

Language recognition system

Diogo Mendonça 93002

Leandro Silva 93446

Margarida Martins 93169

Index

Overview	1
Program options	2
Functions	3
Results	5
Problems and Future Work	15
Conclusion	17
Appendix I - Language sources	18

1. Overview

In this work, our goals were to develop three programs, *lang*, *findlang* and *locatelang*. The first one aimed to estimate the number of bits required to compress a text using a finite-context model trained with a given text. The second program is a language recognition system that guesses the language of a text using a finite-context model for each language. Finally, *locatelang* identifies which languages are present in a file and where they are, using several finite-context models for each language.

The chosen programming language is C++, as it allows for greater control on the data structures and execution of the code.

2. Program options

lang: In order to run this program it is required to pass two file paths, the first one will be used to train the fcm and the second one is the text under analysis. It is also necessary to pass the context size (k) used in training the model and the smoothing parameter (α) used in the entropy calculation.

findlang: This program requires as arguments a path for a directory containing one or more files, each one will be used to train a model for a language, corresponding to the file name, for example **english.txt**; and the path of the text under analysis. Similar to the *lang* program it is necessary to pass the context size (k) and the smoothing parameter (α).

locatelang: This program has three mandatory arguments and four optional ones. First, it is required to pass the path for a directory containing one or more files, each one will be used to train a model for a language. As in *findlang* the name of the file represents the language. Then, the file under analysis is passed in order to find the language. The last argument is the smoothing parameter (α).

The optional arguments are as follows:

- -b or buffer: The value of this argument changes the number of consecutive characters required in order to change the language detected. The default value is 10.
- -k or context_size: If this argument is used, the program will use a single finite context model with order as the value of the argument for each language. By default, the program uses multiple contexts with different orders for each language.
- -accuracy: When provided, the program reads the first line of the file under analysis, which contains the real “locations” of each language in the target text, in order to calculate the accuracy of the model. The line has to follow this format: *<language>:<location>;<language>:<location>...*
- -print: When provided, prints the file with the locations of the languages detected by the model. If the accuracy flag is passed, the segments of the text where the model failed are represented in red.

3. Functions

check_alphabet

Method *check_alphabet* is used to calculate the number of different characters in a text file.

This function is used in *findlang* and *locatelang* programs, where the alphabet size used in the entropy calculation is not the number of different characters in the respective language model, but the number of different characters in the text file under analysis. This way, the alphabet size of a language will not interfere with the entropy calculation.

letter_entropy

This function calculates the entropy of a single character given a certain context, a smoothing parameter and the number of different symbols the alphabet has ($N_{symbols}$).

First is checked whether the given context exists in the model. If it does not exist entropy is maximum, $-\log_2(1/N_{symbols})$. If it exists it is verified if there are any occurrences of the given character immediately after the context. In an affirmative case the entropy is given by, $-\log_2((N_{occurrences} + a) / (N_{context} + a * N_{symbols}))$ where $N_{occurrences}$ is the number of times the character appears immediately after the context and $N_{context}$ is the total of times the context appears. When $N_{occurrences} = 0$, the entropy is given by, $-\log_2(a / (N_{context} + a * N_{symbols}))$.

get_numbits

get_numbits returns the average number of bits needed in order to represent each character present in a file t , passed as argument, using a model trained with another file. To achieve this the function iterates over each character of t , adding the entropy using the *letter_entropy* function to a total entropy. In the end the total entropy is divided by the number of characters of t .

locatelang

Method *locatelang* receives as arguments a list consisting of structures containing three models with different context orders for each language, a file, α and the value of the buffer size, N_{buffer} .

The function iterates over each character of the file and calculates a weighted entropy for each language. This is done by summing the results of the *letter_entropy* function using the three models.

The function considers a language change in the file when at least N_{buffer} consecutive letters have the minimum weighted entropy for a certain language different from the previous one. As sometimes languages share some words in common and even import words from other languages, this buffered strategy prevents an overly sensitive language locator.

locatelang_k

locatelang_k method is similar to *locatelang* with the exception of using only one model to represent each language. The order of the model is passed as an argument.

4. Results

4.1. *lang*

lang program is the simplest program but it is important that it functions appropriately because it will serve as the basis for the *findlang* program.

