Identifying Authors by Their Writings

Inês Diogo, Lara Neves, Susana Paço
20190301, 20190867, 20190821

1 Introduction

The goal of this project is to develop models capable of learning the author of Portuguese book excerpts.

Our training set is composed of a set of several labeled books written by six different authors, Almada Negreiros, Camilo Castelo Branco, Eça De Queirós, José Rodrigues Santos, José Saramago and Luísa Marques Silva. This is not a homogeneous training set, since the number of excerpts for each author is different and the number of words in each excerpt is also diverse.

The test set has unlabeled text excerpts with 500 or 1000 words, which are the ones we will attempt to predict with our models.

2 Method/Approach

2.1 The presence of Metadata

Before beginning the preprocessing of the data, a visual analysis was made in which metadata was detected at the beginning of the texts. We hypothesize that this can lead to a higher bias in our models and as such we decided to remove it from the corpora and test if its presence made a significant difference in the results of our models.

Most of the metadata consists of authors' names, references to the authors' work, such as name, edition number, year and so on.

2.2 Preprocessing the Data

Our first action was to create a standardized form to identify each excerpt and their respective authors, making sure that there would be no duplicate ID and changing the author identification to their siglum for easy use.

In order to remove the metadata, we discarded any mentions of the authors' names and published works while making sure to include all possible variations of their names.

We wanted to test how different levels of preprocessing could affect – positive or negatively - our baseline model as well understand the importance of the presence of punctuation given that one of the authors, José Saramago, is known for abusing its use which could be a factor in identifying his texts.

Therefore, three functions for preprocessing the text were created; two to be able to distinguish the difference between keeping or removing the punctuation and a third one to completely "clean" the texts.

The order decided for the preprocessing was to lowercase, remove the punctuation, remove the stop words - with both *nltk* and *spacy* packages given that they contain different stopwords for Portuguese and so should complement each other - to lemmatize and finally to remove the accents marks.

Removing the accents was deliberately done at the end of the preprocessing, because some words without accents would not be considered as stopwords, and therefore they would not be removed nor lemmatized.

When we went into comparing lemmatization with stemming, we quickly realized that stemming (in its available functions in nltk and spacy) produced nonsensical words in Portuguese such as transforming "almada" into "alm". Thus, to ensure a more coherent text, we decided to move forward with lematization leaving stemming behind.

Finally knowing that the training set is unbalanced in favor of Camilo Castelo Branco, and the need for more data, particularly for the neural network, we divided all texts into 500-word chunks. We also made sure to include the *F1 score* as an evaluation metric beside *Accuracy*, as it integrates both *Precision* and *Recall* and is particularly informative for unbalanced datasets.

2.3 Brief Analysis

Now that we have cleaned the texts there are several analyses that can be made.

First, we realized that the style of Portuguese present in the texts can vary as the authors belong to different time periods, which should reflect in their use of vocabulary and grammar. We payed special attention to what we call contemporary authors - José

Rodrigues Santos for example – and older authors like Camilo Castelo Branco.

To see evidence of this we developed word-cloud visual analysis of the most frequent words written in each book as seen in figures A1 and A2, first from one contemporary author and the second from an older author.

Additionally, we thought it would be interesting to visualize the most frequent words in the entirety of our corpora, bar plot A3.

To test the importance of the punctuation, referred on page 1, a dummy classifier was implemented on two texts from Saramago and two texts from José Rodrigues Santos 50 000 times and the overall difference between the classifier score with and without punctuation was determined.

If the classifier is better with punctuation a positive difference should be observed, meaning that the classifier that runs on the text with punctuation should have a better accuracy score than the one without.

We observed a very small change between each classifier. To confirm this, a one-sided hypothesis test was constructed, where the null hypothesis is that the mean of the difference between the scores is zero. With a mean of -0.0021 and a $p - value = 0.1792$ we fail to reject the null hypothesis and thus we can conclude that there is evidence that shows no differences in the performance of the classifier of JS with or without punctuation.

