow an easy experiment with different combinations of parameters, e.g. apply NLP techniques over text that have stopwords to see if our model performs better or not.

2.2. Feature Engineering & Unbalanced Small Dataset

In order to try to improve our model's performance, we created some features that we thought would make sense:

a. **WordsPerSentence**: Average number of words per sentence
b. **Sentences:** Number of sentences normalized by the total number of words (counting with stopwords and punctuation)
c. **UniqueWords:** Number of unique words (counting with stopwords) normalized by the total number of words (counting with stopwords)
d. **ExpressionSentences:** Ratio of expressions ("!" and "?") per sentence

The aim of these features is to **capture some specific author characteristics**: longer or shorter sentences (a), vocabulary richness (c) and expressiveness (d).

Since our corpora has a small number of observations and is unbalanced in terms of number of texts per author and text size, we designed and implemented a **function to get more text samples given a specified number of words and a multiplier considering a balanced or unbalanced training set**. Since we believed a balanced training could be more interesting, we decided to take the same amount of samples from each author's texts as partial texts with a prespecified number of words. To achieve it, we implemented a least common multiple of each author's quantity of texts dividing it by the number of texts available per author (LCM equals 360 samples in this training case). The implemented multiplier allows us to multiply the LCM, to increase the number of samples. A second problem is the text's length, smaller or larger than the required sample size. Our method selects a random word on the text and collects the next requested number of words. If we reach the end of text, we wrap it up from the beginning.
This allows us to increase the number of observations and decrease or increase our texts length, improving the performance of more complex algorithms such as neural networks, that work better when more observations are available or KNN having a more dense cloud of observations for each author text. We are aware that this generates some bias as we can have overlapping samples that give redundant and duplicate information to the model, but we think that models will benefit more than if we gave them unbalanced data and very few observations.
When we use this function, our features (2) and (3) are transformed, respectively, into number of sentences and number of unique words (counting with stopwords) as they do not need to be normalized because our text samples have equal sizes.

2.3. Baseline

Our baseline is composed by the specified pre-processing, a bag-of-words (BOW) language model with a binary encoding and n-grams up to 3 and the KNN algorithm with 7 neighbors. In this first phase, the observations used were the original 63 and the results obtained can be seen in the appendix A.

This **simple baseline** already shows a good performance with **92% accuracy** on the validation set. Only Eça de Queirós' text wasn't assigned correctly. However, with so few observations on the validation set, these results can be misleading because the model might be worse when generalizing, and just got "lucky". Also, our training and validation observations are composed by entire texts which we consider as not advisable and not totally correct as we will predict only parts of texts and this can lead to poor results when applying the model on a score set. Thus, we will explore other approaches to get more reliable results than this simple baseline.

2.4. Pipeline & Models

To simplify a testing approach, we organized our code in functions, and, at the end of the python file, we designed a pipeline with multiple hypotheses for each phase of the NLP modelling that helped us to conduct our tests and tune the hyper parameters.

We also developed and implemented the sample generating function, the TF-IDF language model, a neural network with 3 layers and a multinomial logistic regression perceptron, besides the BOW, KNN and the aforementioned extra features with the expectation of improvements.

It should be noted that when applying the train test split, we are fixing the seed but the sample generating process is random, allowing us to do a cross-validation like process in order to analyze the robustness of our models when run several times with the same parameters. Also, in the train test split (0.8/0.2), we used the stratified parameter that assures us an equal distribution in the train and validation as in the original set, regarding the target.

3. Results & Discussion

We tested the **multinomial logistic regression perceptron (MLRP)** with softmax combining different parameters and our best MLRP model[1] got an **accuracy** of 100% on the validation but it always took a long time to run (more or less 1 hour), so we decided to try a different approach.

Using our sampling function in a **simple KNN**[2] with 7 neighbors, cosine distance and BOW, getting 3456 training observations and 864 validation observations, we got an **accuracy of almost 100% in the seven times** we ran it, always failing in 1 of the 864 observations predicted, except in two perfect runs (appendix B). We tried this model without stopwords, with stemming, with the extra features and combinations of these and other parameters, but the results were always worse. This might seem strange and the most senseful conclusion we can make is that our preprocessing became the most important piece of our puzzle.
The KNN is a simpler and easier to interpret model, that at first, we thought was not the best to use in this case study but turned out to be fast and present great results.

We also implemented a **Neural Network** composed of a dense layer (activation=relu), a dropout layer and an output layer (activation=softmax), optimizing accuracy using "adam" as optimizer and categorical cross-entropy as the loss function. The results were promising, but we ran into some memory problems that prevented us from exploring this algorithm in more detail. If we had a higher computational power this would be an algorithm to work on, however, as we reached great results with a simpler model, in our case it would not be necessary.

