Text Mining Project

Bruno de Lima Vieira, Leonardo Motta Perazzo Lannes, Alex Panchot
M20190922, M20180036, M20190546

Introduction

The aim of this project is to take a small corpus (500 or 1000 words) of raw text, and preprocess it, and by use of a model, understand who the author of the text is. The training data for this project is a non-balanced corpora derived from six different Portuguese authors. The corpora has several works from each author such that there are a total of 63 different source documents.

Method/Approach

The approach to this problem was developed in several steps. First, the text needs to be cut up into 500 word blocks as this is the size of the test data. Next, the raw text of the corpora needs to be cleaned in a relatively simple and reproducible way. Finally, the corpora is run through a machine learning model to determine the author of the text.

Preprocessing

The goals of preprocessing are to clean the corpora such that it contains only real words, and then to eliminate the possibility that a single word can be counted more than once. We also aim to reduce the total number of vocabulary words as a larger vocab size falls victim to the curse of dimensionality.

Because we wanted to test different levels of preprocessing, we created three different corporas from the original with different amounts of preprocessing. The simplest is just minor changes, like lowercasing. Then one has all of the steps applied “complete” and one with everything except removing stopwords “medium”.

The first step of the preprocessing is to remove all accents from the corpora. We do this because we determined that firstly, the accents do not significantly contribute to the understanding of a word; and secondly, the use of accents in the spelling of Portuguese has changed over the last 100 years. Because some of the documents were written as far back as the beginning of the 20th century and some of the documents were written only a few years ago, this preprocessing is important to ensure that the same word is recognized by a model, regardless of when it was written.

Similarly, we separated contractions into their expanded forms. This again means that the same word is not considered to be a separate word by the model just because it has a different spelling.

Next, we use regular expressions to replace all special characters with a whitespace. This helps to remove punctuation as well as any non letter or digit characters. We also lowercase all of the words as this again removes the chance that the same word is counted twice (but it does open up the chance that two words that are different only in their capitalization are counted as the same word).

Regarding the process of removing word affixes in order to get to a base form of the word, we tried stemming and lemmatization. At first, the lemmatizer available for Portuguese (Spacy library) seemed to not work well, but we tested against stemmers and the results with lemmatization were slightly better.

Finally, we removed stop words from the text. After removing them, we do not see a significant reduction in the vocabulary size, but we do eliminate frequent words that would slow the model down from learning.

In addition to just cleaning the data, we also balanced the corpora with undersampling. Because neural networks need more data than the traditional machine learning models, there are different levels of undersampling applied for the respective model.

Model

For the final model, we tested many combinations of ML algorithms and input methods. We used a grid search to find the best model from six different ML algorithms: Naive Bayes, Logistic Regression, SVM, SVM with Stochastic Gradient Descent, Random Forest, and Gradient Boosting. For the input we used Bag-of-Words and TF-IDF as well as trying both uni- and bi-grams. Each of these models were tested with both of the more preprocessed training sets. Additionally, the CBOW algorithm was used to create word embeddings from the least preprocessed training set. This data was then run through all of the ML algorithms as well. In addition to the traditional ML algorithms, RNN and CNN were also tried but did not achieve any meaningful success. These will be discussed a little bit in the appendix.

Baseline

In order to get a baseline for the models to beat, we created a function that predicted the most frequent class in the training data. Unsurprisingly, it just chose the author with the highest number of training samples every time. In order to beat this model, our model needs to at least pick another author some of the time.

Results and Discussion

After the grid search finishes, it returns the best algorithm for the input parameters. With this particular algorithm, we can use it with the test data to return the authors for the test documents.

Results

The authors of the test documents are listed below with the confusion matrices and classification reports in the appendix.

	Complete Clean	Medium Clean	CBOW
0	JoseSaramago	JoseSaramago	JoseSaramago
1	AlmadaNegreiros	AlmadaNegreiros	AlmadaNegreiros
2	LuisaMarquesSilva	LuisaMarquesSilva	LuisaMarquesSilva
3	EcaDeQueiros	EcaDeQueiros	EcaDeQueiros
4	CamiloCasteloBranco	CamiloCasteloBranco	CamiloCasteloBranco
5	JoseRodriguesSantos	JoseRodriguesSantos	JoseRodriguesSantos
6	JoseSaramago	JoseSaramago	JoseSaramago
7	AlmadaNegreiros	AlmadaNegreiros	AlmadaNegreiros
8	LuisaMarquesSilva	LuisaMarquesSilva	LuisaMarquesSilva
9	EcaDeQueiros	EcaDeQueiros	EcaDeQueiros
10	CamiloCasteloBranco	CamiloCasteloBranco	CamiloCasteloBranco
11	JoseRodriguesSantos	JoseRodriguesSantos	JoseRodriguesSantos

