TAI 2020-21

Projeto 2

Tema: **Identificador de línguas**
Autores: **Mário Silva nºmec 93430, Daniel Gomes nºmec 93015, João Magalhães nºmec 79923**
Date: **28/12/2021**

Index

1 Introdução

Este trabalho consiste em considerar o problema de determinar a ”similaridade” entre um texto alvo, *t*, e alguns textos de referência, *ri*, isto é, cada *ri* pode ser um texto de amostra de um determinado idioma e *t* um texto alvo cujo idioma precisa de ser determinado.

Normalmente esta tarefa de reconhecimento de texto trata-se de um problema de classificação, envolvendo o processo de extração de *features* e treino de um modelo. No entanto, neste projeto pretende-se obter uma aproximação às linguagens de referências através de *Finite Context Models* e, com estes, comparar o número de bits necessários à compressão do texto alvo, determinar o(s) idioma(s) presente(s).

2 Objetivos

Este trabalho tem como um dos objetivos a identificação da língua de um determinado texto, para isso é necessário construir modelos representativos de cada língua e comparar o quão adequado é cada modelo para representar/comprimir esse texto. Outro objetivo do trabalho é processar um texto contendo segmentos escritos em idiomas diferentes e retornar a posição do caracter em que começa o segmento, bem como o idioma onde o segmento foi escrito.

3 Implementação

Nesta secção iremos abordar como realizamos a nossa implementação, que organização utilizamos, os módulos e as estruturas de dados desenvolvidas, bem como a estratégia utilizada.

3.1 Organização

Na figura abaixo é possível visualizar toda a organização do trabalho desenvolvido. Na pasta ***src***, encontram-se presente os módulos de código criados, enquanto que na pasta ***datasets*** estão presentes ficheiros de texto utilizados representativos de cada língua para a recolha de informação feita pelo identificador de línguas. Além disto, o documento ***README.md*** apresenta as instruções de como correr os programas elaborados. Finalmente, o ficheiro ***report.pdf*** representa este documento, o relatório de trabalho produzido.

```
Language Identifier
├── src
│   ├── fcm.py
│   ├── findlang.py
│   ├── lang.py
│   ├── locatelang.py
│   ├── main.py
│   └── utils.py
├── datasets
├── report
│   └── report.pdf
└── README.md
```

3.2 Módulos Desenvolvidos

3.2.1 FCM

Este módulo foi retirado do trabalho previamente elaborado, pelo que não foram realizadas quaisquer alterações ao seu funcionamento ou implementação. Relembrando o seu principal objetivo, o modelo de contexto finito permite analisar sequências de caracteres presentes numa fonte de informação (um ficheiro de texto) e como estas estão organizadas, e assim obter as suas probabilidades de ocorrerem.

3.2.2 Lang

O Módulo *Lang*, tem como principal objetivo calcular o número de bits necessários para comprimir um texto *t*, tendo como referência um modelo gerado por outro texto *ri*. Para conseguir gerar este modelo, recorre-se ao módulo FCM referido na subsecção anterior.

De notar que o texto alvo, ao qual calculamos o número de bits necessários para a compressão, poderá conter caracteres que não estão presentes na tabela de ocorrências/probabilidades do modelo, impedindo alcançar o objetivo pretendido. Assim, através do método ***merge_alphabets****()*, é feita uma primeira leitura ao ficheiro de texto alvo, onde se adicionam ao alfabeto do texto de referência os símbolos até então desconhecidos por este, e consequentemente, atualizando a tabela de ocorrências/probabilidades.

Seguidamente, o ficheiro de texto alvo é lido novamente, onde através da tabela de probabilidades armazenada no modelo do texto de referência, poderemos então calcular o número de bits de cada símbolo, segundo um determinado contexto, recorrendo à formula $-log_2 P(e|c)$.

3.2.3 FindLang

O módulo *FindLang*, através de *Lang*, desempenha o papel de identificar a língua presente num texto alvo, por um conjunto de ficheiro de texto de línguas diferentes como referência. O primeiro passo consiste

na instanciação de objetos *Lang*, onde o texto de referência em cada um será obtido do conjunto referido anteriormente. Assim, poder-se-á calcular o número de bits necessários para comprimir o texto alvo, segundo o texto representativo de cada língua.

Finalmente, o objeto *Lang* que obteve o menor número de bits de todo o conjunto de línguas, tem o texto de referência representativo da língua presente no texto alvo. Isto pode ser justificado pelo facto de que menos bits necessários para a compressão está relacionado com uma maior probabilidade de ocorrência dos caracteres do texto alvo, segundo um contexto, para o modelo do texto de referência em causa.

3.2.3.1 Otimizações e Funcionalidades Extra

Neste módulo foi também implementada uma funcionalidade extra e uma otimização. Foi criada uma opção para ler também de um diretório ficheiros alvos, e para cada um deles correr o *FindLang*. Desta forma, os objetos *Lang* que têm um objeto *FCM*, já com as tabelas de probabilidades construídas com os ficheiros de referência, podem ser reaproveitados e desta forma facilitar os testes.

Uma otimização efetuada foi também a obtenção do alfabeto do texto alvo (ou de todos os textos alvos se for com a implementação descrita acima) primeiramente, e depois este é passado para os objetos *Lang*, aumentando a rapidez de execução do programa, pois assim estes objetos não terão de ir ler novamente o texto alvo para obter o alfabeto.

3.2.4 LocateLang

Com vista a chegar ao último objetivo proposto, ou seja, identificar num ficheiro com segmentos de texto com línguas diferentes, a língua presente em cada um, o módulo *LocateLang* foi criado.

Durante o processo de implementação tivemos de tomar algumas decisões para alcançarmos uma solução que fosse não só precisa, mas de alguma forma também eficiente.

Assim, duas estratégias foram implementadas com base no tamanho do ficheiro de texto alvo: em ficheiros cujo tamanho seja superior a 2 MB é utilizado um método de localização da língua em cada segmento com base em ”blocos”, e em caso contrário com base em ”janelas”. Estes métodos serão seguidamente explicados.

3.2.4.1 Estratégia com base em ”Blocos”

Na estratégia com base em blocos, cada segmento de texto analisado terá um tamanho definido previamente, o tamanho de cada bloco, onde recorrendo às funcionalidades do módulo *Lang*, identifica-se a língua em causa, recorrendo ao mínimo valor de bits de entre todas as línguas, obtendo-se assim o número de bits necessários para a compressão. Caso a proporção entre o número de bits calculados para o bloco em causa e o bloco anterior seja superior a um *threshold* previamente definido, assumimos que houve uma mudança de língua representada no texto para o bloco em causa. Consequentemente, ir-se-á repetir o processo de identificação da língua neste bloco, excluindo a língua previamente identificada. Contudo, é importante notar que a utilização deste método apesar de ser bastante mais rápido irá devolver resultados menos precisos, já que apenas se pode identificar onde uma nova língua surge no início e fim de cada bloco, e nunca em posições internas do bloco em causa. Além disto, a definição dos valores dos *thresholds* e do tamanho de cada bloco tiveram como base a realização de testes que irão ser apresentados posteriormente.

