Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática
Mestrado em Engenharia Informática
Ano Letivo 2020/2021

Teoria Algorítmica da Informação

Trabalho Prático nº 2

Aveiro, 29 de novembro de 2020

António Ramos, 101193
Luís Laranjeira, 81526
Rafael Sá, 104552

Índice

1. Desenvolvimento do Programa

O objetivo do programa, é determinar a similaridade entre dois textos, *target* e *reference*. Uma das soluções é utilizar *finite-context models* para representar a *reference* e através desse modelo determinar o número de bits necessários para fazer a compressão do *target*. Para o sistema de reconhecimento de linguagens devem ser fornecidas várias *references* correspondentes a várias linguagens, de forma a que o sistema seja mais preciso. Um dos desafios consiste na seleção de boas *references*, tanto em tamanho como em conteúdo.
Um objetivo adicional é o programa fazer o reconhecimento das linguagens do *target* por segmentos.

1.1. Estrutura do Programa

A estrutura do programa consiste num conjunto de modelos, sendo que cada um dos modelos contém uma *hash table* utilizada para guardar a contagem, que representa o número de vezes que cada símbolo aparece em cada contexto, o alfabeto usado no modelo, e a soma total do número de ocorrências.

A *hash table* utiliza uma estrutura chave-valor, na qual a chave é cada contexto e o valor é uma nova *hash table,* na qual é guardada a contagem de cada caracter no respectivo contexto e a soma do número de entradas. Com esta abordagem, o objetivo é aliar as vantagens de usar uma tabela às vantagens de usar uma *hash table*, sendo que a primeira *hash table* representa as linhas da tabela e a segunda *hash table* representa as colunas.

Cada linha representa um modelo de probabilidade que é usado para representar um determinado símbolo de acordo com os últimos símbolos processados. Os contadores são atualizados cada vez que um símbolo é processado.

Estes modelos procuram representar as dependências entre dados e dependem do uso de *Markov models*. O objectivo é criar modelos que apresentem boas representações de cada texto de referência e consequentemente da linguagem que pretendemos representar. Com cada um destes modelos criados e treinados é possível identificar qual o modelo que melhor identifica uma linguagem.

O programa recebe um conjunto de parâmetros no qual a sua execução se deve basear. Entre eles estão a ordem k, o parâmetro de *smoothing* alpha e o ficheiro de texto target, que vai ser classificado. Caso os parâmetros k e alpha tenham valores inválidos, os valores por defeito são k = 3 e alpha = 0. Deve ser ainda indicado o ficheiro que contém os textos de referência, podendo as referências estarem distribuídas por vários ficheiros e nesse caso deve ser indicado o ficheiro que contém a listagem de todos os ficheiros, ou então deve ser indicado que se pretende utilizar as referências do *dataset* definido.

Para além disso, deve ser indicado se se pretende fazer uma classificação considerando o texto todo, ou por segmentos. Caso seja por segmentos, existem dois parâmetros opcionais, nomeadamente o valor de *smoothing* do *low pass filter* e o tamanho mínimo do segmento. Caso estes valores não sejam indicados, ou possuam valores inválidos, os valores por defeito são *smoothing* = 40 e tamanho mínimo do segmento = 5 símbolos.

1.2. Alfabeto dos Modelos

O alfabeto é produzido automaticamente com base nos caracteres existentes tanto no texto de referência para a linguagem, como no texto *target*. Todos os caracteres não duplicados são adicionados à estrutura de dados responsável por armazenar o alfabeto, evitando assim que existam caracteres no *target* cujo modelo da referência desconheça..
Esta união de alfabetos foi a estratégia utilizada, ao invés de, por exemplo, se utilizar todos os símbolos da tabela ASCII.

1.3. Linguagens e Textos de Referência

Os textos usados como referência para treinar os modelos para as várias linguagens foram retirados do *dataset* WiLI-2018[1]. Este *dataset* contém 1000 parágrafos para 235 linguagens distintas, totalizando 235000 parágrafos. Os parágrafos são pequenos excertos de texto retirados de páginas da Wikipédia correspondentes às várias linguagens, sendo o seu conteúdo variado, abrangendo as mais diversas áreas, desde política, história, arte, música, ciência, entre outros. Este conteúdo diversificado e pouco redundante permite criar melhores modelos para cada linguagem, aumentando assim a precisão do programa.

