Language Guesser and Locator

Teoria Algorítmica da Informação
Ano letivo 2022/2023

Dinis dos Santos Lei, nMec 98452
João Pedro Saraiva Borges, nMec 98155
Vicente Samuel Gonçalves Costa, nMec 98515

Universidade De Aveiro

Introduction

This project tackles the problem of classification of text using data compression. The problem at hand involves determining the similarity between a target text (t) and a set of reference texts (ri), where each reference text represents a sample from a specific language and the goal is to identify the language of the target text.

Traditionally, this classification problem is approached by extracting and selecting features from the texts. These features are then fed into a function that maps them to different classes and performs the classification.

Alternatively, the representation of the original data can be viewed as a form of lossy data compression, where a small set of features captures the essence of the data. The idea is to create for each class of the reference text (ri) a compression model, then with each model, try to compress the target text (t). We assign to the target, the class that requires the fewer bits to compress it.

Lang

This program is given a reference file and a target file and generates a file with the amount of information generated by each symbol of the target file. This program uses two compressions models, a copy model that is similar to the one in the previous Lab, and a context model.

Copy Model

The copy model is a data compression technique that tries to predict the next outcome by looking at a reference in the past.

The model estimates the probability of the next symbol being the same as its past reference, and based on that prediction it can calculate the amount of information gained.

The probabilities are calculated by having counters of hits (Nh) and misses (Nm). These counters will give the relative frequency Nh/(Nh+Nf). However, due to the problem of giving 0 probability to events that haven't appeared in the model, for example, the first time an event is calculated, a smoothing parameter alpha is introduced.

With the probability calculated we can then estimate the amount of information generated by the new symbol, s, -log2(P(s)).

Context Model

We use a finite-context model to represent data dependencies. This model computes the probability of a given symbol S appearing after a sequence of order k.
This is achieved by storing a table of the counts of each given symbol of the alphabet for each sequence.
Then with the table constructed we can estimate the probability of symbol S given sequence c (P(S|c) with formula [1], with α being a smoothing parameter for dealing with problems of zero probability.

$$P(S|c) = \frac{N(e|c) + \alpha}{\sum s \in \Sigma N\,(s|c) + \alpha|\Sigma|} \qquad [1]$$

Implementation

From these two models we generated 4 solutions for encoding the target text.
Only using the Copy Model for encoding the targets (M1).
Only using the Context Model for encoding the targets (M2).
Using the Context Model when the Copy Model isn't active (M3).
Using the Context Model when the Copy Model misses (M4).

Every approach was evaluated using the findLang program. The reference samples used ranged 11 different languages, each with a size of around 4 Mb. The target samples were 29 excerpts from books of 5 different languages (Portuguese, English, Spanish, Greek and Latin), all with around 10 kb each.

We evaluate the accuracy of the model, that is, the number of excerpts classified correctly (at least 60% confidence that the excerpt is from the correct language), as well as the confidence of the model, that is the mean of the accuracy of the correct language for each target sample.

Results

	M1	M2	M3	M4
Accuracy	0.0	0.414	0.0	0.586
Confidence	0.182	0.612	0.214	0.659

From the tests we can check that the best performance is from the M4 (Context Model when the Copy Model misses).
The copy model alone doesn't provide a good enough confidence for it to be useful in the language detection, but combined with the context model it generates good results in both confidence and accuracy. This combined model is however a lossy compressor because it doesn't take into account the switches between the two models.

FindLang

This program uses the Lang program to guess what language a given sample text file is written in.

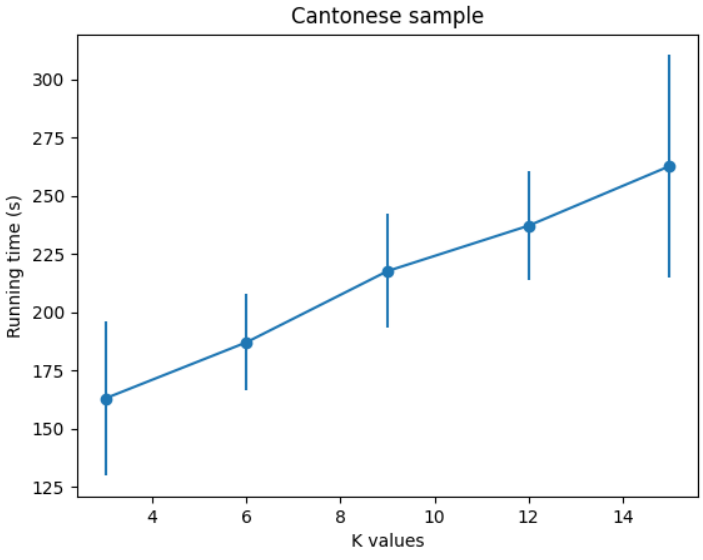
It starts by calling the Lang program for each language reference in the folder *“examples/language”* with the target sample file. Then, it calculates the accuracy of each language by counting the number of times when the amount of information in each symbol is lower than the expression *log2(#alphabet)/2* (#alphabet is the size of the alphabet).