3.2.4.2 Estratégia com base em ”Janelas”

Para o método com base em janelas, o texto alvo é lido sequencialmente, em segmentos do tamanho da janela definida, e movendo a janela apenas uma posição para a frente após cada leitura e processamento.

O processamento de cada janela consiste no cálculo de número de bits necessários para a compressão da janela, e obter a média de bits necessários tendo em conta o tamanho da ”janela”. Tendo este valor da média de bits é preciso ter um *threshold* definido para saber se esse número de bits é pequeno o suficiente para definir a janela como sendo da língua usada como referência.

Este valor de *threshold* pode ser definido através do valor da entropia dos *FCM*s de cada língua, ou também ter em conta o tamanho do alfabeto da língua em causa, como irá ser demonstrado posteriormente na secção dos resultados obtidos. Para o tamanho da janela, obtivemos melhores resultados utilizando uma janela 8 vezes superior ao máximo *k* passado como argumento.

Após o processo de identificação de línguas é realizado um cálculo das posições que possuem as mesmas línguas identificadas de forma sequencial através do método *merge_locations*, ou seja, se desde uma posição *x* até *y*, onde existem várias ”janelas”, foram sempre reconhecidas as mesmas linguagens, é feita uma junção destas mesmas para esse dado intervalo.

3.2.4.3 Método *compare_lang*

Este método é bastante semelhante ao anterior, com base em ”janelas”, mas recorrendo à utilização de um *threshold* diferente.

O *threshold* determinado neste método é com base na média do número de bits de todas as línguas em todas as ”janelas”. De seguida, definiu-se também um valor (que para os casos testados deu melhores resultados) para que não fosse apenas considerado as línguas que necessitaram de menos bits do que o valor médio de bits, mas que fosse pelo menos 40% menor que esse valor médio. Esta percentagem é um valor ”*hardcoded*” em que há probabilidade de não funcionar como desejado quando são utilizadas outras línguas de referências. Contudo, foi o valor que melhor se ajustou para os casos de uso testados.

Desta forma, permite-se obter uma comparação entre os diferentes idiomas, no entanto, é um método bastante *greedy* pois poderá haver línguas que, naturalmente, necessitem de mais bits para a compressão de textos desse mesmo idioma.

3.2.4.4 Funcionalidades Extra

Uma funcionalidade extra foi a possibilidade de fazer este processamento em janelas usando diferentes valores de *k* para o objeto *FCM*. Nesta estratégia o processo é igual, mas para cada janela é calculada a média do número de bits necessários para a sua compressão usando *FCM*s com o mesmo texto de referência e com *k*s diferentes, e consequentemente, com a utilização desta funcionalidade é necessário mudar também os *thresholds* para uma média das entropias das *FCM*s com diferentes *k*s de cada idioma.

3.2.5 Main

Como é percetível pelo nome, este módulo representa o programa principal, onde é fornecido um leque de opções aos utilizadores, com vista ao que pretendam desempenhar dentro do programa. A listagem abaixo representa todas as opções possíveis de se realizarem:

- *-r*, o nome do ficheiro de referência a ser utilizado;

- *-d*, o nome de um diretório, com diversos ficheiros de referência a serem utilizados pelo módulo *FindLang*;
- *-t*, o nome do ficheiro alvo a ser utilizado;
- *-td*, o nome de um diretório, com diversos ficheiros alvos a serem utilizados pelo módulo *FindLang*;
- *-k*, o valor do tamanho de cada sequência a ser utilizada pelo *FCM*;
- *-a*, o valor de alpha a ser utilizado pelo *FCM*;
- *-l*, a utilização do módulo *LocateLang*;
- *-ta*, ter em conta o tamanho do alfabeto das línguas de referência para o valor do *threshold* de decisão do módulo *LocateLang*;
- *-c*, a utilização do método *compare_lang*;
- *-m*, os valores de múltiplos valores de *k* a serem usados pelo módulo *LocateLang*.

4 Resultados Obtidos

Nesta secção serão apresentados os resultados que obtivemos e testes efetuados com vista à validação da solução desenvolvida.

4.1 Lang

Com vista a testagem deste módulo, corremos o programa utilizando diferentes línguas de referência e alvo onde comparámos não só o número de bits necessários para compressão entre os diversos testes, mas também com um número estimado de bits, de forma a verificar a precisão dos resultados.

Inicialmente utilizámos como língua de referência Português, assim como língua alvo, onde obtivemos os resultados representados nas figuras seguintes (em todos os testes efetuados foi usado um valor de 3 para *k*).

Utilizando um valor de *alpha* igual a 0.1:

```
2021-12-27 15:43:52,851 - Starting to train FCM with Portuguese with order 3
Entropy of reference file 6.775812536350826
Total characters in target file 170274
Estimated number of bits 1153744.7038146006
Number of bits necessary to compress target file with a trained model 370377.4084508035
```

Figure 1: Teste efetuado ao Modulo Lang com *alpha* de 0.1

Utilizando um valor de *alpha* igual a 0.001:

```
2021-12-27 15:25:36,889 - Starting to train FCM with Portuguese with order 3
Entropy of reference file 2.5806854583422316
Total characters in target file 170274
Estimated number of bits 439423.63573376514
Number of bits necessary to compress target file with a trained model 367457.0279768416
```

Figure 2: Teste efetuado ao Modulo Lang com *alpha* de 0.001

De notar que o número estimado de bits representado nas figuras é simplesmente o tamanho do ficheiro alvo a multiplicar pela entropia calculada pelo *FCM*.

Pela observação das figuras conseguimos concluir que o valor final de bits necessários apresenta um desvio enorme quando o *alpha* é um valor maior em comparação a um *alpha* mais pequeno, que apenas apresenta um desvio ligeiro do valor estimado. Isto pode ser explicado porque um *alpha* maior significa que os contextos existentes no ficheiro alvo tem uma probabilidade menor, pois os contextos que nem aparecem no ficheiro de treino são mais contabilizados, ou seja, um *alpha* maior causa a existência de mais ”ruído” na tabela de probabilidades do *FCM*.

De seguida, recorreu-se ao Espanhol como língua de referência, e novamente Português como língua alvo.

```
2021-12-27 15:55:54,832 - Starting to train FCM with Spanish with order 3
Entropy of reference file 2.012238505113251
Total characters in target file 170274
Estimated number of bits 342631.89921965374
Number of bits necessary to compress target file with a trained model 854718.9046186864
```

Figure 3: Teste efetuado ao Módulo Lang utilizando Espanhol como língua de referência

Como era expectável, o número de bits necessários para a compressão é bastante superior ao obtido no teste anterior, já que o modelo foi gerado a utilizar uma língua de referência diferente à do alvo.

