



Identificação de idiomas utilizando um modelo de cópia e um modelo de contexto-finito

Teoria Algorítmica da Informação

**Docentes Armando Pinho e Diogo Pratas
2022/2023**

Mestrado em Engenharia Informática

Artur Romão, 98470
João Reis, 98474
Pedro Sobral, 98491
Tiago Coelho, 98385

Índice

1. Explicação da implementação dos exercícios propostos.....	3
1.1 Programa lang.....	3
1.2 Programa findlang.....	4
1.3 Programa locatelang.....	5
1.4 Modelos de contexto-finito (Programa fcm).....	7
2. Resultados relevantes obtidos.....	9
2.1 Resultados lang e findlang: comparação entre o CPM e o FCM.....	9
2.2 Resultados locatelang.....	12
3. Conclusão.....	17

1. Explicação da implementação dos exercícios propostos

1.1 Programa *lang*

No primeiro exercício, é-nos pedido para desenvolver um programa, *lang.cpp*, que recebe dois ficheiros, um que representa uma certa linguagem, r_i , e o outro, t , que contém o texto que vamos analisar. O objetivo é implementar, neste programa, o modelo de cópia desenvolvido no projeto anterior, de maneira a estimar o número de bits necessários para comprimir t , tendo como base os diferentes modelos r_i .

Para executar este programa, após a compilação, podemos correr o seguinte comando:

```
$ ./lang <ri_file> <target_file> (optional: -a <alpha: int> -t <threshold: float> -k <K: int> -f)
```

Para além dos argumentos obrigatórios, expressos acima, podem também ser passados, opcionalmente, os parâmetros necessários para o modelo de cópia $-a$, *alpha*, $-t$, *threshold*, e $-k$, caso não sejam passados, os valores *default* serão usados, 1, 4 e 4, respetivamente. Quanto à opção $-f$, é usada para ativar o *fcmodel* em vez do *cpm model* que é usado como *default*.

No caso de usar o *cpm model*, na *main* é apenas invocada a função *get_estimated_bits*, que, tal como o nome sugere, estima o número de bits necessários para representar o *target_file* tendo o *ri_file* como referência.

Nesta função *get_estimated_bits*, é aplicado o *cpm* no texto de referência (r_i) e é guardado o modelo, que é um *unordered_map* criado no *cpm* que possui todas as *words* de tamanho k , e para cada uma possui todas as posições em que as mesmas aparecem no texto e o *vector_model* que é um vector com todo o conteúdo do texto de referência. É também guardado o número de diferentes símbolos (caracteres) para uso posterior.

Tendo o modelo é então estimado o valor de bits do ficheiro *target*, usando o modelo do ficheiro de referência adquirido do *cpm*. Este valor é calculado de forma semelhante ao *cpm*, porém aqui o *unordered map* e o vetor do modelo não são atualizados durante o processo. Durante todo o procedimento é guardado num mapa, o índice de cada caracter do ficheiro *target*, o caracter em si e o número de bits necessários para o representar.

Este mapa não tem qualquer utilidade neste programa, mas é criado para uso posterior no programa *locatelang*.

Neste processo todo surgiu um problema/dúvida que referente ao número máximo de bits necessário para representar um símbolo que não constasse no modelo. Inicialmente pensámos em fazer o \log_2 de um número fixo, como o tamanho do alfabeto (26), porém denotamos que não seria correto usar este valor visto que nem todas as línguas possuem o mesmo tamanho de alfabeto. Assim, após alguns testes e comparações optámos por usar o \log_2 do número de diferentes símbolos presentes no texto de referência. Apesar de ter se mostrado superior e constante nos resultados comparado com o tamanho do alfabeto e outros testes, é também necessário notar que o valor de diferentes símbolos também pode variar de um texto de referência mais “rico” para um mais “pobre” em conteúdo.

Assim, após todo o processo de estimativa de bits para o target, é escrito na consola o valor do mesmo.

No caso de querermos usar o modelo fcm através da opção “-f”, o programa irá detectar essa opção e inicialmente começa por treinar o modelo fcm com o ficheiro de referência, e quando completado, aplica-o no ficheiro target. O processo de treino e aplicação do fcm é realizado no *cpm.cpp* e irá ser explicado posteriormente neste relatório todo o processo concretamente.

1.2 Programa *findlang*

Tendo em conta a implementação do exercício 1, o ficheiro *lang.cpp*, é pedido para construir um sistema de reconhecimento de idiomas, que se encontra no ficheiro *findlang.cpp*.

Para executar o programa, basta executar o seguinte comando após a compilação dos ficheiros necessários.

```
$ ./findlang <ri_foldername> <target_file> (optional: -a <alpha: int> -t <threshold: float> -k <K: int> -f)
```

O programa necessita que o utilizador indique a pasta onde se encontram os ficheiros que contêm todos os textos de diferentes linguagens, como pedido no exercício 3, e também o nome do ficheiro alvo, ou seja, o ficheiro que contém a linguagem a ser identificada. Todos os outros parâmetros são opcionais.