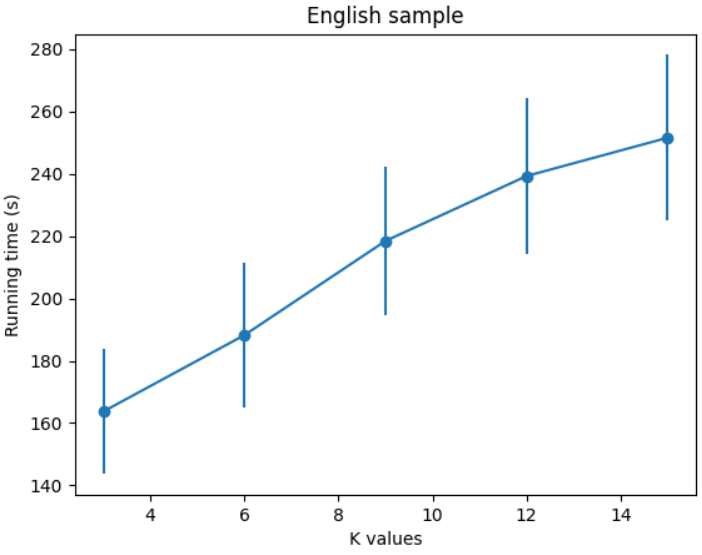
Finally we assume that the sample text is written in the language with an accuracy above 60% .

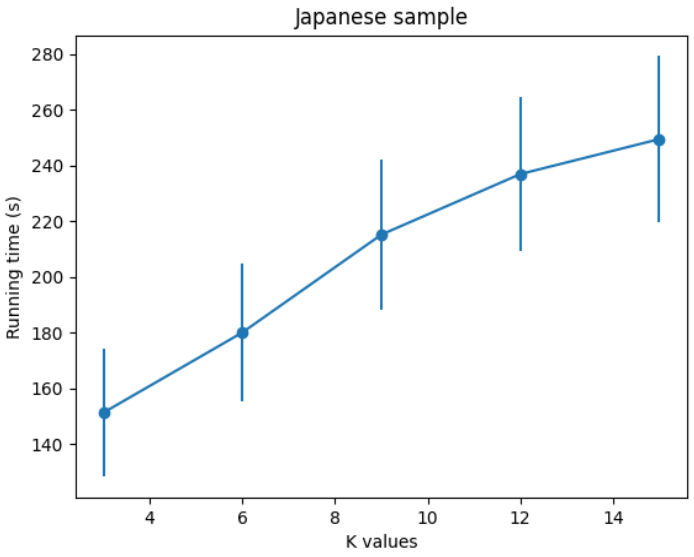
Results

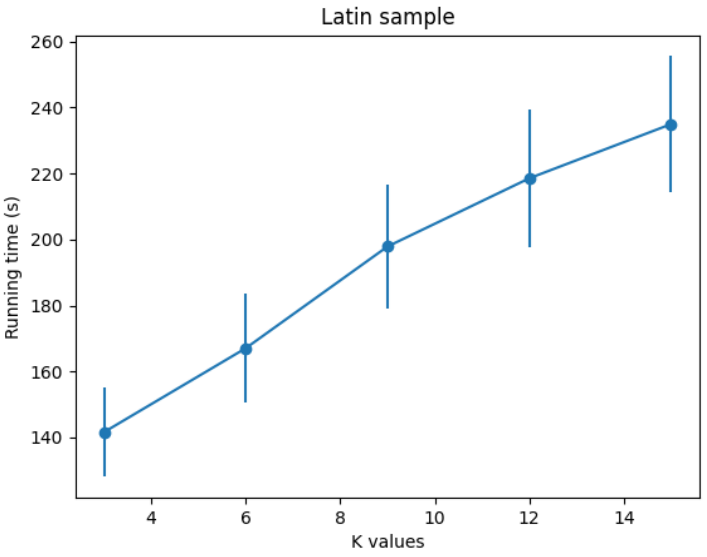
Computing Time Test

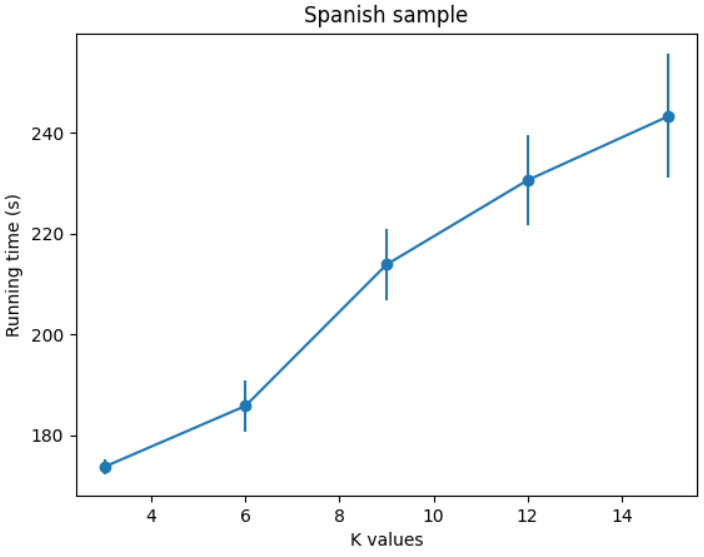
As a reference file we use 11 language : arab (7,5mb), cantonese (7,5mb), english (3,9mb), greek (7,5mb), japanese (7,5mb), korean (7,5mb), latin (4,0mb), portuguese (4,0mb), russian (7,5mb), scottish (4,3mb) and spanish (4,0mb). As a sample file we used 8 examples: cantonese (2,6kb), english (2,9kb), japanese (801 bytes), latin (1,4kb), portuguese (1,3kb), russianEnglish (2,6kb), russian (1,9kb) and spanish (6,2kb)
To evaluate the computing time of this program, we ran this program with 5 different values of k for 8 different samples 3 times. Below are the results:

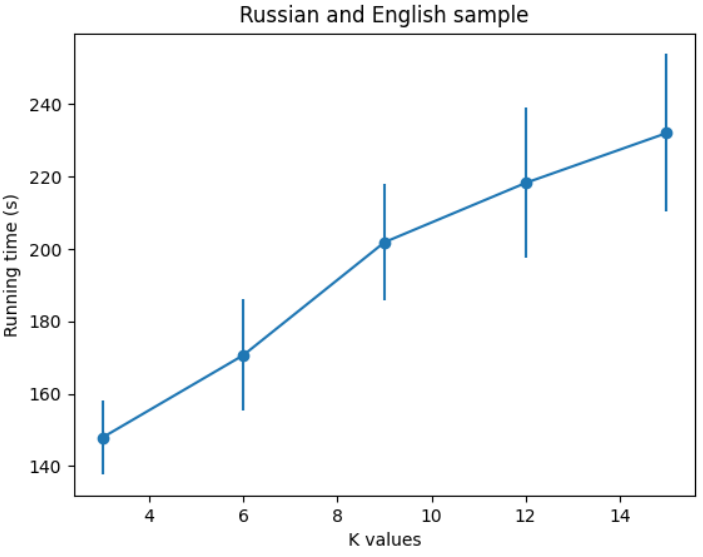


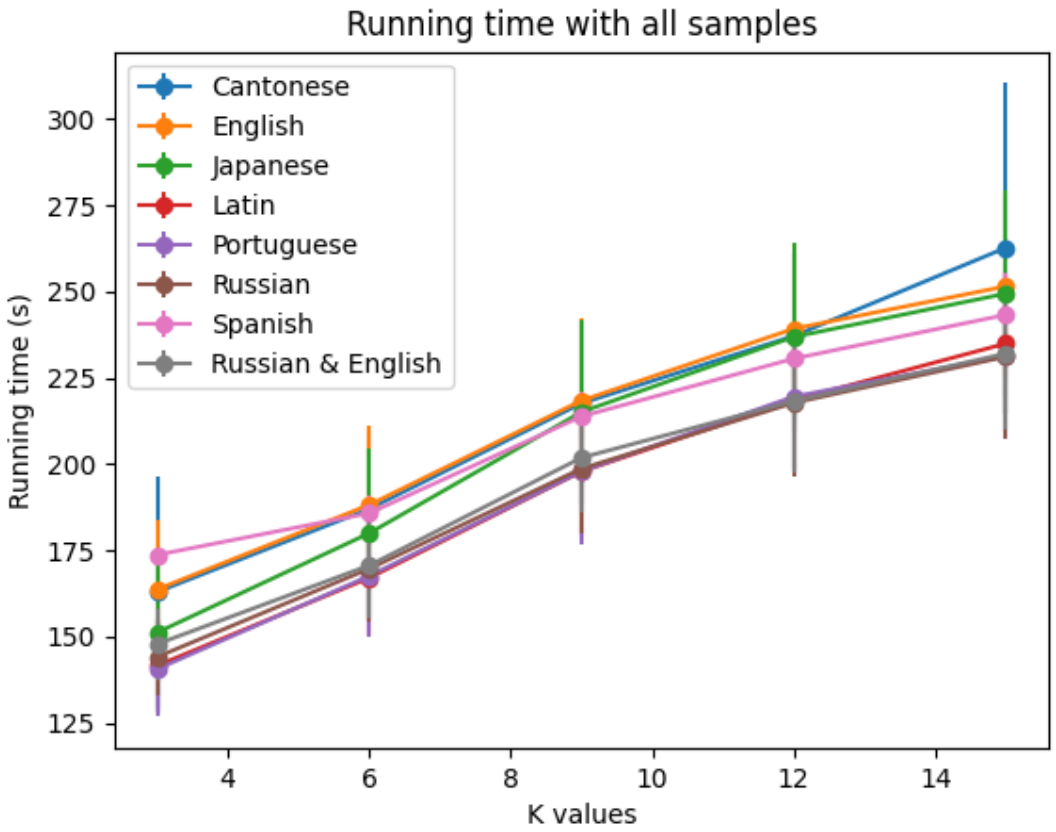



Japanese sample
Running time (s)
K values
280
260
240
220
200
180
160
140
4
6
8
10
12
14


Latin sample
Running time (s)
K values
260
240
220
200
180
160
140
4
6
8
10
12
14