Recorrendo ao Árabe como língua de referência, e Português como língua alvo, foram obtidos os resultados na figura abaixo.

```
2021-12-27 15:56:40,264 - Starting to train FCM with Arabic with order 3
Entropy of reference file 9.490674517822335
Total characters in target file 170274
Estimated number of bits 1616015.1128476802
Number of bits necessary to compress target file with a trained model 1648779.5344564249
```

Figure 4: Teste efetuado ao Módulo Lang utilizando Árabe como língua de referência

Observando os valores representados na figura acima, conseguimos percecionar que o valor de bits necessários para a compressão é muito superior tanto no teste utilizando Espanhol como referência e também o Português como referência. A razão deste acontecimento deve-se ao facto de o Espanhol ser uma língua bastante semelhante ao Português, já que ambas derivam do Latim, e consequentemente haverá sequência de caracteres idênticas, diminuindo assim o número de bits necessários. Já o Árabe é uma língua com sequência de caracteres bastantes diferentes, visto que utiliza um alfabeto totalmente diferente, justificando assim o valor enorme obtido de bits estimado.

4.2 FindLang

No contexto do módulo *FindLang*, para efeitos de testagem, foram usados um *alpha* de valor 0.001, *k* de valor 3, e *datasets* diferentes.

Em cada teste é calculada também uma *accuracy* cumulativa, ou seja, o número de línguas previstas corretamente a dividir pelo número de previsões feitas.

Num primeiro teste simples, foram utilizadas 4 línguas diferentes como textos de referência, e apenas um texto alvo contendo Italiano, onde as línguas de treino foram Italiano, Português, Inglês e Espanhol.

```
2021-12-26 16:35:35,670 - Starting to train FCM with files inside ../datasets/languages_reference_small/
2021-12-26 16:35:35,953 - Starting to train FCM with Portuguese with order 3
2021-12-26 16:35:54,339 - Starting to train FCM with English with order 3
2021-12-26 16:36:13,100 - Starting to train FCM with Spanish with order 3
2021-12-26 16:36:33,979 - Starting to train FCM with Italian with order 3

For Italian file:
        - Guessed language: Italian
        - Accuracy: 1.0
```

Figure 5: Teste efetuado ao Módulo FindLang utilizando Italiano como alvo

Pela figura acima representada, consegue-se perceber que a precisão foi de 1 e que a língua ”adivinhada” foi a correta, ou seja, Italiana.

De seguida, para percecionar a semelhança entre línguas, utilizou-se como referência Português, Inglês e Espanhol, e como língua alvo Italiano. A figura seguinte representa os resultados obtidos.

```
2021-12-26 16:40:42,776 - Starting to train FCM with files inside ../datasets/languages_reference_small/
2021-12-26 16:40:43,063 - Starting to train FCM with Portuguese with order 3
2021-12-26 16:40:59,462 - Starting to train FCM with English with order 3
2021-12-26 16:41:18,817 - Starting to train FCM with Spanish with order 3

For Italian file:
        - Guessed language: Spanish
        - Accuracy: 0.0
```

Figure 6: Teste efetuado ao Módulo FindLang sem usar Italiano como referência

Já que Italiano não foi utilizado no conjunto de línguas de referência, a precisão obtida foi de 0 tendo sido identificado Espanhol como sendo a língua mais semelhante ao Italiano.

Para alargar o leque de línguas a serem utilizadas como referência e assim efetuar um teste com um conjunto maior, utilizámos textos com as seguintes línguas como referência: Italiano, Português, Espanhol, Inglês, Japonês, Hindi, Árabe, Polaco, Alemão e Francês. Por outro lado, a língua alvo foi Português. A figura abaixo contém os resultados deste teste.

```
2021-12-26 17:21:17,266 - Starting to train FCM with files inside ../datasets/languages_train/
2021-12-26 17:21:17,577 - Starting to train FCM with Polish with order 3
2021-12-26 17:21:28,939 - Starting to train FCM with Portuguese with order 3
2021-12-26 17:21:38,369 - Starting to train FCM with German with order 3
2021-12-26 17:22:04,307 - Starting to train FCM with French with order 3
2021-12-26 17:22:16,770 - Starting to train FCM with Arabic with order 3
2021-12-26 17:23:04,823 - Starting to train FCM with English with order 3
2021-12-26 17:23:24,810 - Starting to train FCM with Spanish with order 3
2021-12-26 17:23:46,389 - Starting to train FCM with Hindi with order 3
2021-12-26 17:24:32,897 - Starting to train FCM with Japanese with order 3
2021-12-26 17:26:22,806 - Starting to train FCM with Italian with order 3

For Portuguese file:
        - Guessed language: Portuguese
        - Accuracy: 1.0
```

Figure 7: Teste efetuado ao Módulo FindLang com um conjunto alargado de ficheiros de texto como referência

Finalmente, como último teste deste módulo, repetiu-se o conjunto de línguas referência e como alvo utilizou-se um conjunto de ficheiros de textos representativos de cada uma das línguas utilizadas como referência.

```
2021-12-27 16:09:27,965 - Starting to train FCM with files inside ..\datasets\languages_train/
2021-12-27 16:09:28,131 - Starting to train FCM with Arabic with order 3
2021-12-27 16:09:51,463 - Starting to train FCM with English with order 3
2021-12-27 16:10:07,522 - Starting to train FCM with French with order 3
2021-12-27 16:10:17,000 - Starting to train FCM with German with order 3
2021-12-27 16:10:26,414 - Starting to train FCM with Hindi with order 3
2021-12-27 16:10:35,611 - Starting to train FCM with Italian with order 3
2021-12-27 16:10:44,485 - Starting to train FCM with Japanese with order 3
2021-12-27 16:12:13,201 - Starting to train FCM with Polish with order 3
2021-12-27 16:12:23,380 - Starting to train FCM with Portuguese with order 3
2021-12-27 16:13:47,159 - Starting to train FCM with Spanish with order 3

For Arabic file:
        - Guessed language: Arabic
        - Accuracy: 1.0
For English file:
        - Guessed language: English
        - Accuracy: 1.0
For French file:
        - Guessed language: French
        - Accuracy: 1.0
For German file:
        - Guessed language: German
        - Accuracy: 1.0
For Hindi file:
        - Guessed language: Arabic
        - Accuracy: 1.0
For Italian file:
        - Guessed language: Italian
        - Accuracy: 1.0
For Japanese file:
        - Guessed language: Japanese
        - Accuracy: 1.0
For Polish file:
        - Guessed language: Polish
        - Accuracy: 1.0
For Portuguese file:
        - Guessed language: Portuguese
        - Accuracy: 1.0
For Spanish file:
        - Guessed language: Spanish
        - Accuracy: 1.0
```

Figure 8: Teste efetuado ao Módulo *FindLang* com um conjunto alargado de ficheiros de texto como referência e alvo

Pelos resultados da figura representada acima, consegue-se concluir que em qualquer referência a língua ”adivinhada” foi sempre a correta.