Discussion

While the three best models are in agreement about the authors (we would be in trouble if not!), they interestingly do not share the exact same parameters. Both the complete and medium preprocessed ones use the Multinomial Naive Bayes, but the complete uses TF-IDF with unigrams and the medium uses B.O.W with bigrams. The CBOW model uses a logistic regression. While each model is not exactly the same, they do produce the same results with all of them getting an accuracy of at least 97% (reaching 99.3%!). What is unknown though, is how accurate the second best permutation in the grid search is. Perhaps the other ML algorithms are just as effective with the test set but were only a tiny bit worse for the development set. This would be an important thing to research more as some algorithms would generalize better even if their performance in a specific case is a little worse. For instance if we did not balance the dataset, maybe these algorithms would perform much worse.

Conclusion

For us, the main conclusion to this project is that while there are many existing tools available to preprocess and build a model for a specific corpora, it still requires knowledge about NLP to understand why a certain preprocessing step may help or hurt the accuracy of a model. As each corpora and task are different, the exact same steps that we used here cannot be reused as for example the use of accents may be important. In fact, perhaps by not removing accents and contractions, this may have made it easier for a model to discern the age and style of a writer.

Beyond the NLP knowledge, we also had to remember that the computer cannot see letters or words, but just numbers. Because of this, we had to properly embed the words so the computer could make sense of them. Even though the models were fine with sparse matrices, using vectorized embeddings could have helped to improve performance even more. This is especially true with neural networks.

After careful preprocessing and selection of a suitable model, the computer is able to nearly instantly “read” a block of text and tell us who the author is.

References

Dipanjan Sarkar. 2019. *Text Analytics with Python: A Practitioner's Guide to Natural Language Processing.* Springer, New York

This is the link to our project Github page: https://github.com/apanchot/text_mining

Appendix

Model Confusion Matrix / Classification Report

Baseline

The baseline tables show that it just guessed the author with the highest number of training samples.

	AlmadaNegreiros	CamiloCasteloBranco	EcaDeQueiros	JoseRodriguesSantos	JoseSaramago	LuisaMarquesSilva
AlmadaNegreiros	0	0	0	20	0	0
CamiloCasteloBranco	0	0	0	19	0	0
EcaDeQueiros	0	0	0	22	0	0
JoseRodriguesSantos	0	0	0	29	0	0
JoseSaramago	0	0	0	26	0	0
LuisaMarquesSilva	0	0	0	18	0	0

	precision	recall	f1-score	support
AlmadaNegreiros	0.000	0.000	0.000	20
CamiloCasteloBranco	0.000	0.000	0.000	19
EcaDeQueiros	0.000	0.000	0.000	22
JoseRodriguesSantos	0.216	1.000	0.356	29
JoseSaramago	0.000	0.000	0.000	26
LuisaMarquesSilva	0.000	0.000	0.000	18
accuracy			0.216	134
macro avg	0.036	0.167	0.059	134
weighted avg	0.047	0.216	0.077	134

Complete Preprocessing (removing stopwords)

	AlmadaNegreiros	CamiloCasteloBranco	EcaDeQueiros	JoseRodriguesSantos	JoseSaramago	LuisaMarquesSilva
AlmadaNegreiros	20	0	0	0	0	0
CamiloCasteloBranco	0	19	0	0	0	0
EcaDeQueiros	0	0	21	0	1	0
JoseRodriguesSantos	0	0	0	29	0	0
JoseSaramago	0	0	0	0	26	0
LuisaMarquesSilva	0	0	0	0	0	18

	precision	recall	f1-score	support
AlmadaNegreiros	1.000	1.000	1.000	20
CamiloCasteloBranco	1.000	1.000	1.000	19
EcaDeQueiros	1.000	0.955	0.977	22
JoseRodriguesSantos	1.000	1.000	1.000	29
JoseSaramago	0.963	1.000	0.981	26
LuisaMarquesSilva	1.000	1.000	1.000	18
accuracy			0.993	134
macro avg	0.994	0.992	0.993	134
weighted avg	0.993	0.993	0.993	134

Medium Preprocessing (without removing stopwords)

	AlmadaNegreiros	CamiloCasteloBranco	EcaDeQueiros	JoseRodriguesSantos	JoseSaramago	LuisaMarquesSilva
AlmadaNegreiros	19	0	0	0	1	0
CamiloCasteloBranco	0	19	0	0	0	0
EcaDeQueiros	0	0	21	0	1	0
JoseRodriguesSantos	0	0	0	28	1	0
JoseSaramago	0	0	0	0	26	0
LuisaMarquesSilva	0	0	0	0	1	17