In order to verify if the *lang* program is correctly calculating the number of bits required to compress a file, tests were done where the text representing the model and the text under analysis were the same. If the calculations were correct the number of bits should be the same as the entropy of the model which is obtained using the *fcm* program. The results are as follows:

Example 4.1.1 German:

```
./lang ../models-tiny/german.txt ../models-tiny/german.txt 3 0
```

2.129250

```
./fcm german 3 0.000000001
```

2.12925

Example 4.1.2: Greek:

```
./lang ../models-tiny/greek.txt ../models-tiny/greek.txt 3 0
```

1.881061

```
./fcm greek 3 0.000000001
```

1.88108

Another test was performed to see if the number of bits was lower for a file with the same language as the model, compared with one of a different language. As we can see in example 4.1.3 using an english text as a model the minimum number of bits required to compress another english text is lower than those required for a portuguese one. However, as expected, this value is not as low as the entropy of the model.

Example 4.1.3 English vs Portuguese:

```
./fcm english 3 0.000000001
```

2.01617

```
./lang ../models-tiny/english.txt ../tests-final/eng.txt 3 0.001
```

4.011566

```
./lang ../models-tiny/english.txt ../tests-final/por.txt 3 0.001
```

7.33779

4.2. *findlang*

Before going into the results of the *findlang* program, it is necessary to point out that, for non-existing contexts in the trained models, the entropy will be lower in languages with smaller alphabet cardinality (for example english), leading to some wrong language identification of texts. To solve this problem, the used cardinality depends on the text under analysis, normalizing the maximum entropy between all models.

Using the *findlang* program, some results were generated in order to understand which variables would influence the performance of the program. Observing the examples below, we can see that this program accurately identifies the language in which the text under analysis is written, and also identifies similar languages with high ranks (latin languages in example 4.2.1). We can also observe that higher context sizes lead to a bigger difference in entropy between the first and second ranked languages, while the order remains very similar. As expected, by increasing the value of alpha, we get more uniform values between the different languages, which may lead to misidentification for high values of alpha.

Example 4.2.1: Spanish (k=3, a=0.01)

Top nearest languages:		...
#1	spanish: 3.012 avg bits	#13 hungarian: 6.848 avg bits
#2	catalan: 4.287 avg bits	#14 dutch: 6.922 avg bits
#3	portuguese: 4.801 avg bits	#15 slovak: 6.965 avg bits
#4	italian: 5.574 avg bits	#16 polish: 6.990 avg bits
#5	french: 5.790 avg bits	#17 german: 7.002 avg bits

Example 4.2.2: Portuguese (k=3, a=0.01)

Top nearest languages:		...
#1	portuguese: 3.326 avg bits	#13 hungarian: 6.790 avg bits
#2	spanish: 4.743 avg bits	#14 slovak: 6.819 avg bits
#3	catalan: 5.094 avg bits	#15 polish: 6.914 avg bits
#4	italian: 5.763 avg bits	#16 german: 7.011 avg bits
#5	french: 5.913 avg bits	#17 dutch: 7.144 avg bits

Example 4.2.3: Spanish (k=1, a=0.01)

Top nearest languages:		...
#1	spanish: 3.572 avg bits	#13 dutch: 5.471 avg bits
#2	catalan: 3.791 avg bits	#14 serbian: 6.089 avg bits
#3	portuguese: 4.029 avg bits	#15 greek: 6.754 avg bits
#4	french: 4.471 avg bits	#16 ukrainian: 7.682 avg bits
#5	italian: 4.533 avg bits	#17 russian: 8.450 avg bits

Example 4.2.4: Spanish (k=3, a=0.5)

Top nearest languages:

#1 spanish: 3.205 avg bits
#2 catalan: 4.276 avg bits
#3 portuguese: 4.487 avg bits
#4 italian: 4.982 avg bits
#5 french: 5.067 avg bits

...
#13 polish: 5.985 avg bits
#14 serbian: 6.352 avg bits
#15 greek: 6.364 avg bits
#16 ukrainian: 6.397 avg bits
#17 russian: 6.405 avg bits

An experiment was conducted to check if this program could identify major linguistic groups. For this, *findlang* was executed with each training text as the target under analysis, and the resulting values of average bits for each model was stored and used to design a matrix where the edge weights are proportional to entropy between the two corresponding languages.

In Figure 1, the results of this experiment are shown, in which we can see similar languages close to each other, allowing the easy identification of different linguistic groups such as latin languages in red, germanic languages in green and eastern slavic languages in blue.