In our case, applying TF-IDF language modelling on our machine learning algorithms always drove us to worse results (e.g. 60% of accuracy on MLRP, 99% of accuracy on KNN failing 12 observations) than using BOW, so we discarded the TF-IDF option when tuning our models.

It might be important to notice that while our validation set presents a high accuracy, so does the train, which could be a sign of overfitting, and our sampling method could have leakage from training to validation. But the fact is that at the end, we ran our best model (the KNN) and the predicted labels matched the true ones (appendix C) - found after some searching. In the appendix D you can find the labels predicted by our KNN.

4. Conclusion

After diving into three different algorithms and trying several combinations of parameters, we found that the KNN algorithm was the best in predicting the author for each text sample. If we have a simpler model, easier to interpret that does the work, there is no need to go for a more complex one.

As we found, pre-processing is a fundamental part of an NLP project and can really make a difference on the performance of the machine learning algorithm, thus we should spend time to understand our corpora characteristics and what type of problem we want to solve in order to get the best results possible. This may be a time consuming but worthy task that should be carefully handcrafted with reasoning.

[1] Balanced sampling with 1000 words and multiplier of 2; base cleaning removing stopwords and applying stemming; Bag of Words with n-grams up to 3 and binary counts; without extra features; MLRP over 1 epoch

[2] Balanced sampling with 1000 words and multiplier of 2; base cleaning maintaining stopwords and not applying stemming; Bag of Words with n-grams up to 3 and binary counts; 7 neighbors; without extra features

References

Brownlee, J. (2016, June 1). Multi-Class Classification Tutorial with the Keras Deep Learning Library. Machine Learning Mastery. https://machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library/

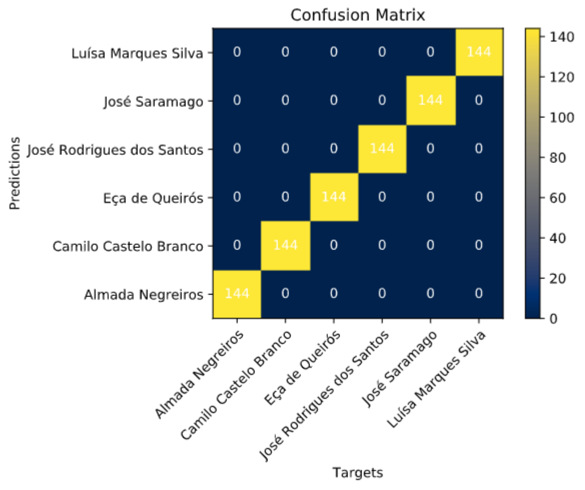
Python, R. (n.d.). Practical Text Classification With Python and Keras – Real Python. Retrieved April 9, 2020, from https://realpython.com/python-keras-text-classification/

Appendices

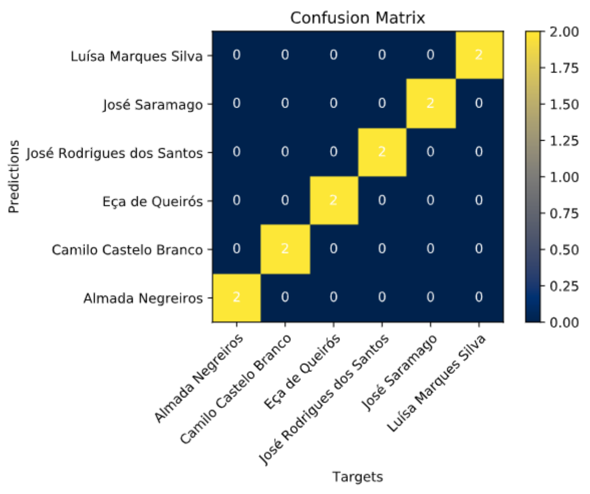
A – Baseline Performance

	Precision	Recall	F1-Score	Support
Almada Negreiros	1.00	1.00	1.00	2
Camilo Castelo Branco	1.00	0.8	0.89	5
Eça de Queirós	0.00	0.00	0.00	0
José Rodrigues dos Santos	1.00	1.00	1.00	2
José Saramago	1.00	1.00	1.00	2
Luísa Marques Silva	1.00	1.00	1.00	2
Accuracy			0.92	13

B – Validation Performance with the Final Model



C – Test Performance with the Final Model



D – Final Predicted Labels Table

Text Name 500 & 1000 words	Label
Text 1	José Saramago
Text 2	Almada Negreiros
Text 3	Luísa Marques da Silva
Text 4	Eça de Queirós
Text 5	Camilo Castelo Branco
Text 6	José Rodrigues dos Santos

The original table can be seen in the "Predict Test Files Authors" section in the python file.