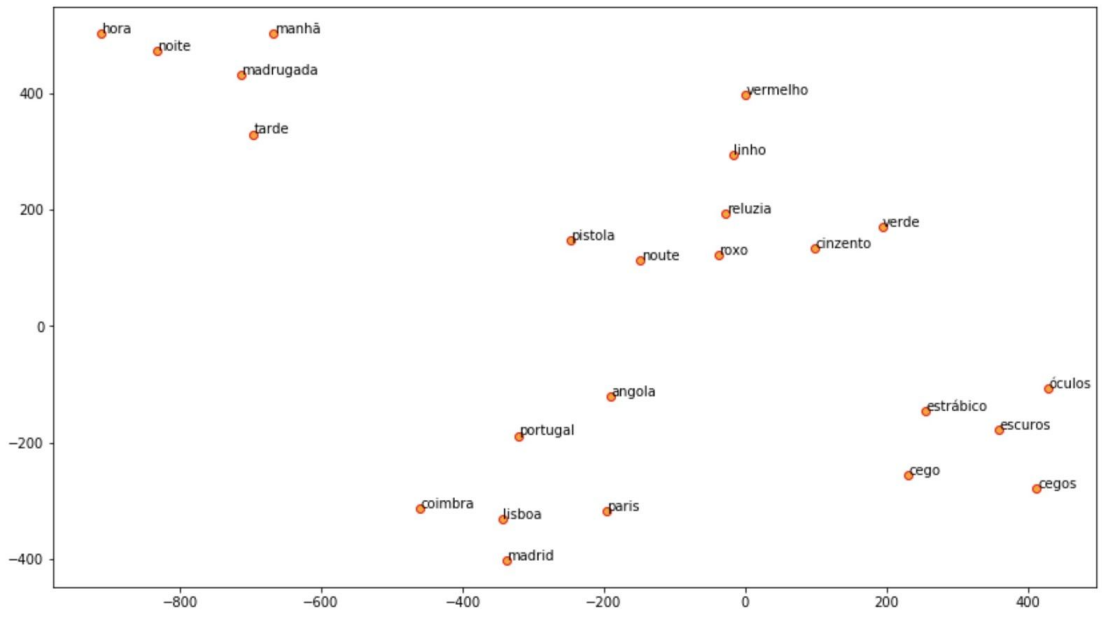
	precision	recall	f1-score	support
AlmadaNegreiros	1.000	0.950	0.974	20
CamiloCasteloBranco	1.000	1.000	1.000	19
EcaDeQueiros	1.000	0.955	0.977	22
JoseRodriguesSantos	1.000	0.966	0.982	29
JoseSaramago	0.867	1.000	0.929	26
LuisaMarquesSilva	1.000	0.944	0.971	18
accuracy			0.970	134
macro avg	0.978	0.969	0.972	134
weighted avg	0.974	0.970	0.971	134

CBOW

	AlmadaNegreiros	CamiloCasteloBranco	EcaDeQueiros	JoseRodriguesSantos	JoseSaramago	LuisaMarquesSilva
AlmadaNegreiros	19	0	1	0	0	0
CamiloCasteloBranco	0	19	0	0	0	0
EcaDeQueiros	0	0	22	0	0	0
JoseRodriguesSantos	0	0	0	28	1	0
JoseSaramago	0	0	0	0	26	0
LuisaMarquesSilva	0	0	0	0	0	18

	precision	recall	f1-score	support
AlmadaNegreiros	1.000	0.950	0.974	20
CamiloCasteloBranco	1.000	1.000	1.000	19
EcaDeQueiros	0.957	1.000	0.978	22
JoseRodriguesSantos	1.000	0.966	0.982	29
JoseSaramago	0.963	1.000	0.981	26
LuisaMarquesSilva	1.000	1.000	1.000	18
accuracy			0.985	134
macro avg	0.987	0.986	0.986	134
weighted avg	0.986	0.985	0.985	134

As the CBOW word embeddings are vectors, we can plot them. Below are a few words that are similar in meaning and because of that the vectors line up appropriately.



NNs

In addition to the “traditional” ML algorithms, we tried to implement two types of neural networks. Unfortunately, they were not successful for a few reasons and we therefore could not use them to classify the authors. Despite the problems we had to tune the NNs, they only had difficulties to make predictions about the less frequent classes (Almada Negreiros and Luisa Marques Silva), reaching the target class in all other cases.

RNN

The RNN model used a single LSTM layer followed by a dense output layer. One of the main problems with neural networks is that we cannot definitely say where the problem is. However there are a few places of concern. As we did not have many training samples the NN could not generalize when shown the new data. This is shown as it looks like the model overfit the training data quite quickly. However, if we used many more training samples we would need to let the model run for a long time to train. This is difficult without a GPU.

CNN

The CNN is very much the same as the CBOW word embedding. It uses a window to predict not a center word but just who wrote that block of text.