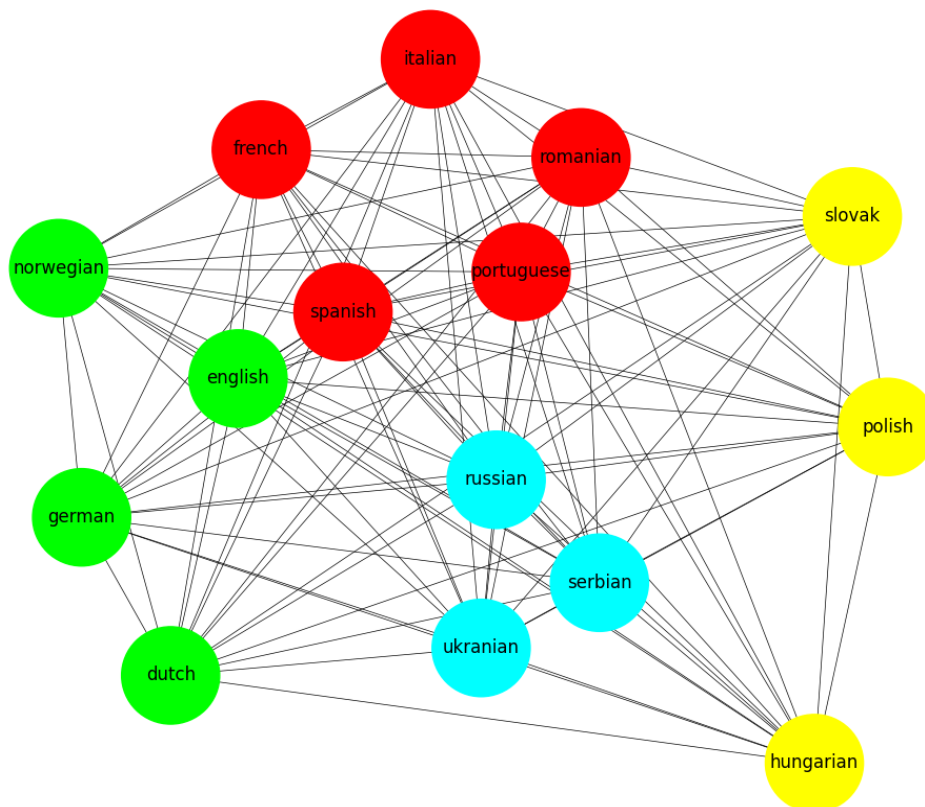


Figure 1: Linguistic groups graph

It was also possible to observe that, for languages not included in the training models, similar languages were identified, as can be seen in example 4.2.5, with the identification of Spanish and Portuguese as the closest languages to the text in galician.

Example 4.2.5 Galician (k=3, a=0.01)

Top nearest languages:

#1	spanish: 4.328 avg bits	...	#13	slovak: 6.763 avg bits
#2	portuguese: 4.535 avg bits		#14	dutch: 6.955 avg bits
#3	catalan: 5.007 avg bits		#15	polish: 6.957 avg bits
#4	italian: 5.490 avg bits		#16	hungarian: 6.973 avg bits
#5	french: 6.045 avg bits		#17	german: 7.105 avg bits

This result led us to conduct a simple experiment for other languages not included in our training models, as to make a linguistics study of these languages. This was achieved by running *findlang* for the languages in study, and normalizing the values of average bits between 0 and 1 for each language included in the axes of the radar graph depicted in Figure 2.

In the graph below, we can see that, for example, Galician has high proximity with Portuguese and Spanish, as expected, and some similarity with Italian; and that Danish has a high proximity with Norwegian and some influences from Dutch, German and English.