Ao realizar testes preliminares com os dados deste *dataset* na identificação de linguagens por segmentos, foi verificado que os resultados devolvidos eram pouco precisos. Isto deve-se ao facto de existir um número elevado de dialetos e ramificações das várias "linguagens-mãe", e as diferenças linguísticas entre essas linguagens muitas vezes serem mínimas. Concluiu-se que os 1000 parágrafos utilizados para treinar os modelos não eram suficientes para fazer distinções corretas e precisas entre vários dialetos e sub-línguas.

Para solucionar o problema, e valorizando a precisão em detrimento de um número elevado de linguagens, procedeu-se a uma verificação manual de todas as linguagens do *dataset*. O objetivo passou por selecionar apenas as linguagens consideradas mais relevantes, tendo em conta critérios como o número de falantes, linguagens oficiais de países, entre outros. Com esta seleção retirou-se os dialetos e sub-línguas menos comuns e pouco utilizadas, e das 235 linguagens iniciais do *dataset*, restaram 102 linguagens que foram consideradas de maior importância.

O Anexo A apresenta a listagem das 102 linguagens disponíveis no programa.

1.4. Identificação das Linguagens

A identificação das linguagens é feita de duas formas distintas, fazendo a identificação tendo por base o texto completo ou fazendo a identificação por segmentos.

A identificação da linguagem considerando o texto *target* completo baseia-se na escolha da linguagem cujo modelo ofereça o menor valor de bits para comprimir o *target*. O cálculo desse valor consiste na soma do número de bits para comprimir cada símbolo do *target*. O número de bits para comprimir cada símbolo é calculado através da fórmula $-log\,P(e|c)$,

[1] WiLI-2018 - Wikipedia Language Identification database

onde a $P(e|c)$ consiste na probabilidade da ocorrência de um dado símbolo num determinado contexto.

Para além deste tipo de identificação, o programa oferece a possibilidade de fazer a identificação das linguagens por segmentos.

1.4.1 Identificação por Segmentos

Para efetuar a identificação das linguagens por segmentos é necessário uma abordagem diferente. Ao invés de fazer a identificação de segmentos tendo como base frases ou parágrafos, optou-se por fazer a identificação caractere a caractere, o que vai melhorar a precisão dos segmentos e vai permitir identificar excertos de frases numa linguagem distinta, como por exemplo estrangeirismos.

Numa fase inicial é calculado o número de bits para comprimir cada símbolo do *target* para todos os modelos de referência. Estes valores podem ser representados sob a forma de um gráfico com a relação símbolo/bits para comprimir o símbolo. A figura 1.1 apresenta um exemplo com os dados para a compressão dos símbolos obtidos através de um dos modelos.