Após validar se o ficheiro e a pasta que o utilizador indicou realmente existem, apenas vamos calcular os bits estimados para comprimir o ficheiro alvo aplicando o modelo de cópia usado para comprimir o ficheiro de uma determinada linguagem, tal como explicado no tópico anterior.

Para tal, é chamada a função *get_estimated_bits* do ficheiro *lang.cpp* para realizar este cálculo.

O modelo de cópia de um determinado idioma que comprimir o ficheiro alvo com menor número de bits será a resposta final, isto é, determina a linguagem do ficheiro alvo com a menor quantidade de bits estimada.

O programa consegue identificar com sucesso a linguagem do ficheiro alvo.

1.3 Programa *locatelang*

Quanto ao programa *locatelang*, é capaz de processar texto que possui diversos segmentos de diferentes linguagens e retornar a posição do carácter em que cada segmento começa e a respetiva linguagem identificada.

Para executar o programa, basta executar o seguinte comando após a compilação dos ficheiros necessários.

```
$ ./locatelang <ri_foldername> <target_file> (optional: -a <alpha: int> -t <threshold: float> -k <K: int> -f)
```

O programa necessita que o utilizador indique a pasta onde se encontram os ficheiros que contêm todos os textos de diferentes linguagens, e também o nome do ficheiro alvo, ou seja, o ficheiro que contém a linguagem a ser identificada. Todos os outros parâmetros são opcionais.

Após validar se o ficheiro e a pasta que o utilizador indicou realmente existem, o programa irá percorrer todos os ficheiros da pasta referida e para cada um, no caso de usar o modelo cpm, estima-se o valor de bits para o ficheiro *target* utilizando a função *get_estimated_bits* do programa *lang*. No caso de ser escolhido o modelo fcm através da opção “-f”, irá se treinar o modelo com o texto de referência (função *train_fcm* do *cpm.cpp*) e após isso irá se aplicar o modelo fcm no ficheiro *target* (função *apply_fcm* do *cpm.cpp*).

Estimando o valor de bits, é guardado num mapa os diferentes ficheiros e o resultado que corresponde a um mapa referido anteriormente que é criado na função *get_estimated_bits*, que possui o índice de cada caracter do ficheiro target, o caracter em si e o número de bits necessários para o representar.

Obtendo os diversos resultados, encontramos o primeiro problema, em que estes resultados são bastante inconstantes ao longo do ficheiro, subindo e descendo consideravelmente em cada posição do ficheiro target. Assim é necessário que os mesmo passem por um processo de suavização, para tal percorremos o mapa resultante de cada ficheiro de referência, calculamos a média de três valores de *bits* consecutivos e atribuímos a média de volta ao valor atual de *bits*. Isto é feito para suavizar os valores do mapa e reduzir o impacto dos valores individuais.

Assim, tendo os diversos mapas suavizados, podemos passar para a identificação das linguagens. Para tal, iremos percorrer os diferentes índices do ficheiro target e separar por “palavras”, tendo assim o segundo problema encontrado na realização deste problema, pois uma linguagem não irá variar a meio de uma palavra. Desta maneira também excluimos algumas variações que se encontram no número de bits para os diferentes índices de uma palavra.

Para cada palavra, iremos aos resultados de cada ficheiro de referência e somar os diferentes bits necessários para representar cada símbolo da palavra, no caso deste valor ser menor que o valor de threshold (\log_2 do número de diferentes símbolos do ficheiro de referência), irá ser considerado como linguagem provisoriamente, pois se outro ficheiro de referência também for menor que o threshold e menor que o número de bits da linguagem identificada anteriormente, esta rá ser substituída. Este representa o terceiro problema encontrado, em que no caso de diferentes modelos passarem o threshold, é necessário escolher apenas uma das linguagens representativas, e a nossa opção foi escolher o modelo que consegue representar a palavra em menos bits. Também testamos outros valores de threshold mas nenhum mostrou melhores resultados finais.

Realizando este processo para todo o texto *target* e guardando as diferentes linguagens e respetivos índices onde iniciam num mapa, podemos prosseguir com o processo de identificação. Continuando e tentando reduzir novamente possíveis falsos positivos e variações, o mapa passa por um processo de remoção destes falsos positivos e variações pequenas. Este processo, corresponde ao último problema encontrado, em que havia identificações de linguagens em bocados de texto bastante curtos e para reduzir estas inconstâncias, são removidas todas as identificações

linguagem que durem menos de $K*2$ caracteres, sendo K o número de caracteres de cada palavra usado no modelo.

Desta forma, obtemos o resultado final do mapa com as diferentes linguagens identificadas e os respectivos índices iniciais, e por fim, escrevemos na consola todo o texto do ficheiro target, atribuindo uma cor a cada linguagem encontrada, e o respetivo mapa.