Spanish sample
Running time (s)
K values
240
220
200
180
4
6
8
10
12
14

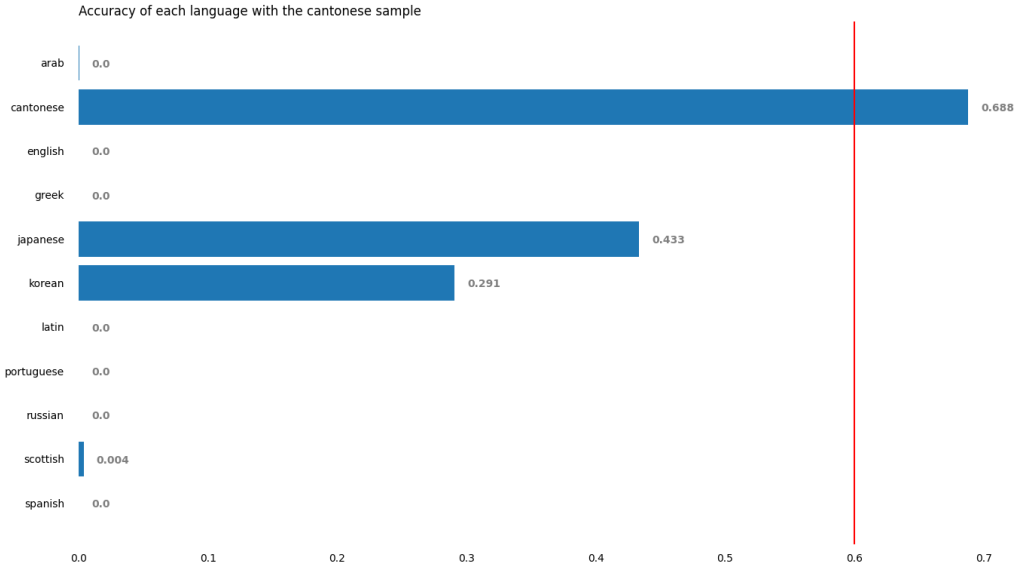

Russian and English sample
Running time (s)
K values
240
220
200
180
160
140
4
6
8
10
12
14

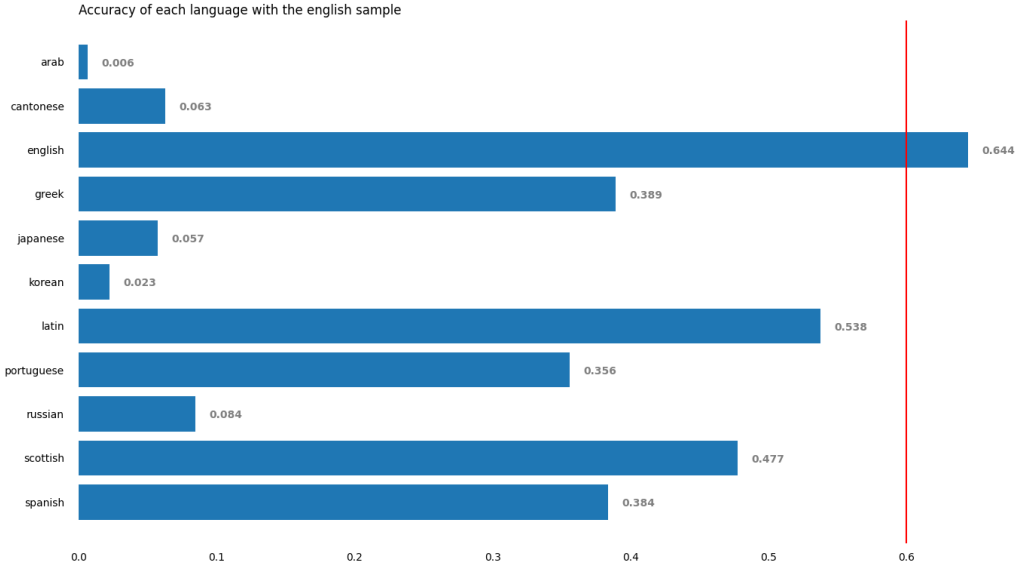


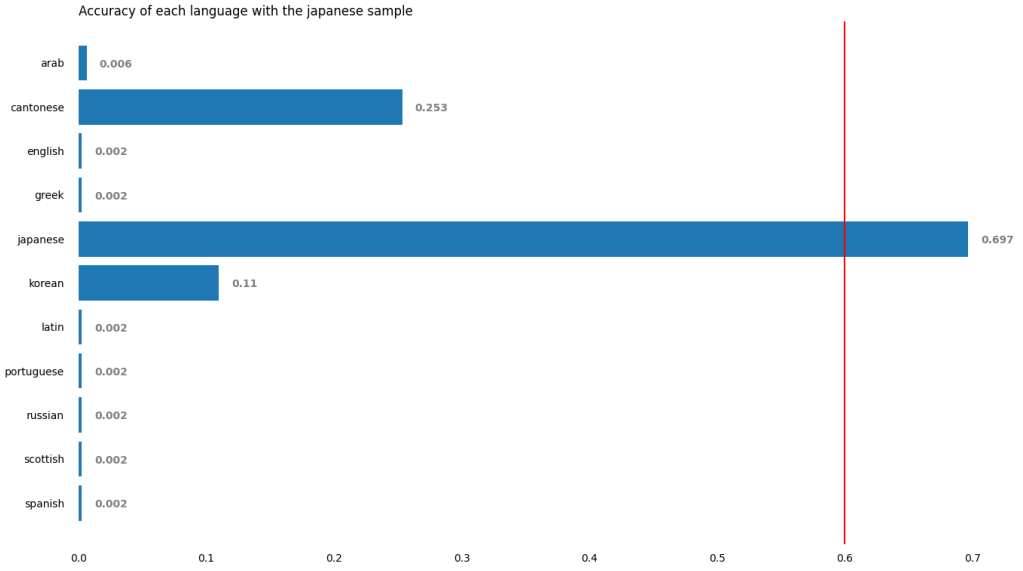
As we can see in the results above, the computing time for each sample is directly proportional with the value of *"k"*. We can also visualize that the cantonese sample was the longest to run with a *"k"* of 15 but the spanish sample was the longest with a *"k"* value of 3.