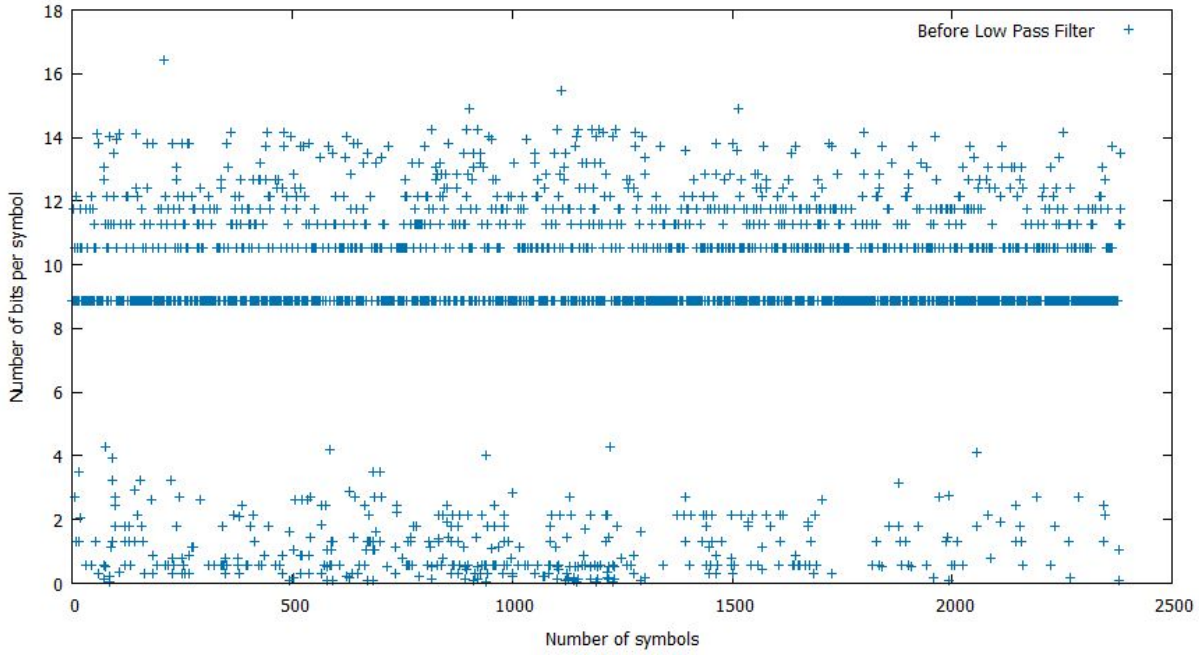


Figura 1.1. Número de bits para comprimir cada símbolo antes do Low Pass Filter

Através da análise à figura 1.1, é possível verificar que não é possível identificar com clareza segmentos que estejam escritos na linguagem do modelo.

Para solucionar este problema e melhorar a precisão da identificação dos segmentos, aplicou-se um *low pass filter*[2] simples aos dados. O *low pass filter* tem como objetivo regularizar o valor dos bits, de forma a criar uma representação mais suave dos dados. Isto será útil para eliminar ruído e possíveis picos que poderiam quebrar uma sequência. O *low pass filter* possui um parâmetro variável que regula o *smoothing* que é aplicado aos dados.

[2] Algoritmo base do Low Pass Filter

Para compreender qual o valor ideal a utilizar para realizar os testes, realizou-se vários testes e fez-se uma análise gráfica. No anexo B é possível consultar todos os valores de *smoothing* e respetivos gráficos analisados.

Feita a análise aos vários valores, concluiu-se que para valores de *smoothing* elevados (valores maiores que 50), a suavização é feita em demasia e é perdida informação, o que consequentemente origina uma pior precisão na detecção dos segmentos. Por outro lado, o valor do *smoothing* também não deve ser muito reduzido (valores menores que 25), pois corre-se o risco de ainda existir ruído e existirem quebras de sequências indesejáveis.

Posto isto, definimos que o valor ideal para o parâmetro de *smoothing* deve ser um entre os valores 25 e 50, mais concretamente o valor de *smoothing* igual a 40. A figura 1.2. apresenta o efeito do low pass filter com o parâmetro de *smoothing* com o valor 40.