1.4 Modelos de contexto-finito (Programa *fcm*)

Para construir um modelo de contexto-finito, foi construída a classe *FiniteContextModel*, no ficheiro *fcm.cpp*. A classe *FiniteContextModel* é um modelo de contexto finito que tem como objetivo modelar a probabilidade de ocorrência de um próximo símbolo numa sequência, com base nos símbolos anteriores. Ela utiliza uma palavra de tamanho k (order), determinado pela ordem especificada na criação do objeto.

As principais componentes e funcionalidades da classe são:

- Atributos privados:
 - ***int order***: É um inteiro que determina o tamanho da palavra utilizada para calcular as probabilidades.
 - ***unordered_map<string, unordered_map<char, int>> symbolCounts***: É um mapa não ordenado (*unordered_map*) que armazena todos os símbolos que apareceram após uma determinada palavra de tamanho *order*. Armazena também uma contagem de quantas vezes esse símbolo aparece após essa palavra.
 - ***unordered_map<string, int> KWordCounts***: É um mapa não ordenado que regista o número de vezes que cada palavra com tamanho *order* aparece na sequência.
- **Método *train()***: Este método é responsável por treinar o modelo. Basicamente, o que este método irá fazer é atualizar os mapas anteriormente explicados. Primeiro, vai extrair uma palavra de tamanho *order* e vai extrair também o caractere que se segue a essa palavra e vai incrementar os valores nos mapas.

- **Método *get_probability()*:** Este método é utilizado para obter a probabilidade de ocorrência de um próximo símbolo, dado uma sequência e um símbolo subsequente. Ele calcula a probabilidade usando a fórmula do modelo de contexto finito, levando em consideração a contagem do símbolo subsequente e a contagem total da palavra de tamanho order correspondente.

$$P(e|c) \approx \frac{N(e|c) + \alpha}{\sum_{s \in \Sigma} N(s|c) + \alpha|\Sigma|},$$

Figura 1: Fórmula do modelo de contexto finito

- **Método *set_order()*:** Esse método permite alterar a ordem do modelo. Para além disto, as estruturas de contagem (os mapas) são reiniciados.

Portanto, a classe *FiniteContextModel* é utilizada para construir e utilizar um modelo de contexto finito, no qual podemos treinar o modelo com uma sequência de caracteres e obter a probabilidade de ocorrência de um próximo símbolo com base na palavra anterior.

Após a implementação do modelo de contexto-finito questionamo-nos o quão poderia melhor o nosso modelo de cópia, usando esta classe para calcular a probabilidade e assim calcular o número de bit tal como já tínhamos feito no projeto passado.

Para isso, implementamos este modelo de contexto-finito em todos os programas até então explicados, o *lang.cpp*, o *findlang.cpp* e o *locatelang.cpp*. Para utilizá-lo basta colocar a flag “-f” na linha de comandos que este modelo será aplicado para o cálculo da probabilidade, como está indicado na linha abaixo.

```
$ ./findlang <ri_foldername> <target_file> -f
```

Caso a flag “-f” esteja ativada, o programa irá primeiro treinar o modelo utilizando a função *train_fcm()* e, em seguida, irá aplicar esse modelo com a função *apply_fcm()*. Estas funções estão localizadas no fim do ficheiro *cpm.cpp*, e aplicam os métodos da classe *FiniteContextModel*.

Utilizando este modelo, os programas *lang.cpp* e *findlang.cpp* realizam os seus respectivos objetivos com sucesso, porém com resultados diferentes comparando com os resultados obtidos com o nosso modelo de cópia.

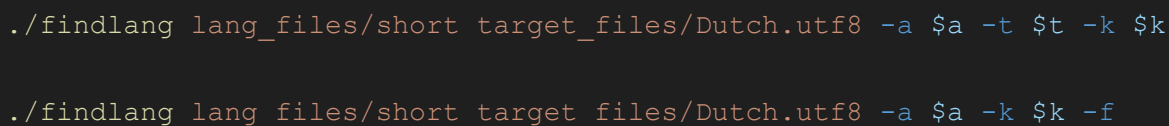
2. Resultados relevantes obtidos

Para obter todos os resultados que iremos apresentar, foram usados textos de diferentes linguagens (20-(French, Swedish, Portuguese, Slovak, English, Hungarian, German, Estonian, Romanian, Dutch, Czech, Turkish, Italian, Slovenian, Spanish, Danish, Luganda, Maltese, Jamaican, Polish)). Todos os textos e excertos usados para teste foram retirados do “Language-Aware String Extractor” (<https://sourceforge.net/projects/la-strings/>).

2.1 Resultados *lang* e *findlang*: comparação entre o CPM e o FCM

Para explorar os resultados destes programas e tentar tirar algumas conclusões, o programa *findlang*, que utiliza o *lang*, foi executado para diferentes valores de alpha, threshold, K e utilizando ou não o finite-context model (fcm).