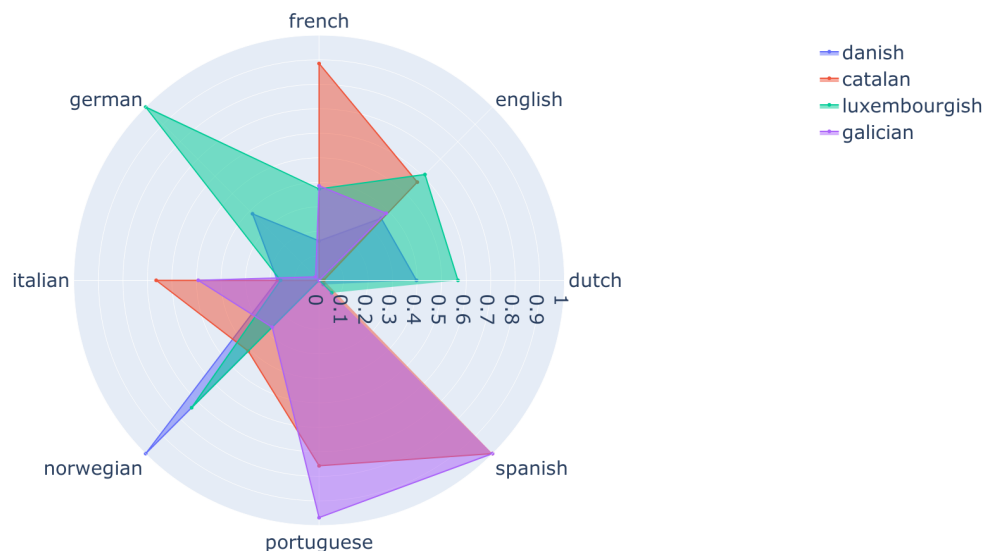


Figure 2: Language proximity graph

4.3. locatelang

In the *locatelang* program, several texts with their language segments correctly labeled were tested. This way, we could calculate the model accuracy for the language segments it had returned, although this accuracy will obviously depend on the size of the text and on the number of language segments.

By using the folder */models-tiny* as our collection of language models, and the file *multi_test2.txt* as our text under analysis, we retrieved the following results for models with different orders.

Model with order 3

Input:

```
./locatelang models-tiny tests/multi_test2.txt 0.0001 -k 3 -b 4 --print --accuracy
```

Output:

```
Real locations received:
  english: 0
  portuguese: 135
  spanish: 300
  slovak: 367
Locations detected:
  english: 0
  spanish: 168
  portuguese: 175
  spanish: 182
  portuguese: 233
  spanish: 363
  slovak: 386

<english:0>... to support civil society, to
support the interim transitional
national council and Mr Jibril (we have
had many meetings with him); Votei a
favor deste relatório, po<spanish:168>rque
ac<portuguese:175>redito <spanish:182>que o Tribunal de Contas pode oc
beneficiar da experiê<portuguese:233>ncia de Hans
Gustaf Wessberg na gestão financeira
das instituições. Presentar sus
observaciones en la resolución que
figura más ab<spanish:363>ajo; Aby sa dosiahla
k<slovak:386>omplexná kvalitná dohoda, ktorá siaha
nad rámeč

Accuracy: 61.04 %
```

Figure 3: Locatelang for multi_test2.txt with a model of order 3 and a buffer size of 4

Model with order 4

Input:

```
./locatelang models-tiny tests/multi_test2.txt 0.0001 -k 4 -b 4 --print --accuracy
```

Output:

```
Real locations received:
  english: 0
  portuguese: 135
  spanish: 300
  slovak: 367
Locations detected:
  english: 0
  portuguese: 137
  spanish: 168
  portuguese: 186
  spanish: 347
  ukrainian: 374
  slovak: 397

<english:0>... to support civil society, to
support the interim transitional
national council and Mr Jibril (we have
had many meetings with him); Vo<portuguese:137>tei a
favor deste relatório, po<spanish:168>rque
acredito que <portuguese:186>o Tribunal de Contas pode
beneficiar da experiência de Hans
Gustaf Wessberg na gestão financeira
das instituições. Apresentar sus
observaciones en la resolución q<spanish:347>ue
figura más abajo; Aby sa<ukranian:374> dosiahla
komplexná kv<slovak:397>alitná dohoda, ktorá siaha
nad rámeč
Accuracy: 78.15 %
```

Figure 4: Locatelang for multi_test2.txt with a model of order 4 and a buffer size of 4

Model with order 5

Input:

```
./locatelang models-tiny tests/multi_test2.txt 0.0001 -k 5 -b 4 --print --accuracy
```

Output:

```
Real locations received:
  english: 0
  portuguese: 135
  spanish: 300
  slovak: 367
Locations detected:
  english: 0
  ukrainian: 134
  portuguese: 138
  spanish: 315
  ukrainian: 374
  slovak: 379
  ukrainian: 390
  slovak: 395
  ukrainian: 432
  slovak: 438

<english:0>... to support civil society, to
support the interim transitional
national council and Mr Jibril (we have
had many meetings with him); <ukranian:134> Vot<portuguese:138>ei a
favor deste relatório, porque
acredito que o Tribunal de Contas pode
beneficiar da experiência de Hans
Gustaf Wessberg na gestão financeira
das instituições. Apresentar sus
<spanish:315>observaciones en la resolución que
figura más abajo; Aby sa<ukranian:374> dosi<slovak:379>ahla
kompl<ukranian:390>exná <slovak:395>kvalitná dohoda, ktorá siaha
nad ráme<ukranian:432>c<slovak:438>
Accuracy: 90.54 %
```

Figure 5: Locatelang for multi_test2.txt with a model of order 5 and a buffer size of 4

As we can see from the above results, the context size has a great influence in the model predictions. But having only one context size can be restrictive. A better approach would be to use combinations of several finite-context models, with different context sizes, to represent each language, and make the average between them.