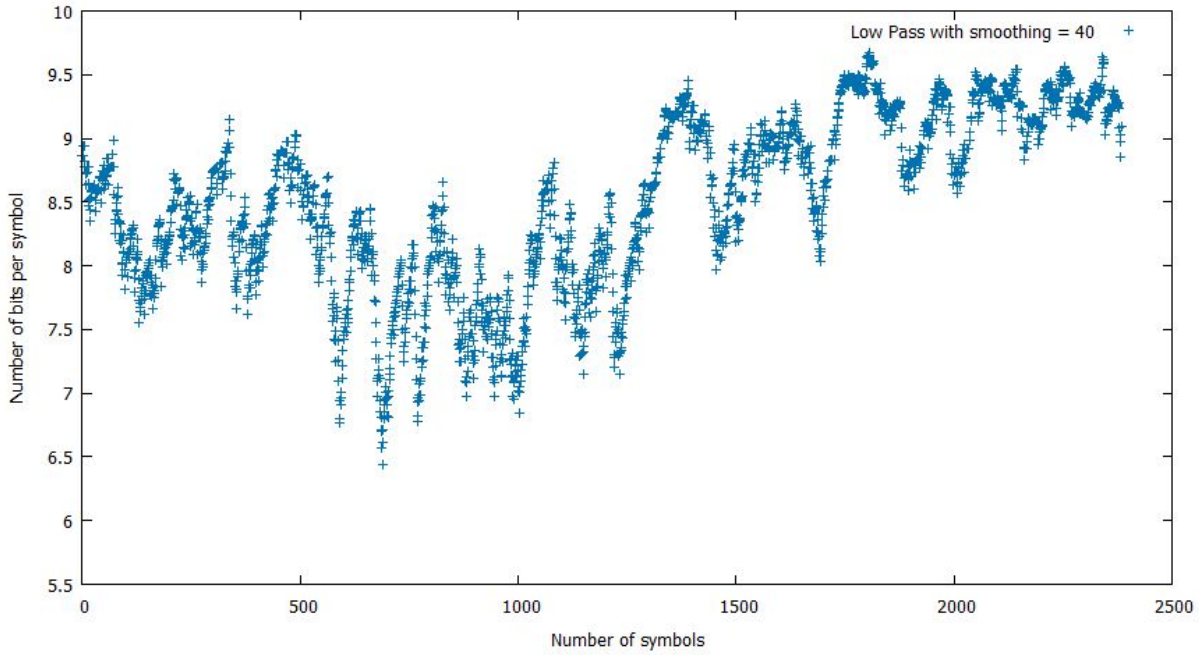


Figura 1.2. Número de bits para comprimir cada símbolo depois do Low Pass Filter com valor de smoothing = 40

Após a aplicação do *low pass filter*, é necessário definir os vários segmentos do texto *target*, e a respetiva linguagem em que está escrito. Um parâmetro adicional foi acrescentado ao programa que define o tamanho mínimo de símbolos que uma sequência deve ter. Isto vai evitar que existam sequências com um tamanho muito reduzido, já que o mais provável é que essas sequências sejam ruído que não foi totalmente eliminado. O valor *default* é de 5 símbolos, mas pode ser alterado através de um dos parâmetros opcionais do programa.

Para definir os vários segmentos, vai-se iterar sobre todos os símbolos do texto *target*, e é verificado qual o modelo que oferece um menor valor de compressão para o símbolo. Caso o modelo escolhido seja o mesmo que foi escolhido nos símbolos anteriores, então significa que a sequência atual mantém-se e avança-se para o símbolo seguinte.

Caso o modelo escolhido seja diferente do escolhido nos símbolos anteriores, então significa que foi detectado um possível segmento noutra linguagem. Antes de se assumir que a sequência anterior foi quebrada, é verificado se a nova sequência terá o tamanho mínimo que é necessário. Caso não tenha o tamanho mínimo, então ignora-se esta possível sequência e a continua-se com a sequência anterior. Caso possua o tamanho mínimo, então dá-se por terminada a sequência anterior, e inicia-se a nova sequência com a linguagem do modelo escolhido.

Após iterar sobre todos os símbolos do texto *target*, o programa apresenta a listagem de todos os segmentos detectados. Para cada segmento, é indicada a linguagem, a posição do caractere onde é iniciado o segmento e a posição do caractere onde termina o segmento. As posições que indicam onde o segmento começa e termina são valores aproximados, podendo não corresponder com exatidão às posições reais no texto.

2. Análise de Resultados

Para verificar se o programa desenvolvido cumpre com os objetivos propostos para o trabalho, foram efetuados vários testes, avaliando o efeito da variação dos parâmetros do programa, bem como o efeito do tamanho das *references* e do *target* na precisão da identificação da linguagem.

Para obter os resultados relativos à precisão da classificação, criou-se ficheiros *target* com texto aleatório para cada uma das 102 linguagens com um mapeamento da linguagem em que o texto está escrito. A cada classificação do programa é feita uma comparação entre a linguagem sugerida e a linguagem real, sendo que o cálculo da precisão consiste no número de vezes em que a classificação foi correta em relação ao total de classificações.

Os valores reais para precisão da classificação podem diferir em cerca de 1% para os valores apresentados.

2.1. Variação dos Parâmetros *k* e *alpha*

Esta análise tem como objetivo compreender de que forma a ordem do modelo e o valor do parâmetro de *smoothing alpha* afetam a precisão do programa.