Os valores de alpha variam entre 0.1 a 1, os do threshold e de K variam de 4 a 10. Para todos estes valores foi executado o *findlang*, com e sem o fcm ativo.



```
./findlang lang_files/short target_files/Dutch.utf8 -a $a -t $t -k $k  
./findlang lang_files/short target_files/Dutch.utf8 -a $a -k $k -f
```

Figura 2: comandos utilizados para executar o programa

Relativamente ao sucesso da identificação do idioma, a taxa de sucesso utilizando o nosso modelo de cópia é de 100%. Independentemente dos parâmetros passados, o programa identifica em todos os casos a linguagem do ficheiro alvo. Enquanto que a taxa de sucesso utilizando o fcm desce para 62,9%. Um facto interessante sobre esta taxa de sucesso é que, para um valor de K igual a 9 ou 10, o modelo falha na identificação, independentemente dos parâmetros passados. Para um valor de K igual a 8,

se o alpha for menor que 0.5, o modelo identifica com sucesso, porém se for maior, falha.

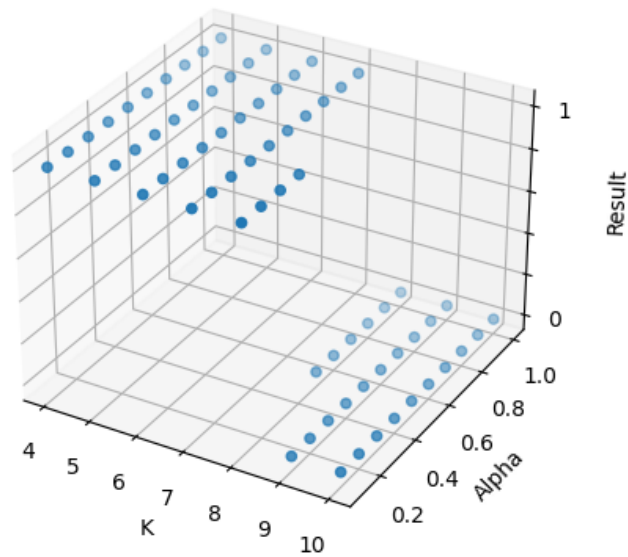


Figura 3: Resultados em relação ao alpha e K escolhidos

Em relação ao tempo de execução, nota-se, de longe, uma grande diferença utilizando o cpm ou o fcm. Ao usar o fcm, o tempo de execução chega a ser cerca de 5 vezes menor do que usando o cpm. Outra observação é que o tempo de execução aumenta à medida que o K aumenta, quando o fcm é aplicado, contudo, quando o cpm é aplicado, o tempo de execução diminui à medida que o K aumenta.

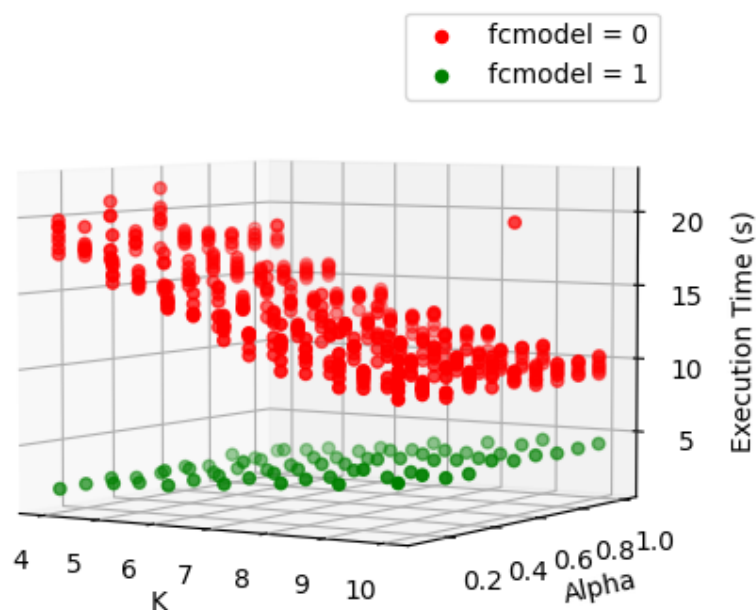


Figura 4: Tempo de execução em relação ao alpha e K escolhidos

Em relação ao número de bits, à medida que o K aumenta, ou seja, à medida que o tamanho da palavra aumenta, maior será o número de bits necessários para codificar.