We decided to use three models, with orders three, four, and five. By making the **average** between the three, we were able to increase our accuracy to **91.22 %**.

However, as we noticed that a context size of 5 had better results than the other, we increased the **weight** for the model with order five, having a weight of 0.6 for this model and 0.2 for the remaining. The resultant accuracy was **93.47 %**.

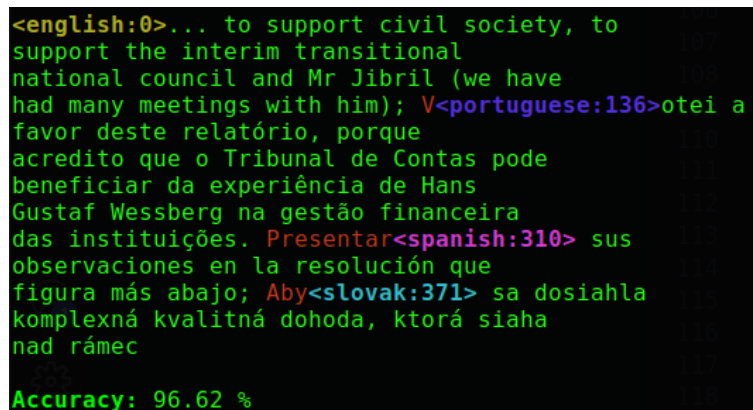
Augmenting the importance of the model with order five had no longer more effect, as the accuracy remained the same. Still, there were some segments where the model was confusing Portuguese with Spanish, so we decided to increase the **sensibility** of the model by incrementing the size of the **buffer**. The following output was the end results.

Model with orders 3, 4 and 5

Input:

```
./locatelang models-tiny tests/multi_test2.txt 0.0001 -b 5 --print --accuracy
```

Output:



```
<english:0>... to support civil society, to  
support the interim transitional  
national council and Mr Jibril (we have  
had many meetings with him); V<portuguese:136>otei a  
favor deste relatório, porque  
acredito que o Tribunal de Contas pode  
beneficiar da experiência de Hans  
Gustaf Wessberg na gestão financeira  
das instituições. Apresentar<spanish:310> sus  
observaciones en la resolución que  
figura más abajo; Aby<slovak:371> sa dosiahla  
komplexná kvalitná dohoda, ktorá siaha  
nad rámec  
Accuracy: 96.62 %
```

Figure 6: Locatelang for multi_test2.txt with a model of combined orders and a buffer size of 5

There was still a large segment where the model confused Spanish with Portuguese, but this was comprehensible, as the word “Presentar” is very similar to the Portuguese language. Thus, this was the best outcome we got from this text.

4.4. Other Applications

The programs built can also be used for different problems other than language detection. In this section it is presented some of the problems explored.

spam detection

Spam detection is a task present in the majority of mail or sms services. It consists of classifying text as spam or ham.

A slightly modified *findlang* program was used as a spam detector.

Two models were tested using the SMS Spam Collection Dataset¹. The dataset was divided in model and testing sets. The testing set consists of 100 ham examples and 100 spam examples while the model set has the remaining examples in the dataset.

findlang was slightly modified in order to go through multiple test files and output the accuracy, percentage of true positives, true negatives, false positives and false negatives.

As we can observe from the examples below the results obtained were quite good with the best accuracy reaching 97,50% and the lowest being above 93%.

Example 4.4.1: Spam detection (k=1, a=0.01)

True positive	49.00 %
True negative	47.00 %
False positive	3.00 %
False negative	1.00 %
Accuracy	96.00 %

Example 4.4.2: Spam detection (k=2, a=0.01)

True positive	50.00 %
True negative	48.00 %
False positive	2.00 %
False negative	0.50 %
Accuracy	97.50 %

Example 4.4.2: Spam detection (k=2, a=1)

True positive	50.00 %
True negative	46.50 %
False positive	3.50 %
False negative	0.50 %
Accuracy	96.00 %

¹ SMS Spam Collection Dataset, available on:
<https://www.kaggle.com/uciml/sms-spam-collection-dataset>

Example 4.4.3: Spam detection ($k=4$, $a=0.01$)

True positive	49.00 %
True negative	44.50 %
False positive	5.50 %
False negative	1.00 %
Accuracy	93.50 %

This problem privileges small contest sizes, two being the ideal size, this can mean that some single characters are important alone (\$, €, etc) in order to classify a text as spam.

virus genome comparison

In theory, the finite context models can also be used to compare the genomes across different viruses. To test this, we collected the genomes of 11 viruses from the National Center for Biotechnology Information², according to their family lineage, expecting to have similarities between viruses that were closely related by lineage, and vice-versa.