4.3 LocateLang

Finalmente, o *LocateLang* foi o último módulo alvo de testes. Neste decidimos testar ambas as estratégias implementadas, e explicadas anteriormente. Para tal em ambas, decidimos utilizar dois ficheiros de texto por nós criados com diversas línguas em alguns dos segmentos de texto presentes.

Um ficheiro de tamanho maior com Inglês até ao caracter 78 162, depois Italiano até ao 128 304 e Português ate 225 422, e outro menor com as mesmas línguas, mas com Inglês até ao caracter 102, Italiano até 265 e Português ate 354.

Como a estratégia em blocos em termos de processamento é mais rápida, o código está dinâmico de modo que a partir de um certo tamanho de ficheiros, utiliza cada uma das estratégias correspondentes.

4.3.1 Estratégia com base em ”Blocos”

Para a estratégia com base em blocos, foram utilizadas como referência ficheiros de texto com as línguas: Italiano, Português, Espanhol e Ingles. Neste caso, os valores utilizados de *alpha* e *k* foram os pré-definidos pelo programa, ou seja, 0.001 e 3, respetivamente. Devido ao caso de uso desta estratégia, ficheiros com tamanho superior a 2MB, o ficheiro de alvo utilizado foi o maior.

Na figura seguinte, encontram-se os resultados da execução do programa, onde foi detetada o Inglês como língua predominante do caracter 0 ao 79 999, o Italiano desde o caracter 80 000 até ao 129 999 e por último Português da posição 130 000 até à posição 230 000.

```
2021-12-28 11:30:16,678 - Starting to train FCM with files inside ../datasets/languages_train/
2021-12-28 11:30:17,022 - Starting to train FCM with English with order 3
2021-12-28 11:30:24,155 - Starting to train FCM with Italian with order 3
2021-12-28 11:30:27,972 - Starting to train FCM with Portuguese with order 3
2021-12-28 11:30:34,297 - Starting to train FCM with Spanish with order 3
Position (0, 0), language: []
Position (1, 80008), language: ['English']
Position (80009, 130013), language: ['Italian']
Position (130014, 225422), language: ['Portuguese']
```

Figure 9: Resultados obtidos do teste efetuado ao Módulo LocateLang com a estratégia com base em ”blocos”

Noutra perspetiva de visualização dos resultados obtidos, o gráfico abaixo contém o número de bits médios necessários para a compressão do texto alvo ao longo deste, para cada uma das línguas utilizadas como referência.

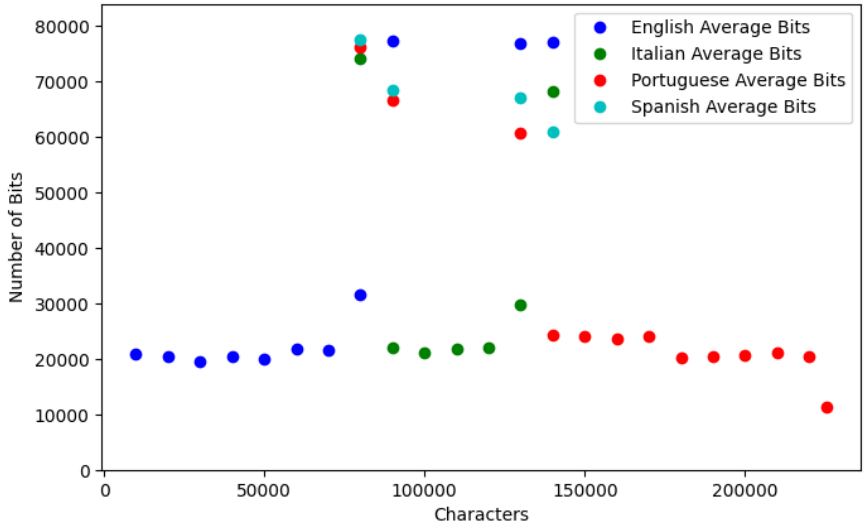

Figure 10: Número médio de bits necessários para a compressão em função da posição com a estratégia com base em ”blocos”

Pelo gráfico consegue-se entender que num primeiro segmento de texto o inglês foi a língua adivinhada devido ao menor número de bits necessários para a compressão, contudo, a partir da posição 79 999 o número de bits necessários aumentou bastante, mais de 10% do valor anterior, pelo que houve uma mudança de língua no segmento de texto em causa, e foi necessário recalcular o número de bits para todas as línguas e identificou-se Italiano como sendo a predominante. No último segmento, ocorreu a mesma situação, tendo sido o Português a língua adivinhada.

Finalmente, consegue-se ter uma visão geral do resultado obtido através da próxima figura, notando-se a existência de blocos com diferentes línguas.

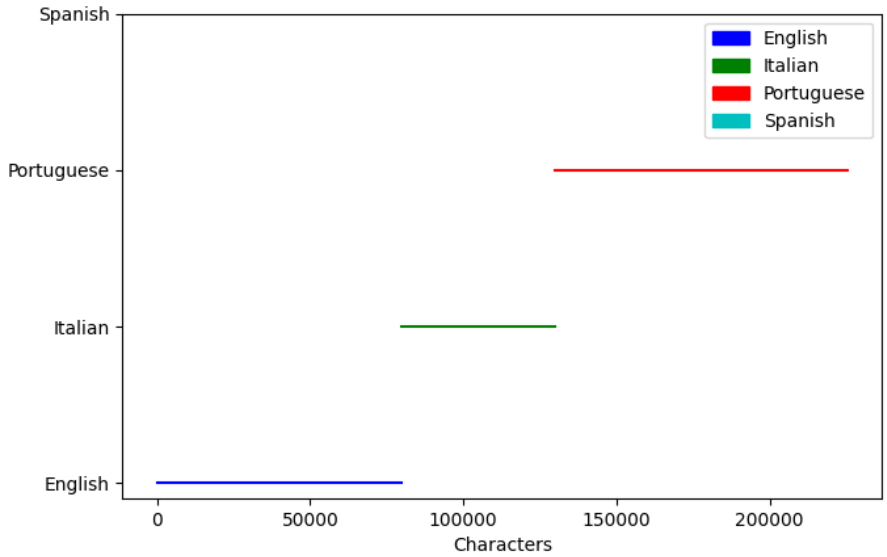

Figure 11: Língua ”adivinhada” em função da posição no ficheiro com a estratégia com base em ”blocos”

4.3.2 Estratégia com base em ”Janelas”

Já para a estratégia com base em ”janelas”, utilizou-se primeiramente o mesmo conjunto de línguas como referência, e como alvo o ficheiro de menor tamanho contendo o misto das línguas Inglês, Italiano e Português devido à razão explicada na subsecção anterior.