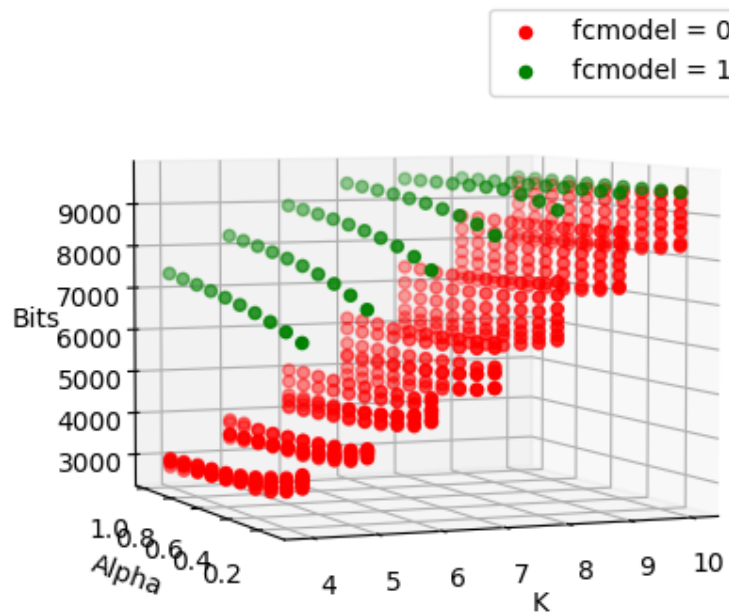


Figura 5: Bits usados em relação ao alpha e K escolhidos

Por exemplo, para um $\alpha=0.5$, um $\text{threshold}=4$ e um $K=8$, o número de bits para codificar o ficheiro, usando o nosso modelo de cópia, “Dutch.utf8” é 7386.58. Para as mesmas condições, mas aplicando o fcm, o número de bits é 9520.85.

Neste exemplo anterior, que acaba por representar a generalidade, podemos verificar que o modelo de cópia comprime melhor do que o fcm, uma vez que apresenta sempre menos bits necessários.

2.2 Resultados *locatelang*: comparação entre o CPM e o FCM

Relativamente ao *locatelang*, foram realizados diversos testes, para valores de K 's diferentes, *threshold*, e para referências pequenas e longas, tanto para o modelo cpm como para o fcm.

Neste relatório iremos apresentar resultados para alguns dos valores testados, que evidenciam as nossas conclusões. Os resultados possuem as diversas linguagens identificadas, o respectivo índice onde iniciam e o

conteúdo do ficheiro target impresso na consola com uma cor para cada linguagem identificada.

Como ficheiro target foi usado um ficheiro que usa excertos de ficheiros de teste, possuindo 4 linguagens diferentes (*Dutch, Portuguese, Maltese e English*), respetivamente por essa ordem.

```
Index: 0 Language: Dutch.utf8
Index: 484 Language: Portuguese.utf8
Index: 860 Language: Maltese.utf8
Index: 1046 Language: English.utf8
Een spiegelneuron of spiegelcel is een neuron dat niet alleen vuurt
als een dier een handeling uitvoert, maar ook als het dier een
handeling ziet uitvoeren door een ander dier (vooral van dezelfde
diersoort). Het neuron weerspiegelt dus als het ware het gedrag van
een ander dier en is op dezelfde manier actief als wanneer het dier de
handeling zelf uitvoert. Dergelijke neuronen zijn gevonden bij
primaten, waaronder de mens, en bij sommige vogels. Sua origem remonta a mais de 200
milhões de anos atrás, quando sua
população se disseminava pelo Nordeste brasileiro. Conífera dióica,
perenifólia, heliófita, pode atingir alturas de 50m, com um diâmetro
de tronco à altura do peito de 2,5m. Sua forma é única na paisagem
brasileira, parecendo uma taça ou umbela. Fuq il-pulizija sawtu bil-lembubi n-nies li
kienu jiehdu sehem f'dimostrazzjoni g
had-drittijiet civili; dan
l-incident hu meqjus bhala l-bidu tal-inkwiet li llum jissejjah The
Troubles. Writing in Simple English means that
simple words are used. It does not mean readers want basic
information. Articles do not have to be short to be simple; expand
articles, add details, but use basic vocabulary.
```

Figura 6: Resultados Modelo CPM para referências curtas com K=10, T=10 e alpha=1

```
Model FCM for file Polish.utf8 done.
target_file_size: 1181
Index: 0 Language: Czech.utf8
Index: 208 Language: French.utf8
Index: 859 Language: Maltese.utf8
Een spiegelneuron of spiegelcel is een neuron dat niet alleen vuurt
als een dier een handeling uitvoert, maar ook als het dier een
handeling ziet uitvoeren door een ander dier (vooral van dezelfde
diersoort).
Het neuron weerspiegelt dus als het ware het gedrag van
een ander dier en is op dezelfde manier actief als wanneer het dier de
handeling zelf uitvoert. Dergelijke neuronen zijn gevonden bij
primaten, waaronder de mens, en bij sommige vogels. Sua origem remonta a mais de 200 milhões de anos atrás, quando sua
população se disseminava pelo Nordeste brasileiro. Conífera dióica,
perenifólia, heliófita, pode atingir alturas de 50m, com um diâmetro
de tronco à altura do peito de 2,5m. Sua forma é única na paisagem
brasileira, parecendo uma taça ou umbela. Fuq il-pulizija sawtu bil-lembubi n-nies li
kienu jiehdu sehem f'dimostrazzjoni g
had-drittijiet civili; dan
l-incident hu meqjus bhala l-bidu tal-inkwiet li llum jissejjah The
Troubles. Writing in Simple English means that
simple words are used. It does not mean readers want basic
information. Articles do not have to be short to be simple; expand
articles, add details, but use basic vocabulary. tiagocoelho@tiagocoelho-Nitro-AN515-54:~/Desktop/MEI/TAI/tai-assignmen
```