2.4 Creating the Baseline

We started by dividing our training set into train and validations sets. This is particularly important as it allows us to inspect the model performance and to adjust the hyperparameters as the model trains. Given the relatively low amount of data, particularly before we divided the excerpts into chunks, we chose to implement a k-fold crossvalidation strategy. The choice of the number of folds was 10 in accordance with (Gareth James, 2013).

In order to compare our models, it’s fundamental to create a naïve baseline model. To do so we started with a Dummy Classifier inside the K-folds that we found to be far too simplistic to be of use. Following that we implemented a Bag-of-Words classifier inside the k-folds.

Our goal was to use the Bag-Of-Words baseline to test the different levels of preprocessing.

2.5 Creating the Models

Finally, two models were developed *FastText* and *Long Short-Term Memory (LSTM)* neural network architecture.

We chose *FastText* in particular due to the fact that its architecture is based on the Continuous Bag of Words but more efficient while still on par with deep learning classifiers (Mikolov, 2016).

The fact that the Portuguese language development lead to a language where the semantics of a word are taken usually within the context of that word while it can adopt many semantics differing in the context in which it is written, informed our choice of using a CBOW based architecture rather than a Skip-Gram model for example.

Finally, an LSTM based architecture was created to check if it could develop improved results. The structure chosen was a simple one, based upon the concepts defined by Hochreiter￼ (Hochreiter, 1997)￼. This LSTM architecture as the possibility of allowing us to surpass some of the limitations associated with a small corpus and the overfitting problems this dataset is prone too. The corpora underwent the same pre-processing as the other models for clean punctuation, removing metadata and lowercasing and then proceeded through a process of tokenization in order to create a proper input for the LSTM recurrent neural network.

3 Results and Discussion

3.1 Results

3.1.1 Baseline Results

As expected, the first baseline created, the Dummy Classifier Baseline, had worse results for the accuracy and f1 score than the second baseline, the Bag-of-Words baseline.

Observing the tests with the Bag-of-Words Baseline for the different levels of Preprocessing, we can observe that cleaning the data does improve the scores of the training set and the evaluation set for both, the *accuracy* and *f1 score*. As we can see on table A4 and graphs A5 and A6.

3.1.2 FastText Results

Given the nature of this model and after several tests in order tune the hyperparameters, the final implementation of the model was based on a 90% training set, 10% validation set split, rather than crossvalidation, as well as using the most complete level of preprocessing – which we determined in the previous section to be the best, while also dividing the excerpts into 500 word chunks.

The hyperparameters with the best results consisted of 300 epochs (*epochs*) with a tokenization based on a minimum of 2 (*min_count*) and with sequences with maximum length of 100 (*maxlen*).

This resulted in an accuracy and F1 score equal to 1 and average loss of 0.00247 for the training set and for the validation set an accuracy of 0.98181, F1 score of 0.98137 and loss of 0.09846.

Implementing the model on our test set we predicted the authors for those texts in comparison with what we think is the ground truth based on some literary research. The result was 50% correct predictions as seen in table A7.

3.1.3 LSTM based architecture results

As the corpora given is a tad bit small, a simpler architecture was created to avoid as much as possible the common problems inherent to neural networks such as the vanishing gradient. 3 layers were created, an embedding one, with 64 neurons, an LSTM layer with also 64 neurons and a dropout of 0.4 and an output layer of 6 neurons, one per each author. All of these parameters were chosen by analysis of similar problems such as Zhou et al 2015 and creation of an adapted structure for our problem.
It attains an average loss of 0.005677 and an average accuracy of 0.99878. As we know that the model is severely imbalanced, it requires some in depth analysis to assess if the accuracy values are not misleading.