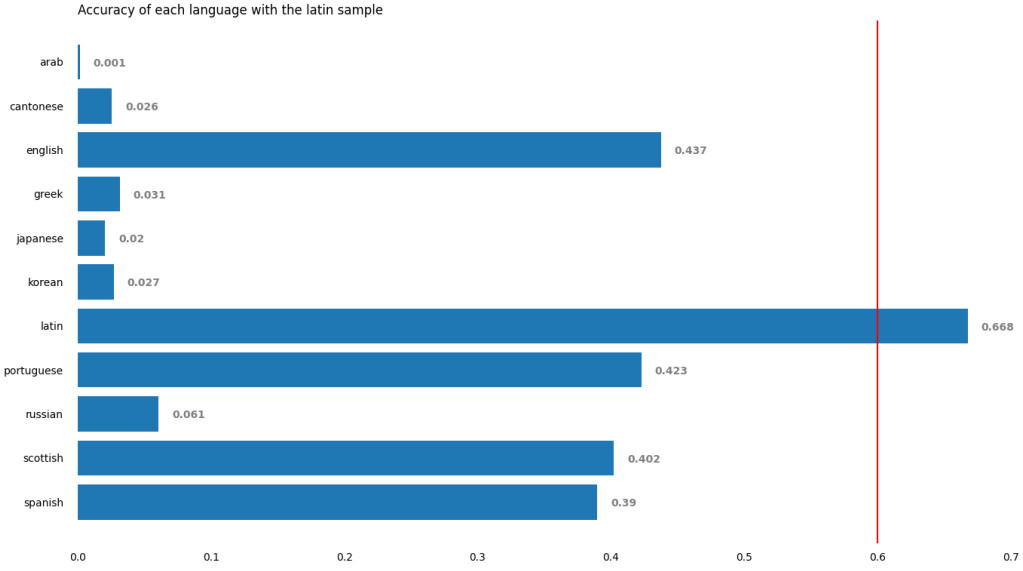
Accuracy Test

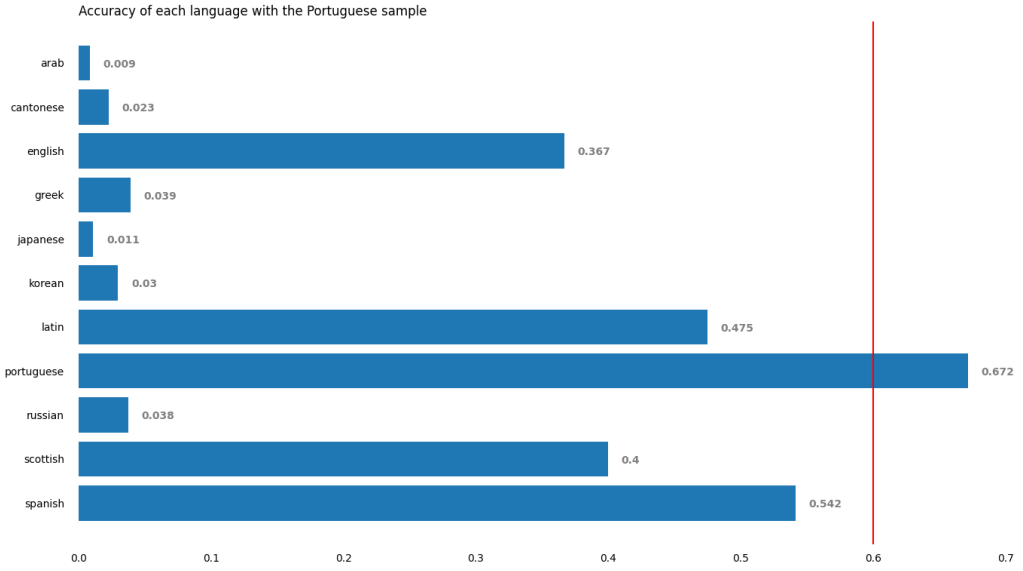
To evaluate the accuracy of the program, we ran the program with a *"k"* value of 6 and with the same 8 samples from the previous test. The results for the 11 reference languages are illustrated in the bar graphs below.