Na figura abaixo, encontram-se os resultados da execução do programa utilizando como valores de *alpha* e *k*, 3 e 0.001, respetivamente.

```
2021-12-28 11:36:59,527 - Starting to train FCM with files inside ../datasets/languages_train/
2021-12-28 11:37:00,005 - Starting to train FCM with English with order 3
2021-12-28 11:37:06,885 - Starting to train FCM with Italian with order 3
2021-12-28 11:37:10,368 - Starting to train FCM with Portuguese with order 3
2021-12-28 11:37:16,480 - Starting to train FCM with Spanish with order 3
2021-12-28 11:37:21,846 - Starting Locating Langs for each window
Position (0, 76), language: ['English']
Position (77, 77), language: ['English', 'Portuguese']
Position (78, 78), language: ['English']
Position (79, 81), language: ['English', 'Portuguese', 'Spanish']
Position (82, 84), language: ['English', 'Spanish']
Position (85, 86), language: ['Spanish']
Position (87, 97), language: []
Position (98, 128), language: ['Italian']
Position (129, 130), language: []
Position (131, 242), language: ['Italian']
Position (243, 255), language: []
Position (256, 322), language: ['Portuguese']
Position (323, 354), language: ['Portuguese', 'Spanish']
```

Figure 12: Resultados obtidos do teste efetuado ao Módulo LocateLang com a estratégia com base em ”janelas”

De forma diferente à estratégia com base em blocos, pode-se notar pelos resultados obtidos que conseguimos

não só identificar todas as posições em que houve mudança de língua (e não apenas entre cada bloco) mas também todas as línguas identificadas em cada segmento.

Na figura abaixo, encontra-se a representação do número médio de bits necessários para compressão de cada uma das línguas utilizadas como referência ao longo do ficheiro de texto alvo. Para facilitar a leitura e compreensão do gráfico foram acrescentadas linhas representativas dos *thresholds* de cada língua.

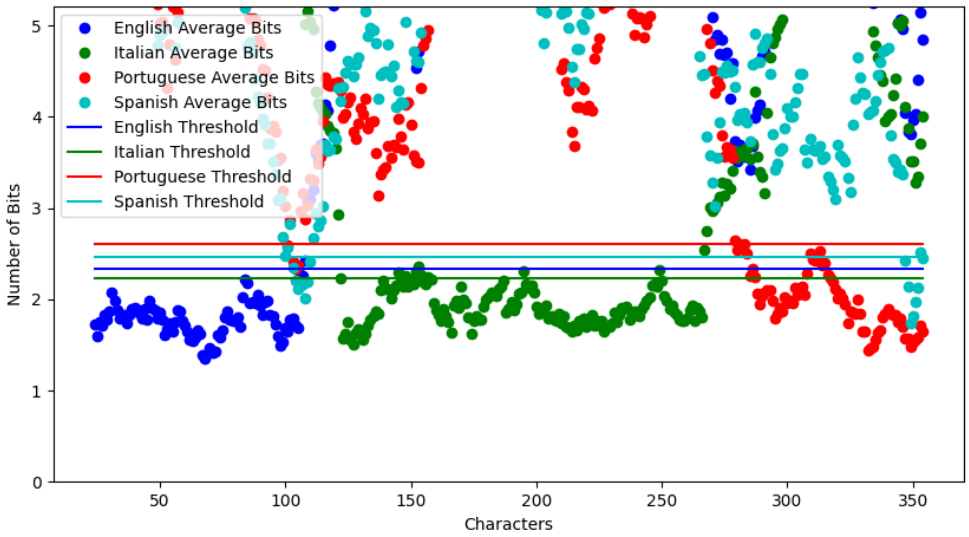

Figure 13: Número médio de bits necessários para a compressão em função da posição com a estratégia com base em ”janelas”

Pela observação do gráfico, pode-se percecionar que as línguas identificadas em cada segmento serão aquelas cujos valores médios de bits necessários encontram-se abaixo dos seus *thresholds*.

Além disto, também que existem alguns segmentos de texto sem língua identificadas e também algumas partes com várias línguas reconhecidas em cada segmento.

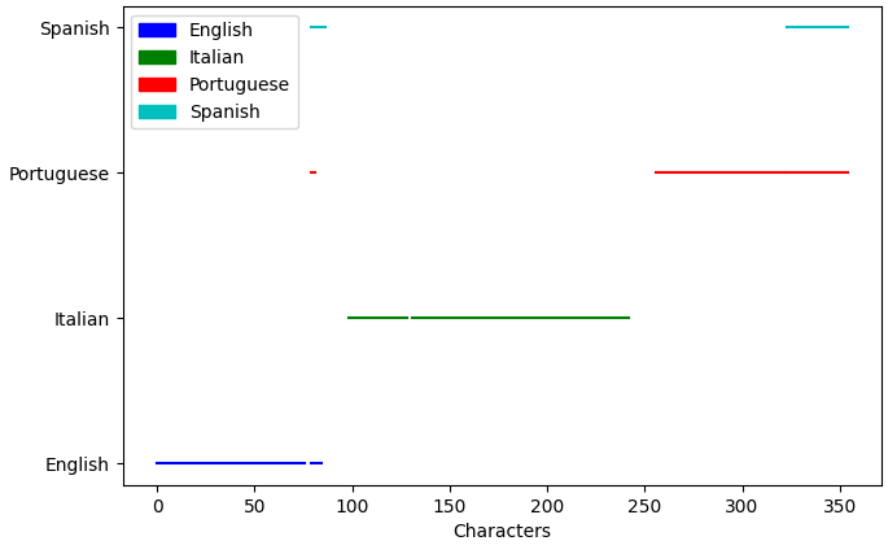

Figure 14: Língua ”adivinhada” em função da posição no ficheiro com a estratégia com base em ”janelas”

4.3.3 Estratégia com base em ”Janelas” utilizando múltiplos valores de *k*

Nesta estratégia com vários valores de *k*, e com o *alpha* igual ao teste anterior (0.001), observa-se uma pequena melhoria nos resultados obtidos em comparação à utilização de um só *k*.

```
2021-12-28 11:42:56,534 - Starting to train FCM with files inside ../datasets/languages_train/
2021-12-28 11:42:56,962 - Starting to train FCM with English with order 2
2021-12-28 11:43:02,378 - Starting to train FCM with Italian with order 2
2021-12-28 11:43:05,279 - Starting to train FCM with Portuguese with order 2
2021-12-28 11:43:10,332 - Starting to train FCM with Spanish with order 2
2021-12-28 11:43:15,319 - Starting to train FCM with English with order 3
2021-12-28 11:43:22,790 - Starting to train FCM with Italian with order 3
2021-12-28 11:43:26,276 - Starting to train FCM with Portuguese with order 3
2021-12-28 11:43:32,299 - Starting to train FCM with Spanish with order 3
2021-12-28 11:43:37,675 - Starting Locating Langs for each window
Position (0, 79), language: ['English']
Position (80, 83), language: ['English', 'Spanish']
Position (84, 86), language: ['Spanish']
Position (87, 97), language: []
Position (98, 128), language: ['Italian']
Position (129, 130), language: []
Position (131, 242), language: ['Italian']
Position (243, 260), language: []
Position (261, 323), language: ['Portuguese']
Position (324, 354), language: ['Portuguese', 'Spanish']
```

Figure 15: Resultados obtidos do teste efetuado ao Módulo LocateLang em ”janela” e com o uso de múltiplos valores de *k*

Comparando com o resultado de um só *k*, é possível notar-se que o Espanhol foi menos vezes reconhecido, e o Português deixou de ser identificado num lugar que não deveria estar.