As we can see in table A8, this model attains an average an 8 out of 12 texts properly classified being able to be right 2/3 of the time ('prediction' being what the model predicts and "possible result" our own research of the text). It is noticeably better for old Portuguese authors than younger ones, being unable for example, to identify correctly any Luísa Marques Silva (LMS) text. Further texts with more corpora and a more balanced dataset are a good idea to further develop this model.

3.2 Discussion

From what we can analyze in results we can conclude that a proper pre-processing does matter in the quality of the corpora that we can feed the models. Baseline differences in our tests with different preprocessing protocols are evidence of this. These results are valid for the Portuguese language but might not translate to other languages.

Our baseline is a very simple structure, yet it attained remarkable results, going until ~0.3 in accuracy in some of our protocols, demonstrating to us the importance of a proper pre-processing.

However, for such a small corpora and imbalanced dataset, we needed to quickly move on to more complex architectures and required the aid of neural networks to complete this task. From our two attempts the FastText architecture came to predict right an average of 6 out 12 texts from the text set while the LSTM architecture was able to reach 8 out of 12. This can be attributed to the Fast Text architecture being more sensitive to imbalanced datasets than the LSTM. However, both architectures show promise and should be attempted with bigger and more balanced datasets.

4 Conclusion

Authorship identification in a language other than English is not an easy task. Many obstacles appear, both in preprocessing text without the appropriate functions and in using models previously tested and created for the English language. However, our results show that good accuracy can be attained and there's a promise of proper authorship identification in the Portuguese language given proper corpora with a sizeable size and a balanced dataset. The differences between old Portuguese (pre 20th century) and new Portuguese (post 20th century) are noticeable in the models and mechanisms to help NLP professionals differ these two types of Portuguese are research milestones worth to be done.

All the code of this project is in the following github: https://github.com/theinsilicobiology/TextMiningFinalProject

References

Gareth James, D. W. (2013). An Introduction to Statistical Learning with applications in R. Springer.

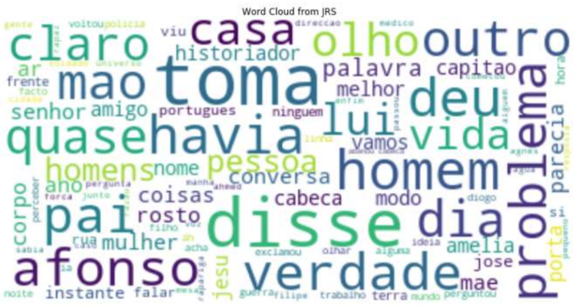
Hochreiter, S. a. (1997). Long short-term memory. *Neural Computation*, 9(8).

Mikolov, A. J. (2016). Bag of Tricks for Efficient Text Classification.

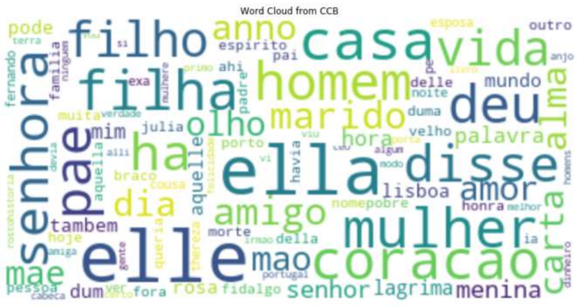
Molapo, C. S. (2000). Automated text scoring, keeping it simple. (I. R. Institute, Ed.)

A Appendices

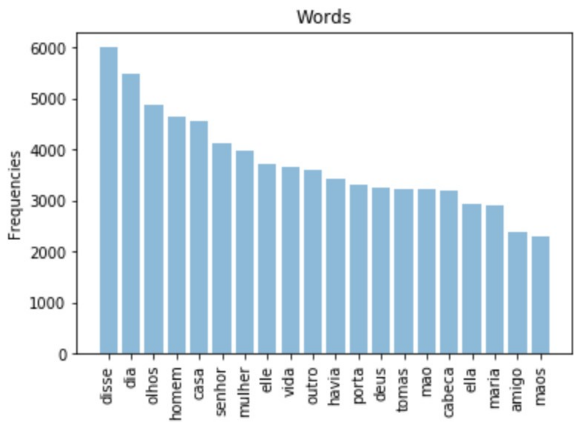
A1 – Word Cloud for José Rodrigues Santos