Figura 7: Resultados Modelo FCM para referências curtas com K=10 e alpha=1

```

Index: 0 Language: Luganda.utf8
Index: 62 Language: Dutch.utf8
Index: 254 Language: Dutch.utf8
Index: 489 Language: French.utf8
Index: 493 Language: Portuguese.utf8
Index: 654 Language: Portuguese.utf8
Index: 821 Language: Maltese.utf8
Index: 1028 Language: English.utf8
Een spiegelneuron of spiegelcel is een neuron dat niet alleen vuurt
als een dier een handeling uitvoert, maar ook als het dier een
handeling ziet uitvoeren door een ander dier (vooral van dezelfde
diersoort). Het neuron weerspiegelt dus als het ware het
gedrag van
een ander dier en is op dezelfde manier actief als wanneer het dier de
handeling zelf uitvoert. Dergelijke neuronen zijn gevonden bij
primaten, waaronder de mens, en bij sommige vogels. Sua origem remonta a mais de 200 milh
ões
de anos atrás, quando sua
população se disseminava pelo Nordeste brasileiro. Conífera dióica,
perenifólia, heliófita, pode atingir alturas de 50m, com um
diâmetro
de tronco à altura do peito de 2,5m. Sua forma é única na paisagem
brasileira, parecendo uma taça ou umbela. Fuq il-pulizija sawtu bil-lembubi n-nies li
kienu jiehdu sehem f'dimostrazzjoni ghad-drittijiet civili; dan
l-incident hu meqjus bhala l-bidu tal-inkwiet li llum jissejjah The
Troubles. Writing in Simple English means that
simple words are used. It does not mean readers want basic
information. Articles do not have to be short to be simple; expand
articles, add details, but use basic vocabulary.
tiagocoelho@tiagocoelho-Nitro-AN515-54:~/Desktop/MEI/TAI/tai-assignment1/assignment2/src$

```

Figura 8: Resultados Modelo CPM para referências longas com K=10, T=10 e alpha=1

```

Index: 0 Language: Bulgarian.utf8
Index: 208 Language: French.utf8
Index: 492 Language: Portuguese.utf8
Index: 842 Language: Maltese.utf8
Index: 1104 Language: English.utf8
Een spiegelneuron of spiegelcel is een neuron dat niet alleen vuurt
als een dier een handeling uitvoert, maar ook als het dier een
handeling ziet uitvoeren door een ander dier (vooral van dezelfde
diersoort). Het neuron weerspiegelt dus als het ware het gedrag van
een ander dier en is op dezelfde manier actief als wanneer het dier de
handeling zelf uitvoert. Dergelijke neuronen zijn gevonden bij
primaten, waaronder de mens, en bij sommige vogels. Sua origem remonta a mais de 200 milhões
de anos atrás, quando sua
população se disseminava pelo Nordeste brasileiro. Conífera dióica,
perenifólia, heliófita, pode atingir alturas de 50m, com um diâmetro
de tronco à altura do peito de 2,5m. Sua forma é única na paisagem
brasileira, parecendo uma taça ou umbela. Fuq il-pulizija sawtu bil-lembubi n-nies li
kienu jiehdu sehem f
'dimostrazzjoni ghad-drittijiet civili; dan
l-incident hu meqjus bhala l-bidu tal-inkwiet li llum jissejjah The
Troubles. Writing in Simple English means that
simple words are used. It does not mean readers want basic
information. Articles do not have to be short to be simple; expand
articles, add details, but use basic vocabulary.
tiagocoelho@tiagocoelho-Nitro-AN515-54:~/Desktop/MEI/TAI/tai-assignment1/assignment2/src$

```

Figura 9: Resultados Modelo FCM para referências longas com K=10 e alpha=1

```

target_file_size: 1101
Index: 0 Language: Czech.utf8
Index: 208 Language: French.utf8
Index: 471 Language: Portuguese.utf8
Index: 859 Language: Maltese.utf8
Index: 1116 Language: English.utf8
Een spiegelneuron of spiegelcel is een neuron dat niet alleen vuurt
als een dier een handeling uitvoert, maar ook als het dier een
handeling ziet uitvoeren door een ander dier (vooral van dezelfde
diersoort).
Het neuron weerspiegelt dus als het ware het gedrag van
een ander dier en is op dezelfde manier actief als wanneer het dier de
handeling zelf uitvoert. Dergelijke neuronen zijn gevonden bij
primaten, waaronder de mens, en bij sommige vogels. Sua origem remonta a
mais de 200 milhões de anos atrás, quando sua
população se disseminava pelo Nordeste brasileiro. Conífera dióica,
perenifólia, heliófita, pode atingir alturas de 50m, com um diâmetro
de tronco à altura do peito de 2,5m. Sua forma é única na paisagem
brasileira, parecendo uma taça ou umbela. Fuq il-pulizija sawtu bil-lembubi n-nies li
kienu jiehdu sehem f'dimostrazzjoni g
had-drittijiet civili; dan
l-incident hu meqjus bhala l-bidu tal-inkwiet li llum jissejjah The
Troubles. Writing in Simple English means that
simple words are used. It does not mean readers want basic
information. Articles do not have to be short to be simple; expand
articles, add details, but use basic vocabulary.
tiagocoelho@tiagocoelho-Nitro-AN515-54:~/Desktop/MEI/TAI/tai-assignment1/assignment2/src$