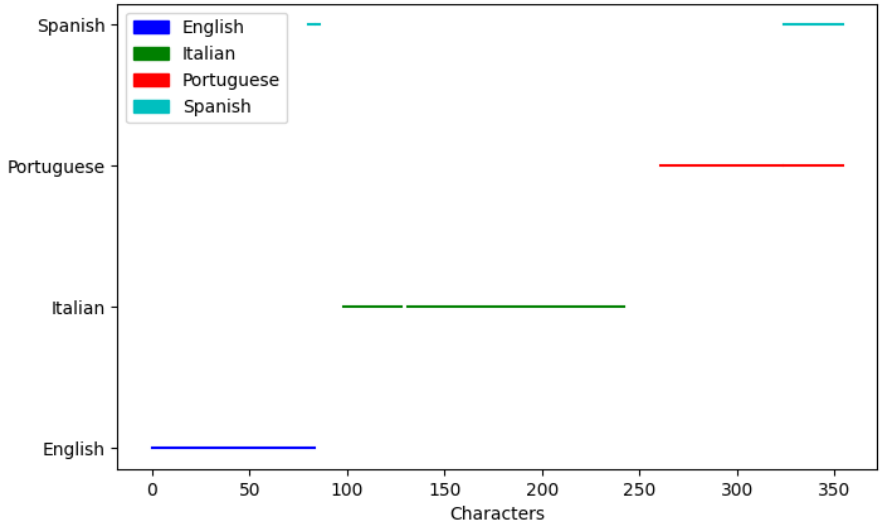

Figure 16: Língua ”adivinhada” em função da posição no ficheiro em ”janela” e com o uso de múltiplos valores de *k*

No entanto, estes resultados obtidos neste teste, não são suficientes para mostrar o impacto que o uso de múltiplos *k* podem ter. Como o programa está escrito em *Python*, não possui a eficiência desejada para testar com ficheiros maiores e com mais línguas de referência para vários valores para *k*. Assim, acreditamos que a utilização de múltiplos *k*s possa ser benéfica, mas como para os nossos casos os resultados variam muito dado o ”tamanho” dos nossos testes, não conseguimos apurar com certeza este acontecimento.

4.3.3.1 Estratégias para os valores dos *thresholds*

Os modelos obtidos estão fortemente dependentes dos valores estipulados para a decisão se o texto será reconhecido como sendo de uma certa língua, ou seja, pelos *thresholds*.

Influência do tamanho dos alfabetos

Na realização de testes com diferentes línguas como referência, sucedeu-nos que para certas línguas, como, por exemplo, Japonês, foram identificadas em todos os segmentos do texto alvo. Rapidamente, pensámos que poderia estar relacionado com o tamanho do alfabeto da língua, visto que se trata de uma língua com um grande número de símbolos e outros caracteres.

A partir do gráfico abaixo pode-se observar que o *threshold* da língua Japonesa está bastante acima de qualquer um dos outros *thresholds*, isto porque apenas se considera a entropia para o *threshold*. Visto que o tamanho do alfabeto do usado no *FCM* da língua Japonesa é de 1479 caracteres, enquanto o de Inglês, Português, Italiano e Espanhol variam apenas entre 80 a 151, a diferença é enorme, e a entropia é muito maior para a língua Japonesa.

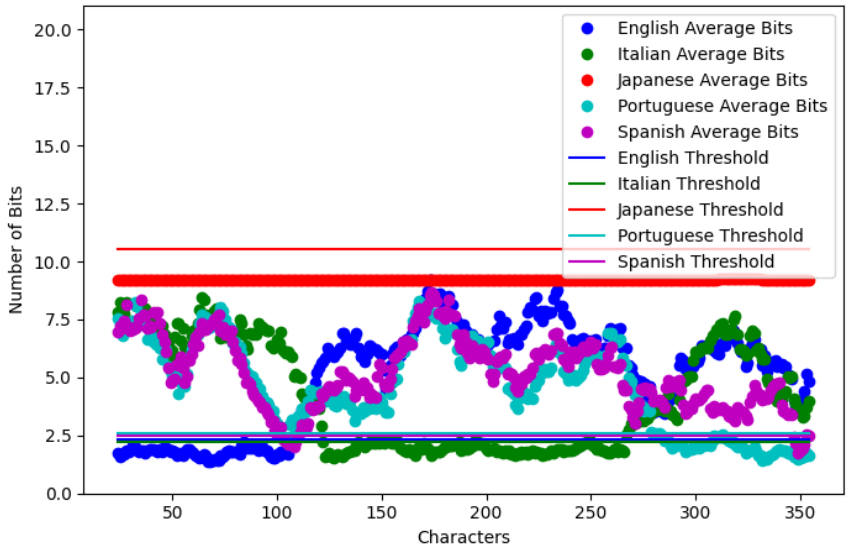

Figure 17: Número médio de bits necessários para a compressão em função da posição utilizando Japonês como uma das línguas de referência

Por esta razão obtém-se a língua Japonesa reconhecida em todo o texto alvo, tal como é possível observar na figura abaixo.

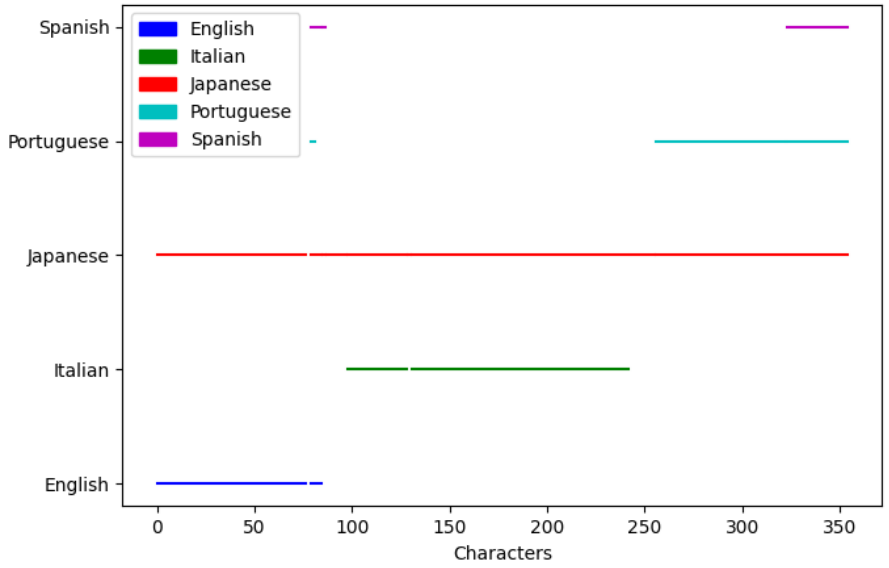

Figure 18: Língua ”adivinhada” em função da posição no ficheiro utilizando Japonês como uma das línguas de referência