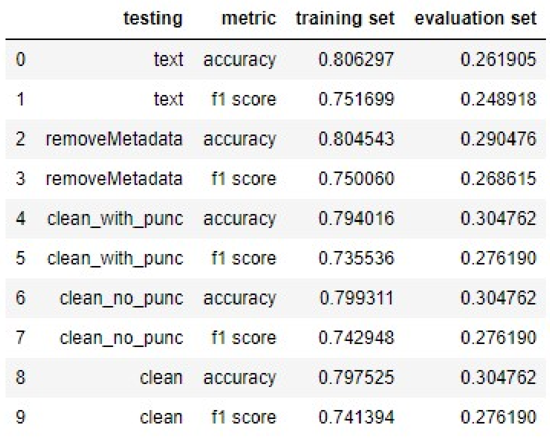
A2 - Word Cloud for Camilo Castelo Branco



A3 – Bar plot of the frequencies of the words

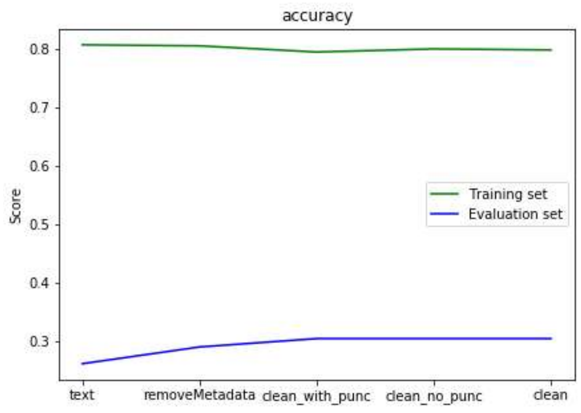


A4 – Table of the scores of the Bag-Of-Words baseline

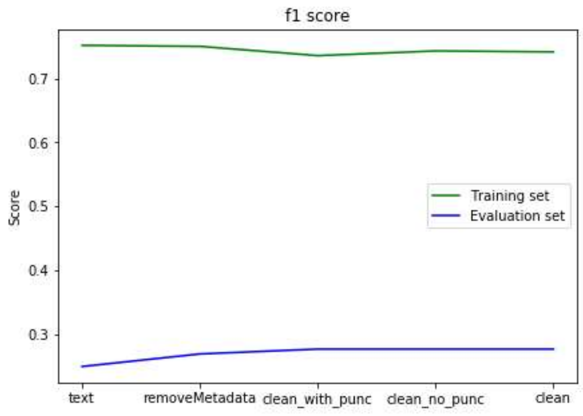


	testing	metric	training set	evaluation set
0	text	accuracy	0.806297	0.261905
1	text	f1 score	0.751699	0.248918
2	removeMetadata	accuracy	0.804543	0.290476
3	removeMetadata	f1 score	0.750060	0.268615
4	clean_with_punc	accuracy	0.794016	0.304762
5	clean_with_punc	f1 score	0.735536	0.276190
6	clean_no_punc	accuracy	0.799311	0.304762
7	clean_no_punc	f1 score	0.742948	0.276190
8	clean	accuracy	0.797525	0.304762
9	clean	f1 score	0.741394	0.276190