```

Figura 10: Resultados Modelo FCM para referências longas com K=5 e alpha=1

```

Index: 493 Language: Spanish.utf8
Index: 508 Language: Spanish.utf8
Index: 529 Language: Portuguese.utf8
Index: 586 Language: Portuguese.utf8
Index: 657 Language: German.utf8
Index: 687 Language: Czech.utf8
Index: 698 Language: Maltese.utf8
Index: 701 Language: Spanish.utf8
Index: 716 Language: Portuguese.utf8
Index: 766 Language: Maltese.utf8
Index: 821 Language: English.utf8
Index: 831 Language: Maltese.utf8
Index: 912 Language: Maltese.utf8
Index: 980 Language: Italian.utf8
Index: 1006 Language: English.utf8
Index: 1059 Language: Dutch.utf8
Index: 1094 Language: Portuguese.utf8
Index: 1102 Language: English.utf8
Index: 1146 Language: Luganda.utf8
Index: 1159 Language: English.utf8
Een spiegelneuron of spiegelcel is een
neuron dat niet
alleen vuurt
als een dier een handeling uitvoert, maar ook als het dier een
handeling ziet uitvoeren door een ander dier (vooral van
dezelfde
diersoort). Het neuron weerspiegelt dus als
het ware het gedrag
van
een ander dier en is
op dezelfde manier actief als wanneer het dier de
handeling zelf uitvoert. Dergelijke neuronen zijn
gevonden bij
primaten, waaronder de mens, en
bij
sommige vogels. Sua origem remonta a mais de
200
milhões
de anos atrás
, quando sua
populaç
ão se disseminava pelo Nordeste brasileiro. Conífera di
óica,
perenifólia, heliófita, pode atingir alturas de 50m, com um di
âmetro
de tronco à altura do
peito de 2
,5m. Sua forma é
única na paisagem
brasileira, parecendo uma taça
ou umbela. Fuq il-pulizija sawtu bil-lembubi n-nies li
kienu jie
hdu sehem f'dimostrazzjoni ghad-drittijiet civili; dan
l-incident hu meqjus bhala l-bidu tal-inkwiet li llum jissejjah The
Troubles. Writing in Simple English means that
simple words are used. It does not mean readers want

```

Figura 11: Resultados Modelo CPM para referências curtas com K=6, T=10 e alpha=1


```

Index: 0 Language: Bulgarian.utf8
Index: 38 Language: Dutch.utf8
Index: 208 Language: French.utf8
Index: 244 Language: Dutch.utf8
Index: 479 Language: Portuguese.utf8
Index: 713 Language: Hungarian.utf8
Index: 759 Language: Portuguese.utf8
Index: 783 Language: Maltese.utf8
Index: 979 Language: English.utf8
Een spiegelneuron of spiegelcel is een neuron dat niet alleen vuurt
als een dier een handeling uitvoert, maar ook als het dier een
handeling ziet uitvoeren door een ander dier (vooral van dezelfde
diersoort).
Het neuron weerspiegelt dus als het
ware het gedrag van
een ander dier en is op dezelfde manier actief als wanneer het dier de
handeling zelf uitvoert. Dergelijke neuronen zijn gevonden bij
primaten, waaronder de mens, en bij sommige vogels. Sua origem remonta a mais de
200 milhões de anos atrás, quando sua
população se disseminava pelo Nordeste brasileiro. Conífera dióica,
perenifólia, heliófita, pode atingir alturas de 50m, com um diâmetro
de tronco à altura do peito de 2,5m. Sua forma é
única na paisagem
brasileira, parecendo uma
taça ou umbela. Fuq il
-pulizija sawtu bil-lembubi n-nies li
kienu jiehdu sehem f'dimostrazzjoni għad-drittijiet ċivili; dan
l-incident hu meqjus bħala l-bidu tal-inkwiet li llum jissejjah The
Troubles. Writing in Simple English means that
simple words are used. It does not mean readers want basic
information. Articles do not have to be short to be simple; expand
articles, add details, but use basic vocabulary.

```

Figura 12: Resultados Modelo FCM para referências longas com K=4 e alpha=0.3

Com estes resultados podemos tirar diversas conclusões quanto ao modelo cpm e fcm referentes a como variam os resultados quando há alteração do tamanho das referências e das variáveis *threshold* e K.