The genomes collected were mainly from families related to the various Human Coronaviruses. We also collected genomes from Influenza viruses, which are viruses that cause the flu, being Influenza A and B viruses the most important ones.

Figure 7 shows the lineage of the viruses collected. The viruses used for comparison are in bold.

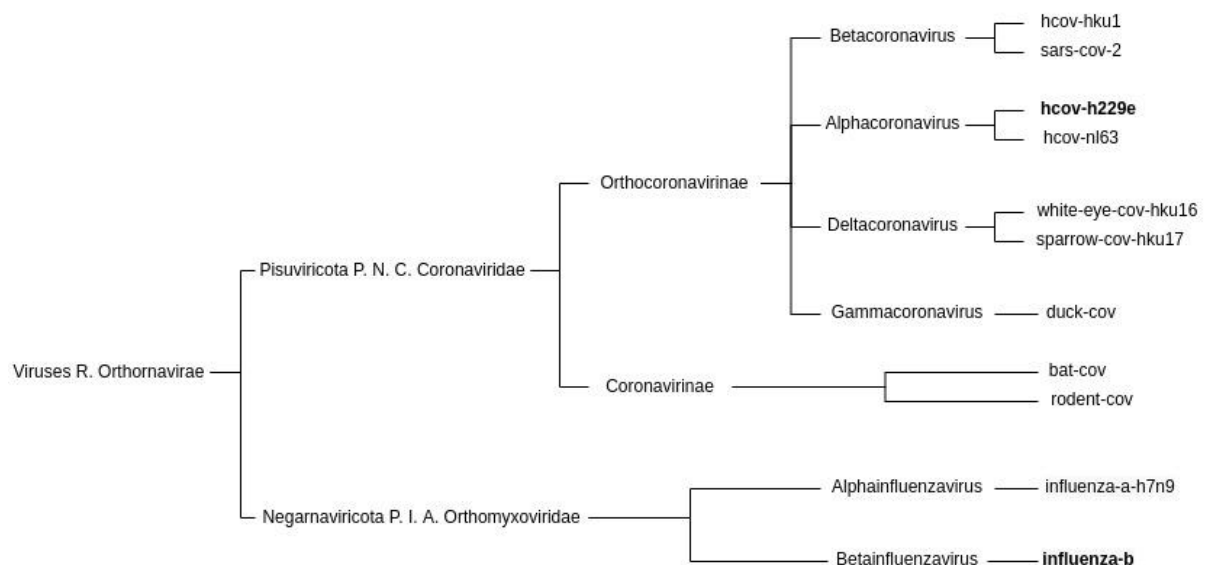


Figure 7: Family lineage of the virus collected.

² NCBI - WWW Error Blocked Diagnostic. (2020). NCI Genome.
<https://www.ncbi.nlm.nih.gov/genome>

To assess the viruses genomes that are more similar to a certain genome, we used the findlang program with a model with order six, slightly higher to better describe and distinguish the genomes, but lower enough, so that the model is not too sparse. Essentially, a context size that gives models with an average lower entropy.

Example 4.4.4 hcov-h229e (k=6, a=0.001)

Top nearest languages:		...	
#1	hcov-h229e: 1.259 avg bits	#7	duck-cov: 4.942 avg bits
#2	influenza-b: 3.506 avg bits	#8	sars-cov-2: 4.979 avg bits
#3	influenza-a-h7n9: 3.583 avg bits	#9	bat-cov: 5.006 avg bits
#4	hcov-hku1: 4.738 avg bits	#10	white-eye-cov-hku16: 5.077 avg bits
#5	hcov-nl63: 4.738 avg bits	#11	sparrow-cov-hku17: 5.443 avg bits

In example 4.4.4, we can see that surprisingly, this human coronavirus has a lot of similarities with the flu viruses, followed by the other human coronavirus. The least related viruses appear to be the ones from Deltacoronavirus.

Example 4.4.5 influenza-b (k=6, a=0.001)

Top nearest languages:		...	
#1	influenza-b: 0.016 avg bits	#7	sparrow-cov-hku17: 3.273 avg bits
#2	influenza-a-h7n9: 2.680 avg bits	#8	rodent-cov: 3.357 avg bits
#3	bat-cov: 3.200 avg bits	#9	duck-cov: 3.368 avg bits
#4	hcov-hku1: 3.211 avg bits	#10	white-eye-cov-hku16: 3.399 avg bits
#5	hcov-nl63: 3.222 avg bits	#11	sars-cov-2: 3.504 avg bits

In example 4.4.5, with a flu virus as comparison, the most related virus besides itself, is the Influenza A, which makes sense when looking into their lineage. In fact, if we use the locatelang program with this virus, it finds many segments similar to the Influenza A genome. This is represented in the above.