Por esta razão decidimos ter em conta não só a entropia, mas também o tamanho do alfabeto da língua correspondente.

```
if not self.threshold_alphabet:
        thresholds[lang_name] = sum_entropies / len(self.multi_k)
else:
        thresholds[lang_name] = (sum_entropies / len(self.multi_k) +
        ↪  log2(self.langs[k][_id].fcm.alphabet_size) / 2) / 2
```

Passando o argumento *-ta* no programa principal, é possível usar a média do valor do *threshold* anterior (média das entropias com o/s valore/s de *k*) com o logaritmo de base 2 do tamanho do alfabeto a dividir por dois.

Assim o valor do *threshold* do Japonês será menor e a média dos bits necessários à compressão já são superiores para o caso de teste.

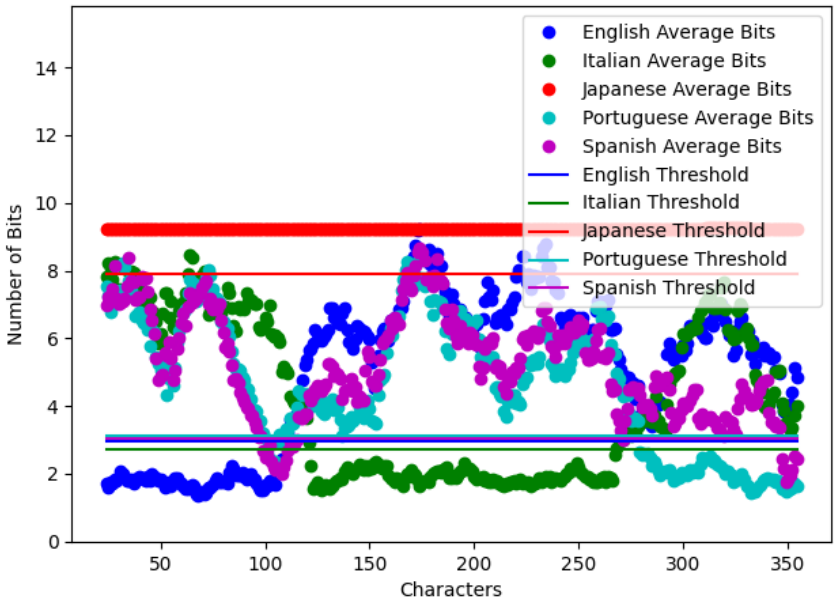

Figure 19: Número médio de bits necessários para a compressão em função da posição, tendo em conta o tamanho do alfabeto na definição do *threshold*

Obtendo-se assim um resultado melhor e com suporte a uma maior variedade de línguas como referência por se ter em conta o tamanho do alfabeto da língua em questão.

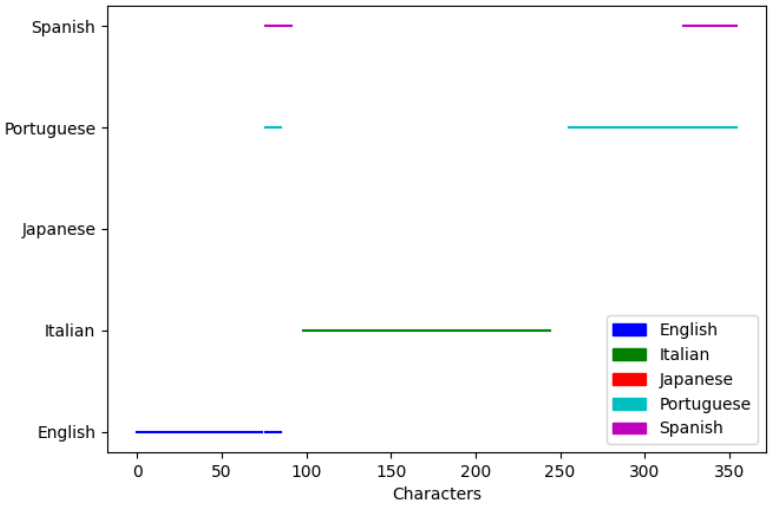

Figure 20: Língua ”adivinhada” em função da posição no ficheiro, tendo em conta o tamanho do alfabeto na definição do *threshold*

Método *compare_lang*

Na figura seguinte são representados os resultados de execução da estratégia com base em ”janelas” recorrendo ao método *compare_lang*. Tal como já foi referido, neste método o *threshold* é definido como 40% do valor médio de bits para a compressão de cada ”janela” de todos os idiomas de referência.

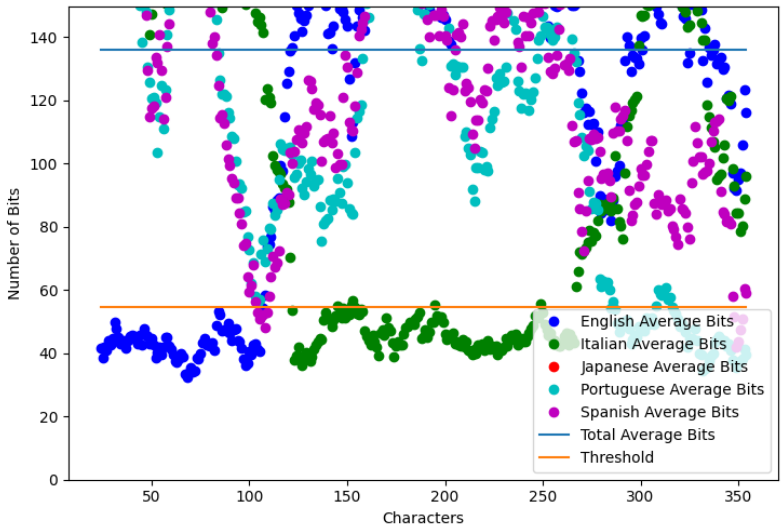

Figure 21: Número médio de bits necessários para a compressão em função da posição em ”janela” e com o uso do método *compare_lang*

Mais uma vez, semelhantemente às estratégias referidas anteriormente, a figura seguinte representa as línguas identificadas nos diversos segmentos do ficheiro de texto alvo, utilizando o método *compare_lang*.