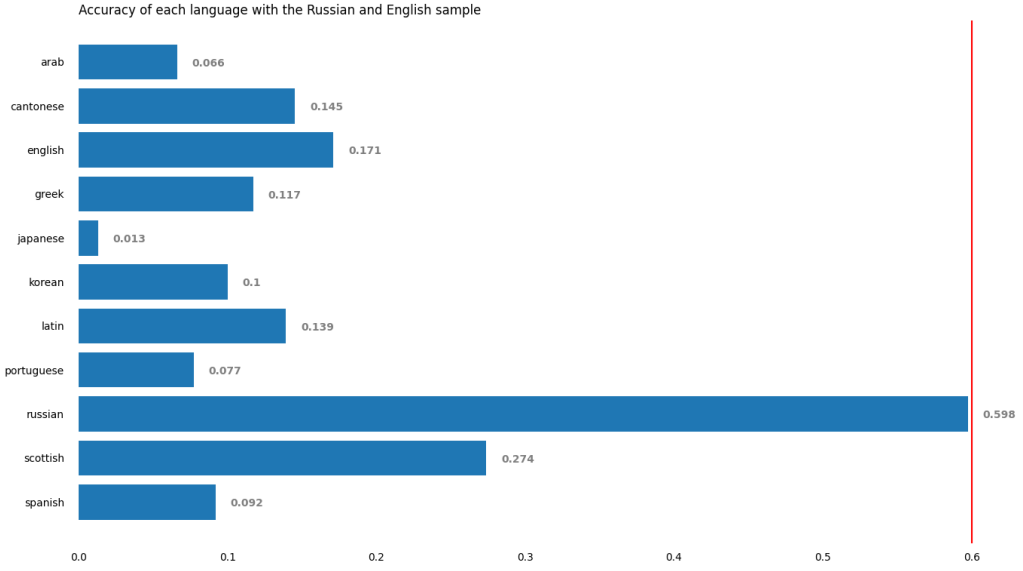

Accuracy of each language with the cantonese sample
arab
cantonese
english
greek
japanese
korean
latin
portuguese
russian
scottish
spanish
0.0
0.688
0.0
0.0
0.433
0.291
0.0
0.0
0.0
0.004
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7

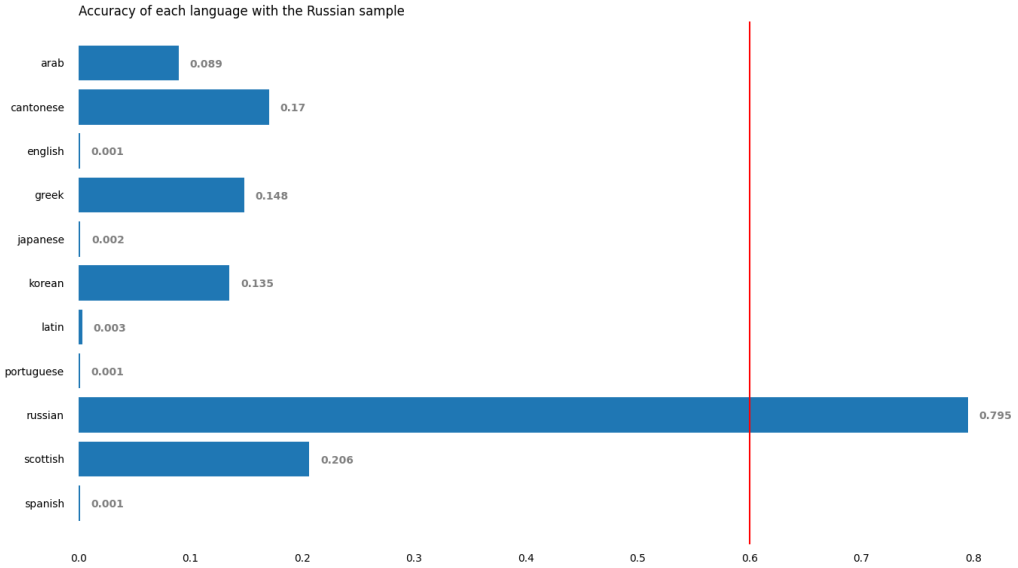

Accuracy of each language with the english sample
arab
cantonese
english
greek
japanese
korean
latin
portuguese
russian
scottish
spanish
0.006
0.063
0.644
0.389
0.057
0.023
0.538
0.356
0.084
0.477
0.384
0.0
0.1
0.2
0.3
0.4
0.5
0.6