Input:

```
./locatelang models-virus models-virus/influenza-b.txt 0.0001 -b 3 --print
```

Output:

```
<influenza-b:0>ggcagaagca cagcatttc ttgtgagctt cgagcactaa taaaactgaa aatcaaaatg
tccaacatgg atattgacag tataaatacc ggaacaatcg ataaaacacc agaagaactg
actccccgaa ccagtggggc aaccagacca atcatcaagc cagcaaccct tgctccgcca
agcaacaaac gaacccgaaa tccatctcca gaaaggacaa ccacaagcag tgaaccgat
atcggaagga aaatccaaaa gaaacaaacc ccaacagaga taaagaagag cgtctacaaa
atgggtggtaa aactgggtga attctacaac cagatgatgg tcaaagctgg acttaagtat
gacatggaaa ggaatctaatt tcaaaatgca caagctgtgg agagaatcct attggctgca
actgatgaca agaaaactga ataccaaaag aaaaggaatg ccagagatgt caa<influenza-a-h7n9:520>agaagg<influenza-b:526>g
aagggaagaaa tagaccacaa caagacagga ggcacctttt ataagatggt aagagatgat
aaaaccatct acttcagccc tataaaaatt acctttttaa aagaagaggt gaaaacaatg
tacaagacca ccatggggag tgatggtttc agtggactaa atcaccattat gattggacat
tcacagatga acgatgtctg ttccaaga tcaaagggac tgaaaagggt tggacttgac
ccttcattaa tcagtacttt tgccggaagc acactaccca gaagatcagg tacaactggt
gttgcaatca aaggagggtg aactttagt gatgaagcca tccgatttat aggaagagca
atggcgagaca gagggg<influenza-a-h7n9:941>tact gagagaca<influenza-b:954>tc aaggccaaga cggcctatga aaagatttctt
ctgaatctga aaaacaagt ctctcgccg caacaaaagg ctctagttag tcaagtatc
ggaagtagga acccagggat tgcagacata gaagacctaa ctctgcttgc cagaagcatg
gtagtgtgca gacctctgt agcgagcaaa gtggtgcttc ccataagcat ttatgctaaa
atacctcaac taggattcaa taccgaagaa tactctatgg ttgggtatga agccatgctt
ctttataata tggcaacacc tgtttccata ttaagaatgg gagatgacgc aaaagataaa
tctcaactat tcttcatgtc gtgcttcgga gctgcctatg aagatctaag agtggtatct
gcactaacgg gcaccgaatt taagcctaga tcagcactaa aatgcaaggg tttccatgctc
ccggctaagg agcaagtaga aggaatgggg gcagctctga tgtccatcaa gcttcagttc
tgggcccaa tgaccagatc tggagggaat gaagtaagt gagaaaggag gtctgggtcaa
ataagttgca gccctgtgtt tgcagtagaa agacctattg ctctaagcaa gcaagctgta
agaagaatgc tgtcaatgaa cgttgaagga cgtgatgcag atgtcaagg aaatctactc
aaaaatgatga atgattcaat ggcaa<influenza-a-h7n9:1743>agaaa accagtgga a<influenza-b:1761>tgctttcat tgggaagaaa
atgtttcaaa tatcagacaa aaacaagtc aatccattg agattccaat taagcagacc
atcccaatt tcttcttgg gagggacaca gcagaggatt atgatgacct cgattattaa
agcaataaaa tagacactat ggctgtgact gtttcagtac gtttgggaag tgggtgttta
ctcttattga aataaatgta aaaaatgctg ttgtttctac t
```

Figure 8: Locatelang for influenza-b.txt with a model of combined orders and a buffer size of 3

For buffer sizes higher than three, the model is less sensitive, so the segments found are fewer. In fact, for a buffer size of 5, the program fails to find any segment other than Influenza B.

5. Problems and Future Work

Languages with special characters, this is, characters that are represented by more than one byte, make our model less accurate. This is because our context is an array of characters, each one with one byte, meaning that for “single-byte” languages, such as English, it would have space enough to store 5 letters, but for another one, such as Russian, it would only have space to store about 1,66 letters.