Comparando a figura 6 e 7, podemos retirar a conclusão de que o modelo CPM mostra-se superior em comparação com FCM quanto à detecção de linguagens quando são dadas referências curtas. Pois, o CPM conseguiu identificar as 4 linguagens representadas no ficheiro, contudo não acertou no índice correto onde iniciam. O FCM identificou apenas 3 linguagens, das quais apenas uma está correta (*Maltese*) e também identificando o Inglês como Maltese.

Comparando as figuras 8 e 9, podemos concluir que o CPM continua a se mostrar superior ao FCM, identificando melhor os diferentes segmentos comparativamente com o FCM. Contudo, analisando as figuras 6 e 7 com a 8 e 9, podemos retirar algumas informações interessantes quanto à importância de referências maiores. Visto que o CPM piorou a sua performance quando utilizou referências mais longas e o FCM mostrou uma

performance mais superior quando utilizou referências mais longas comparativamente com as mais curtas.

Nas figuras 10 e 11 testou-se K's menores com os mesmo parâmetros e o modelo CPM reduziu significativamente a sua performance, mostrando resultados inconsistentes e bastante variáveis. Já no caso do FCM não houve uma alteração significativa de performance, e mostrou-se ser capaz de identificar melhor onde o índice de cada linguagem inicia. Com esta informação e as conclusões obtidas dos resultados do *findlang*, testamos reduzir o alpha para 0.3 (figura 12) que no *findlang* mostrou que o FCM se tornava bastante competente a identificar linguagens. Este resultado pode ser visto na figura 9, onde houve um aumento de performance do FCM e é notável um acerto bastante próximo do índice onde inicia a linguagem.

Outra informação relevante a retirar destes resultados, é que valores de K inferiores a 9-10 e superiores a 13 apresentaram uma queda significativa no desempenho do modelo CPM. Valores inferiores de K podem detectar mais linguagens, mas também tendem a cometer erros na identificação correta das linguagens. Valores superiores de K, combinados com um threshold maior, tornam a detecção de segmentos menores praticamente impossível.

Com base nestas conclusões, podemos observar que o CPM tem vantagens em termos de detecção de linguagens com referências curtas e segmentos maiores, pois necessita de K e *thresholds* elevados enquanto o FCM mostra um desempenho superior com referências maiores e mais capaz com segmentos curtos, visto que a um k baixo aliado com alpha inferior a 0.5, mostra-se bastante capaz de identificar o início da linguagem. Além disso, é importante referir que o FCM apresenta uma melhor velocidade de processamento em comparação com o CPM.

É importante considerar estas conclusões ao escolher o modelo mais adequado para diferentes cenários de detecção de linguagens, levando em conta o tamanho das referências, os parâmetros threshold, K, alpha e a necessidade de desempenho e velocidade.

3. Conclusão

Foram realizados testes utilizando os programas *findlang* e *locatelang* com diferentes configurações. O *findlang* utiliza os modelos CPM e FCM para identificar a linguagem de um arquivo, enquanto o *locatelang* mostra os resultados das detecções de linguagem.

Ao utilizar o *findlang*, observamos que o modelo de cópia (CPM) obteve uma taxa de sucesso de 100% na identificação da linguagem do arquivo alvo, independentemente dos parâmetros utilizados. Por outro lado, o modelo de contexto finito (FCM) teve uma taxa de sucesso de 62,9%. Foi interessante notar que o FCM falhou na identificação quando o valor de K era igual a 9 ou 10, independentemente dos parâmetros. Além disso, para K igual a 8, o sucesso da identificação dependia do valor de α .

Em relação ao tempo de execução, o uso do FCM foi significativamente mais rápido do que o CPM, chegando a ser cerca de 5 vezes mais rápido. O tempo de execução aumentou com o aumento do valor de K quando o FCM foi aplicado, mas diminuiu com o CPM.

Quanto ao número de bits necessários para codificar o arquivo, verificamos que o aumento do valor de K resultou em um maior número de bits necessários. O modelo de cópia (CPM) apresentou uma melhor compressão, exigindo menos bits em comparação com o FCM.

No programa *locatelang*, testamos diferentes valores de K , threshold e referências curtas e longas. Os resultados mostraram que o CPM teve um desempenho superior na detecção de linguagens com referências curtas, enquanto o FCM teve um desempenho melhor com referências longas. Valores menores de K no CPM levaram a resultados inconsistentes, enquanto o FCM foi capaz de identificar melhor o início de cada linguagem.

Apesar de comprimir em mais bits, o modelo de contexto-finito consegue identificar a linguagem muito mais rápido do que o nosso modelo de cópia, porém com alguma taxa de insucesso. Se o objetivo do produto é identificar linguagens independentemente do número de bits comprimidos, recomenda-se o uso do algoritmo FCM. Por outro lado, se o objetivo é a compressão em si, é mais adequado utilizar o nosso modelo de cópia.