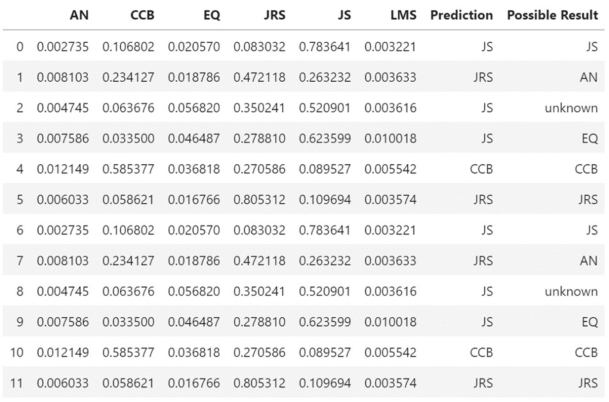
A5 – Accuracy Graph



A6 – F1-Score Graph

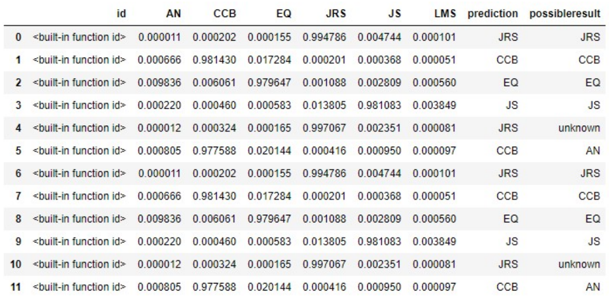


A7 - Table of results of FastText



	AN	CCB	EQ	JRS	JS	LMS	Prediction	Possible Result
0	0.002735	0.106802	0.020570	0.083032	0.783641	0.003221	JS	JS
1	0.008103	0.234127	0.018786	0.472118	0.263232	0.003633	JRS	AN
2	0.004745	0.063676	0.056820	0.350241	0.520901	0.003616	JS	unknown
3	0.007586	0.033500	0.046487	0.278810	0.623599	0.010018	JS	EQ
4	0.012149	0.585377	0.036818	0.270586	0.089527	0.005542	CCB	CCB
5	0.006033	0.058621	0.016766	0.805312	0.109694	0.003574	JRS	JRS
6	0.002735	0.106802	0.020570	0.083032	0.783641	0.003221	JS	JS
7	0.008103	0.234127	0.018786	0.472118	0.263232	0.003633	JRS	AN
8	0.004745	0.063676	0.056820	0.350241	0.520901	0.003616	JS	unknown
9	0.007586	0.033500	0.046487	0.278810	0.623599	0.010018	JS	EQ
10	0.012149	0.585377	0.036818	0.270586	0.089527	0.005542	CCB	CCB
11	0.006033	0.058621	0.016766	0.805312	0.109694	0.003574	JRS	JRS

A8 – Table of results of LSTM



	id	AN	CCB	EQ	JRS	JS	LMS	prediction	possibleresult
0	<built-in function id>	0.000011	0.000202	0.000155	0.994786	0.004744	0.000101	JRS	JRS
1	<built-in function id>	0.000666	0.981430	0.017284	0.000201	0.000368	0.000051	CCB	CCB
2	<built-in function id>	0.009836	0.006061	0.979647	0.001088	0.002809	0.000560	EQ	EQ
3	<built-in function id>	0.000220	0.000460	0.000583	0.013805	0.981083	0.003849	JS	JS
4	<built-in function id>	0.000012	0.000324	0.000165	0.997067	0.002351	0.000081	JRS	unknown
5	<built-in function id>	0.000805	0.977588	0.020144	0.000416	0.000950	0.000097	CCB	AN
6	<built-in function id>	0.000011	0.000202	0.000155	0.994786	0.004744	0.000101	JRS	JRS
7	<built-in function id>	0.000666	0.981430	0.017284	0.000201	0.000368	0.000051	CCB	CCB
8	<built-in function id>	0.009836	0.006061	0.979647	0.001088	0.002809	0.000560	EQ	EQ
9	<built-in function id>	0.000220	0.000460	0.000583	0.013805	0.981083	0.003849	JS	JS
10	<built-in function id>	0.000012	0.000324	0.000165	0.997067	0.002351	0.000081	JRS	unknown
11	<built-in function id>	0.000805	0.977588	0.020144	0.000416	0.000950	0.000097	CCB	AN