Os testes foram realizados utilizando o *dataset* completo para as 102 linguagens, testando com um *target* para cada uma das linguagens. A classificação da linguagem é feita tendo em conta o texto *target* completo. A tabela 2.1. apresenta a precisão da classificação obtida com a variação dos dois parâmetros.

	alpha = 0	**alpha = 0.1**	**alpha = 0.5**
k = 1	0.98%	86.27%	85.29%
k = 2	0.0%	83.33%	82.35%
k = 3	0.0%	84.31%	84.31%

Tabela 2.1. Análise da precisão da classificação variando os parâmetros k e alpha

Analisando a tabela, é possível concluir que o parâmetro *alpha* tem um papel fundamental na precisão da classificação. Quando o parâmetro de *smoothing* não é considerado, verifica-se uma taxa de precisão muito baixa, sendo que praticamente nenhum *target* é classificado corretamente. Por outro lado, considerando a existência de um parâmetro de *smoothing*, verifica-se um aumento significativo da precisão, sendo que para *alpha* igual a 0.1 é onde são registados os melhores valores.

Já em relação ao valor do parâmetro k, verifica-se que, de um modo geral, quanto menor o valor, maior o valor da precisão de classificação, apesar de não ser uma diferença muito significativa. Posto isto, verifica-se que, dos três valores testados, os modelos com ordem igual a 1 são os que apresentam uma melhor precisão de classificação.

2.2. Variação do Tamanho das *References* e do *Target*

Esta análise tem como objetivo compreender de que forma o tamanho dos textos de referência, e também dos textos que se pretende classificar, influencia a precisão da classificação. Nestes testes, a classificação da linguagem é feita tendo em conta o texto *target* completo, e foram utilizados os parâmetros k = 2 e *alpha* = 0.1.

Para realizar estes testes, dividimos os textos de referência e *target* como curtos e longos da seguinte forma:

- *Long references*: 1000 parágrafos para cada uma das 102 linguagens;
- *Short references*: 500 parágrafos para cada uma das 102 linguagens;
- *Long targets*: texto com 2500 palavras para cada uma das 102 linguagens;
- *Short targets*: texto com 200 palavras para cada uma das 102 linguagens.

A tabela 2.2. apresenta a precisão da classificação obtida para as combinações possíveis dos tamanhos definidos.

	Long references	***Short references***
Long targets	83.33%	0.0%
Short targets	89.22%	0.98%

Tabela 2.2. Análise da precisão da classificação variando o tamanho dos textos reference e target

Analisando a tabela, é possível concluir que em relação ao tamanho da referência, quanto menor o tamanho, pior é a precisão da classificação. Já em relação ao tamanho dos *targets*, verifica-se o comportamento oposto, sendo que o valor da precisão da classificação melhora para *targets* mais curtos.

2.3. Variação dos Parâmetros para Identificação de Segmentos

Esta análise tem como objetivo compreender melhor o efeito do parâmetro de *smoothing* do *low pass filter* e também o efeito do tamanho mínimo de um segmento. Para realizar esta análise, foi criado um texto *target* com 10 excertos de português, inglês, francês e alemão. Os excertos variam entre as 8 palavras e algumas centenas de palavras, sendo que os excertos mais curtos estão inseridos entre outras linguagens distintas.

O objetivo com estes testes é variar os parâmetros e verificar a precisão na identificação e classificação dos segmentos. Estes testes foram realizados com os parâmetros k = 3 e *alpha* = 0.001, e utilizando o *dataset* completo para as 102 linguagens.

A tabela 2.3. apresenta a precisão da classificação de segmentos identificados consoante os valores para os dois parâmetros. De realçar que as precisões obtidas são apenas relativas ao *target* utilizado para o teste, e os resultados apenas permitem compreender o efeito dos parâmetros e não a precisão do classificador por segmentos.

	Min. Segment Size = 1	**Min. Segment Size = 5**	**Min. Segment Size = 10**
***smoothing* = 10**	35.37%	52.63%	75% *
***smoothing* = 40**	71.43%	100%	100% *
***smoothing* = 70**	75% *	100% *	100% *

Tabela 2.3. Análise da precisão de classificação de segmentos variando os parâmetros

* Precisão relativa à classificação dos segmentos identificados. Alguns segmentos mais curtos não foram identificados.