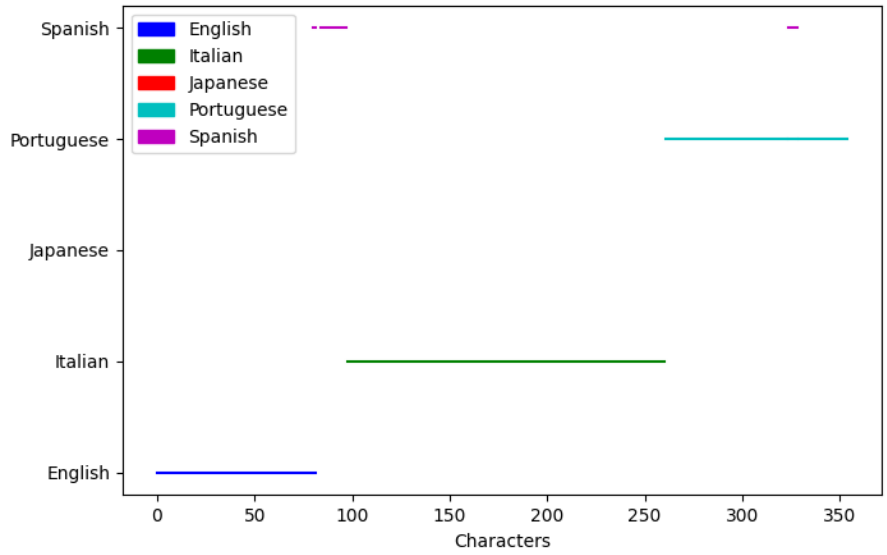

Figure 22: Língua ”adivinhada” em função da posição no ficheiro em ”janela” e com o uso do método *compare_lang*

Como se pode observar pelos gráficos, mesmo sem tendo em conta o tamanho do alfabeto das línguas de referência na definição dos valores dos *thresholds* obtêm-se bons resultados.

5 Caso de uso - Artistas de músicas

Um caso de uso adicional que decidimos testar foi a identificação de artistas/bandas musicais com base em letras das suas músicas. Naturalmente, existem artistas com grande variedade de géneros e temas musicais, e também artistas com músicas com estilos semelhantes, pelo que haverá sempre um erro associado a estes fatores no reconhecimento do autor da música em causa.

Para efetuar este caso de uso, cada ficheiro de texto de referência contém as letras de diversas músicas escritas por um artista, e os ficheiros alvo contêm uma música que não está presente nos outros ficheiros de um desses artistas.

Na figura seguinte, encontram-se os resultados da execução de um teste ao módulo *FindLang*.

```
2021-12-28 20:25:56,124 - Starting to train FCM with files inside ../datasets/music_train/
2021-12-28 20:25:56,129 - Starting to train FCM with beatles with order 3
2021-12-28 20:25:56,403 - Starting to train FCM with bob-marley with order 3
2021-12-28 20:25:56,833 - Starting to train FCM with eminem with order 3
2021-12-28 20:25:57,754 - Starting to train FCM with michael-jackson with order 3
2021-12-28 20:25:58,470 - Starting to train FCM with prince with order 3

For beatles file:
        - Guessed language: prince
-Accuracy: 0.0

For bob-marley file:
        - Guessed language: bob-marley
-Accuracy: 0.5

For eminem file:
        - Guessed language: eminem
-Accuracy: 0.6666666666666666

For michael-jackson file:
        - Guessed language: michael-jackson
-Accuracy: 0.75

For prince file:
        - Guessed language: prince
-Accuracy: 0.8
```

Figure 23: Teste efetuado ao Módulo *FindLang* com um conjunto alargado de ficheiros de letras de músicas como referência e alvo

Pela observação da figura, entende-se que no caso da banda *Beatles* o artista identificado foi *Prince*. Contudo, nos restantes casos e, para nosso espanto, os resultados foram corretos.

Para o teste ao módulo *LocateLang*, utilizou-se a estratégia com base em ”janelas” e teve-se em conta o tamanho do alfabeto para a definição do valor do *threshold*.

Na figura abaixo, encontra-se representado as línguas predominantes ao longo do ficheiro de texto alvo, neste caso, uma música do artista *Eminem*.

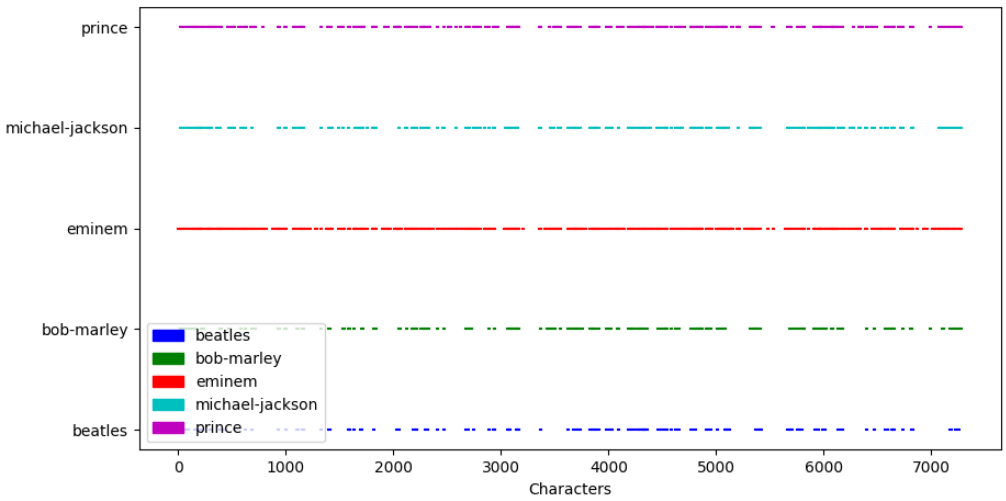

Figure 24: Artista ”adivinhado” em função da posição no ficheiro de música do *Eminem* utilizando ficheiros de letras de músicas como referência

É possível notar que os resultados apesar de não serem tão específicos para o artista em análise, existe uma predominância ao longo da música para esse artista. Estes resultados seriam bastante melhores se os artistas tivessem músicas em diferentes linguagens, mas como foram usados apenas músicas inglesas, implica uma maior dificuldade de distinção entre cada uma.

6 Conclusão

Com este trabalho, pensamos que concluímos com sucesso todos os objetivos propostos, com a criação dos módulos *Lang*, *FindLang* e *LocateLang*, e consequentemente a identificação de línguas em ficheiros de texto.

Através dos testes efetuados, ficámos bastante satisfeitos com o reconhecimento das diversas línguas, e, considerámos que foram os resultados esperados por parte de um sistema com esse objetivo.

Contudo, apesar das otimizações e decisões que tomámos, consideramos que poderá haver alguns melhoramentos de eficiência nos módulos *FindLang* e *LocateLang* pois torna-se um processo demasiado demorado quando utilizamos ficheiros de texto com um tamanho consideravelmente grande, como, por exemplo, o uso de múltiplos processos ou *threads*. Além disso, tal como já foi referido, o facto de os programas serem escritos em Python dificulta este objetivo.