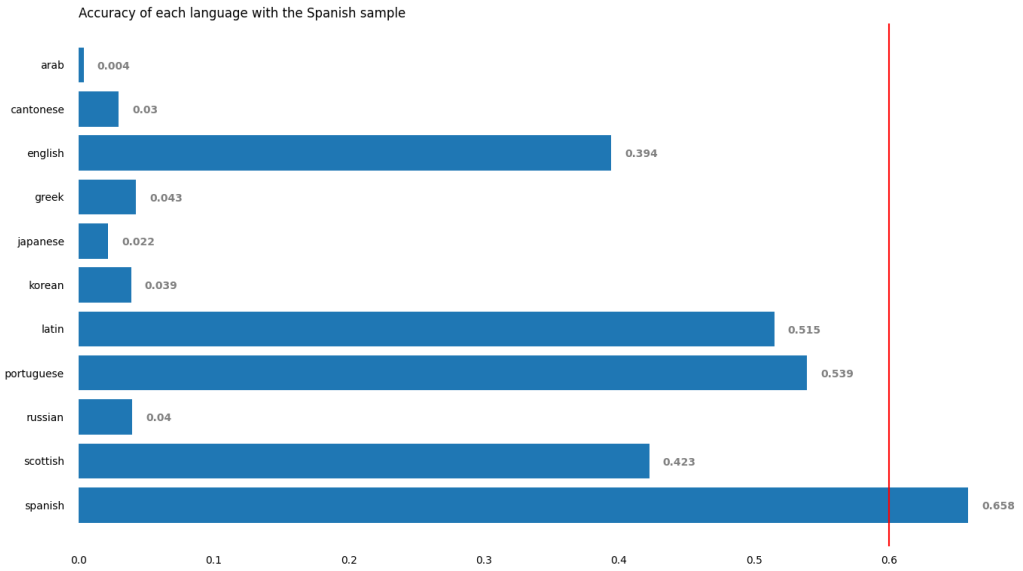

Accuracy of each language with the japanese sample
arab
cantonese
english
greek
japanese
korean
latin
portuguese
russian
scottish
spanish
0.006
0.253
0.002
0.002
0.697
0.11
0.002
0.002
0.002
0.002
0.002
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7


Accuracy of each language with the latin sample
arab
cantonese
english
greek
japanese
korean
latin
portuguese
russian
scottish
spanish
0.001
0.026
0.437
0.031
0.02
0.027
0.668
0.423
0.061
0.402
0.39
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7


Accuracy of each language with the Portuguese sample
arab
cantonese
english
greek
japanese
korean
latin
portuguese
russian
scottish
spanish
0.009
0.023
0.367
0.039
0.011
0.03
0.475
0.672
0.038
0.4
0.542
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7


Accuracy of each language with the Russian and English sample
arab
cantonese
english
greek
japanese
korean
latin
portuguese
russian
scottish
spanish
0.066
0.145
0.171
0.117
0.013
0.1
0.139
0.077
0.598
0.274
0.092
0.0
0.1
0.2
0.3
0.4
0.5
0.6





As we can see in the graphs above, the overall results are satisfactory since the program could guess correctly the language of the sample file on all of the tests, except for the file which is in English and in russian. In this file, the program fails to recognize the russian text and the english text one lowers the accuracy of the other.

LocateLang

This program uses the Lang program to be able to identify in what language a sequence is written and be able to identify
*different language sequences in a single sentence.

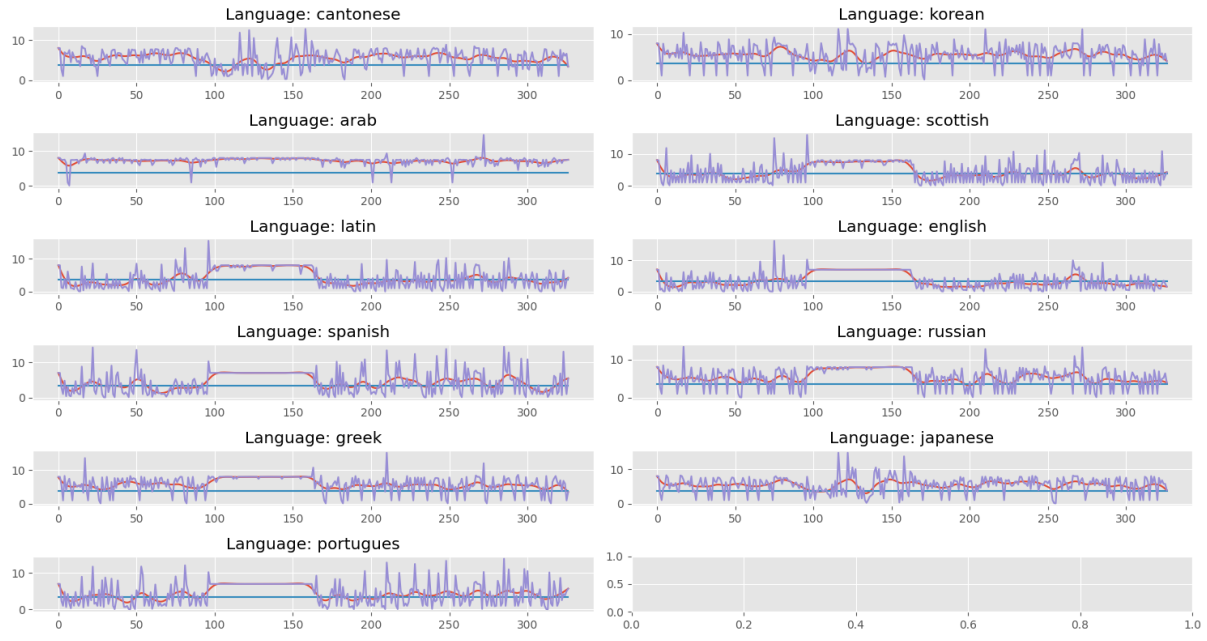
The program starts by running the lang module with many different languages files and a given sample that will be evaluated, this sample contains mixed sequences of phrases from different languages. Then, we gather the results of the lang model, which are various files containing, in the first line, the size of the alphabet, and then the amount of information that each byte (symbol) was encoded to.