Analisando a tabela em relação ao valor do tamanho mínimo do segmento, é possível verificar que para valores muito pequenos, neste caso à escala de um símbolo, o programa identifica demasiados segmentos, originando muito ruído e resultando numa fraca precisão de classificação. Por outro lado, se se colocar o tamanho mínimo do segmento com um valor mais elevado, neste caso superior a 10 símbolos, vai eliminar o ruído mas também vai eliminar segmentos que possam ser relevantes, como por exemplo estrangeirismos. Posto isto, verifica-se que o valor ideal é aproximadamente 5 símbolos, já que assim elimina os segmentos pequenos que consistem em ruído, mas não vai eliminar segmentos com um tamanho significativo que possam ser relevantes.

Já em relação ao valor do *smoothing* para o *low pass filter*, é possível verificar que para valores muito pequenos, neste caso *smoothing* igual a 10, a precisão é muito baixa, pois continua a existir muito ruído e os segmentos estão constantemente a ser interrompidos. Por outro lado, para valores muito elevados, neste caso smoothing igual a 70, a precisão obtém valores melhores, mas nunca consegue detectar todos os segmentos existentes no *target*. Isto acontece pois os dados são suavizados em demasia, resultando em perda de informação relevante. Tendo isto em consideração, verifica-se que o valor ideal é aproximadamente 40, pois assim é capaz de eliminar o ruído sem eliminar informação relevante.

Feita esta análise, é possível verificar que os testes realizados corroboram o valor ideal do *smoothing* para o *low pass filter* obtido através da análise gráfica. Para além disso, é possível constatar que deve existir um equilíbrio entre ambos os parâmetros de forma a otimizar a precisão da identificação e classificação dos segmentos.

2.4. Linguagens Identificadas Incorretamente

Analisando de uma forma mais profunda os resultados obtidos, verificou-se que existem algumas linguagens que o programa não é capaz de classificar corretamente de uma forma repetida. Algumas das linguagens são apresentadas de seguida.

Uma das linguagens que o programa não classifica corretamente é o chinês, e as respectivas variantes. O chinês é uma linguagem bastante complexa, já que envolve um

número elevado de dialetos e sub-línguas, e por isso definir referências abrangentes o suficiente torna-se uma tarefa difícil.

Por outro lado, existe um conjunto de linguagens que o programa classifica de forma errada, porém a linguagem obtida e a linguagem expectável são muito semelhantes. Uma das linguagens classificadas erradamente foi o *Norwegian Nynorsk*, sendo que a linguagem obtida foi o *Norwegian Bokmål*. Ambas as linguagens são bastante semelhantes, sendo as representam as duas formas escritas da língua norueguesa.

Concluindo, das linguagens que o programa erra na classificação, cerca de metade são classificações erradas mas cuja linguagens apresentam um grau elevado de semelhança, e a outra metade são linguagens que o sistema não é efetivamente capaz de classificar corretamente.

3. Conclusões

É possível concluir que os parâmetros do programa possuem um grande impacto na classificação da linguagem. Deve existir um equilíbrio entre os diversos parâmetros, de forma a otimizar a precisão da classificação.

Em relação aos textos de referência, é possível concluir que o conteúdo e o tamanho são extremamente importantes para garantir boas classificações. O conteúdo deve ser variado e pouco redundante, e de preferência deve abranger várias áreas, para ser um bom texto representativo de uma determinada linguagem.

Já em relação à identificação e classificação de segmentos, pode-se concluir que o *low pass filter* tem um papel fundamental, já que sem ele existiria demasiado ruído para que fosse possível uma classificação minimamente precisa. Para além disso, o tamanho mínimo de cada segmento também possui um papel importante, e ambos os parâmetros para a segmentação estão intrinsecamente ligados.

Em suma, e com base nos resultados obtidos nos testes através do programa desenvolvido, conclui-se que é possível efetuar classificação de linguagens utilizando compressão de dados.