Examples of UTF-8 encoding

Character	Binary code point	Binary UTF-8	Hex UTF-8
\$	U+0024	010 0100	24
£	U+00A3	000 1010 0011	C2 A3
₹	U+0939	0000 1001 0011 1001	E0 A4 B9
€	U+20AC	0010 0000 1010 1100	E2 82 AC
한	U+D55C	1101 0101 0101 1100	ED 95 9C
о	U+10348	0 0001 0000 0011 0100 1000	F0 90 8D 88

Figure 9: Examples of UTF-8 encoding representation.

Figure 9 demonstrates how special characters are represented in binary. Essentially, the first byte most significant leading 1’s inform the number of following bytes that have to be read to represent the character.

To overcome this, we would simply have to change the methods and the data structures that we used to read and write these special characters, or even use special libraries to perform this. However, this was not yet implemented. To amend this easily, we only had this into consideration whenever we were returning the positions of the segments and printing the results, not when the model was training. Thus, despite being able to recognize the languages very well, as shown in Figure 10, it should be the next focal point in a future work.

<pre> <ru^{ssian}:0>вечного завета между Мною и между землею. От Ханаана родились: «<uk^{ra}nian:64>Юнаки» - всі члени команди повинні бути перший день змагання. A na<sl^{ov}ak:131>koniec, v súvislosti s rastom vplyvu predovšetkým amerického mainstreamu v oblasti kultúrneho vyžitia je nevyhnutné Európskej unii a členským štátom Accuracy: 98.73 % </pre>	<pre> <ru^{ssian}:0>вечного завета между Мною и между землею. От Ханаана родились: «Юнаки» - всі члени команди повинні бути перший день <uk^{ra}nian:117>змагання. A nakoniec, v súvislosti s rastom vplyvu predovšetkým amerického mainstreamu v oblasti kultúrneho vyžit<sl^{ov}ak:230>ia je nevyhnutné Európskej unii a členským štátom Accuracy: 60.15 % </pre>
--	---

Figure 10: Comparison of the results for “multiple-byte” languages after and before taking into consideration special characters.

6. Conclusion

With this work we can conclude that finite-context models are a good approach in order to perform language detection tasks, as they are able to provide an accurate answer quickly with small training models, meaning that results could be even more accurate if more extensive text files are used.

This type of model is also quite versatile as it can be applied to other tasks such as spam detection or genome comparison.

Our programs could be used in systems such as translators, plagiarism detection or other types of text tasks.

Appendix I - Language sources

Models: <https://sourceforge.net/projects/la-strings/files/Language-Data/>

Test file brazilian:

<https://autoesporte.globo.com/tecnologia/noticia/2021/12/bmw-vai-apresentar-carro-que-muda-de-cor-ao-toque-de-um-botao-na-ces.ghml>

Test file arabic:

<https://www.alarabiya.net/medicine-and-health/2021/12/27/3-%D8%A3%D9%86%D9%88%D8%A7%D8%B9-%D9%85%D9%86-%D8%A7%D9%84%D9%81%D9%88%D8%A7%D9%83%D9%87-%D8%AA%D9%86%D8%A7%D9%88%D9%84%D9%87%D8%A7-%D8%A8%D8%B5%D9%81%D8%A9-%D9%8A%D9%88%D9%85%D9%8A%D8%A9-%D9%84%D8%AA%D8%AD%D8%B3%D9%8A%D9%86-%D8%A7%D9%84%D9%87%D8%B6%D9%85>

Test file mandarin:

<http://cn.chinadaily.com.cn/a/202112/23/WS61c43f3ba3107be4979fea8e.html>

Test file catalan:

https://www.ara.cat/cultura/model-llengua-l-escola-possible_130_4215716.html

Test file danish:

<https://www.berlingske.dk/oplevelser/hop-i-havnen-og-spis-en-varm-skaal-muslinger-her-er-ti-gode-oplevelser>

Test file english:

<https://www.theguardian.com/world/2021/dec/28/other-surfers-respect-me-the-92-year-old-still-riding-waves-in-new-zealand>

Test file galician:

<https://galego.lavozdegalicia.es/noticia/deportes/2021/12/26/firouzja-amenaza-carlsen/00031640543775415200400.htm>

Test file portuguese:

<https://www.dnoticias.pt/2021/12/28/290511-alimentos-para-criancas-com-acucar-ou-sal-adicionados-segundo-estudo-do-insa/>

Test file spanish:

<https://elpais.com/espana/un-futuro-cercano/2021-12-02/el-baix-llobregat-navega-hacia-la-reconversio-n-de-su-industria-y-la-digitalizacio-n-de-sus-cultivos.html>

Test file Slovak:

<https://www.aktuality.sk/clanok/4pt4nhq/v-mjanmarsku-odlozil-sud-dalsi-verdikt-v-procese-so-su-tij/>