With the gathered information, we make a matrix with numpy, which contains in each line the results of each language, and then smooth out the values in the matrix to give better results. Finally, we run through the matrix to find out the lowest values in all columns, and then calculate if the value in question is lower than a threshold, which is the log base 2 of the size of the alphabet in question divided by 2; if this value is not lower than that, then the value is not considered and the symbol does not have any language associated; if it is indeed lower, we simply add to an array with the language.

We also added, as an aesthetic feature, colors to each one of the languages, which print each of the symbols of the sample in the corresponding color.

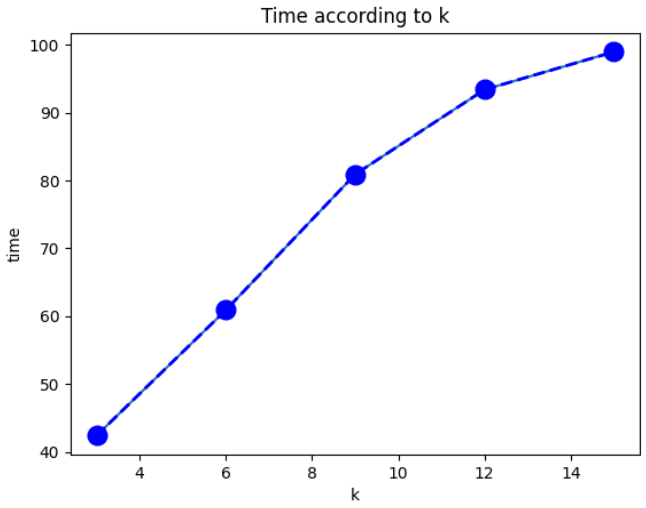
```
cantonese
korean
arab
scottish
latin
english
spanish
russian
greek
japanese
portugues
None
The information, discovered in bank documents obtained Los resultados son aún muy preliminares 成大致可以分为三个阶段。第
一阶段，问题的发生 also contradict statements from the president, who said his family did not receive any money from China
.
Mr. Biden denies knowing about the business deals or having
The information, discovered in bank documents obtained Los resultados son aún muy preliminares 成大致可以分为三个阶段。第
一阶段，问题的发生 also contradict statements from the president, who said his family did not receive any money from China
.
Mr. Biden denies knowing about the business deals or having
```

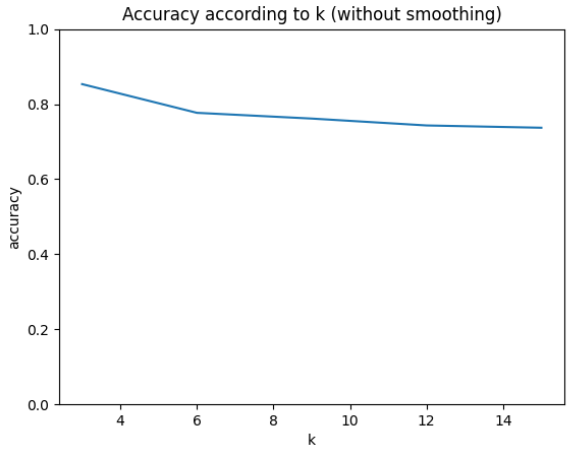
At the end we give a graphic to each corresponding language, that shows how many the amount of information (y axis) in each symbol of the sequence (x axis), where the blue line is the values of the amount of information not smoothed, while the red line shows the smoothed values, and the blue line shows the optimal amount of information after encoding the sequence, any value below than that shows a good encoding of the symbol, which means that it is probable that the symbol is in that corresponding language.

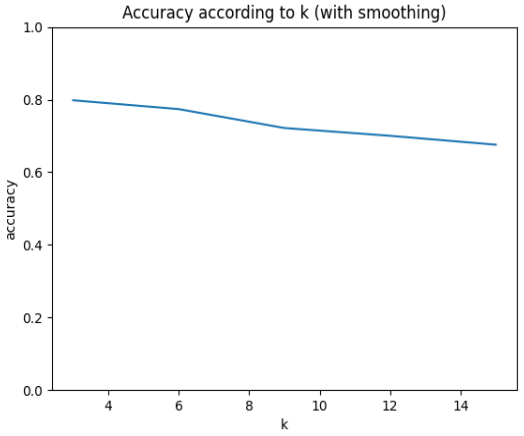


Results

We tested locatelang with different values of K when running the lang program, for the sake of accuracy comparison between these results.







We can conclude that the values obtained in accuracy when running locate lang with higher K values are relatively lower to the ones with lower K values. Most of the times the results are also better when no smoothing is applied to the matrix, giving higher accuracy levels.

Future Work

Something that we could improve in the locatelang program is the context given. The program only gathers the languages in which a specific symbol has the lowest amount of information, not taking into account what language the previous symbol was in. We could improve this by checking the language of the previous symbol, and check the amount of information of the next symbol in the same language; even if it is not the lowest amount of information between all the languages, if it remains lower than log base 2 of the size of the alphabet, it is already in a optimal encoding, which means that it is probable that the symbol is still from the language of the previous symbol.

Conclusion

We believe our program performance in every strand of work is quite notable; from the single module lang to the find and locate lang, we believe our program performance is not only fast, but also quite accurate, it rarely misses the classification of the correct language in the examples given, which proves that a good strategy was used in our lang module to classify our samples in the correct language, or even better, to encode in the best way possible the samples given to the program.