Anexo A - Linguagens Disponíveis

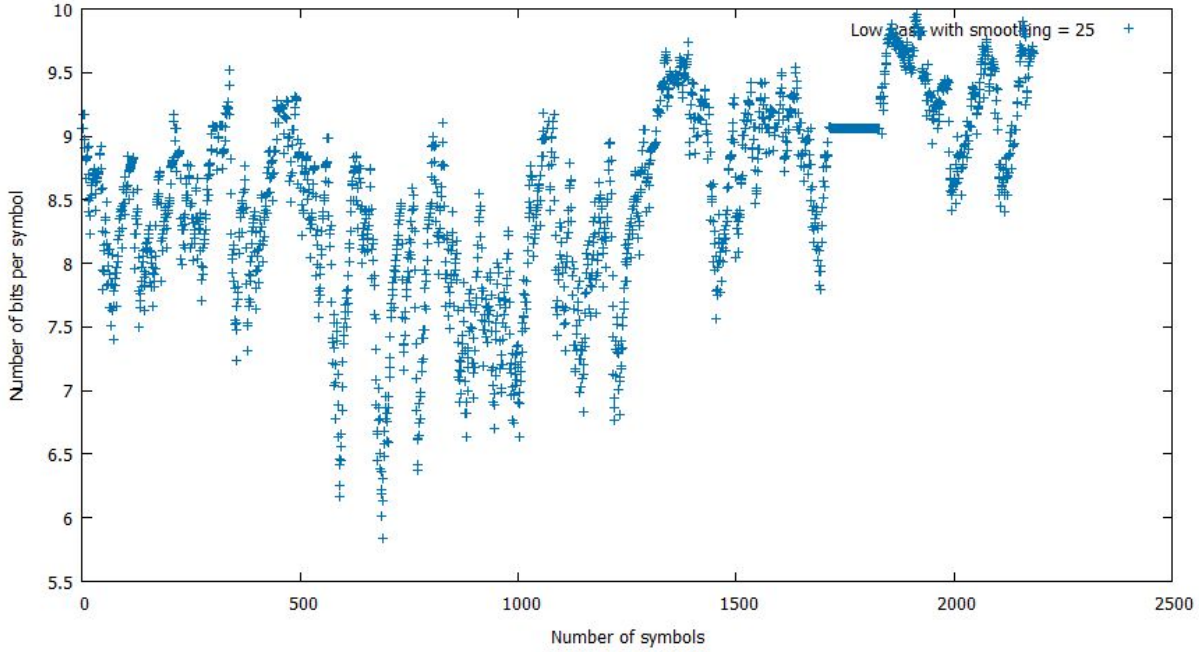
Linguagens Disponíveis no programa:

Standard Chinese, Hungarian, Hebrew, Finnish, Mongolian, Pushto, Turkish, Vietnamese, French, Faroese, German, Jamaican Patois, Persian, Telugu, Min Nan Chinese, Javanese, Irish, Slovak, Burmese, Slovene, Gujarati, Cebuano, Kurdish, Polish, Azerbaijani, Czech, Arabic, Portuguese, Cantonese, Yiddish, Wu Chinese, Tagalog, Malayalam, Uzbek, Marathi, Urdu, Dutch, Egyptian Arabic, Thai, Armenian, Bulgarian, Malay, Assamese, Swahili (macrolanguage), Xhosa, Swedish, Icelandic, Konkani, Macedonian, Estonian, Belarusian, Bengali, Hindi, Korean, Somali, Dhivehi, Danish, Russian, Hakka Chinese, Wolof, Lithuanian, Italian, Nepali (macrolanguage), Haitian Creole, Literary Chinese, Lao, Panjabi, Ukrainian, Hausa, Latin, Latvian, Tamil, Modern Greek, Spanish, Malagasy, Croatian, Norwegian Nynorsk, Central Khmer, Welsh, Tibetan, Serbo-Croatian, Amharic, Quechua, Norwegian Bokmal, Bosnian, Albanian, Tswana, Romanian, Kinyarwanda, Catalan, Sinhala, Kannada, Indonesian, Georgian, English, Kazakh, Oriya, Bhojpuri, Japanese, Afrikaans, Serbian, Luganda.

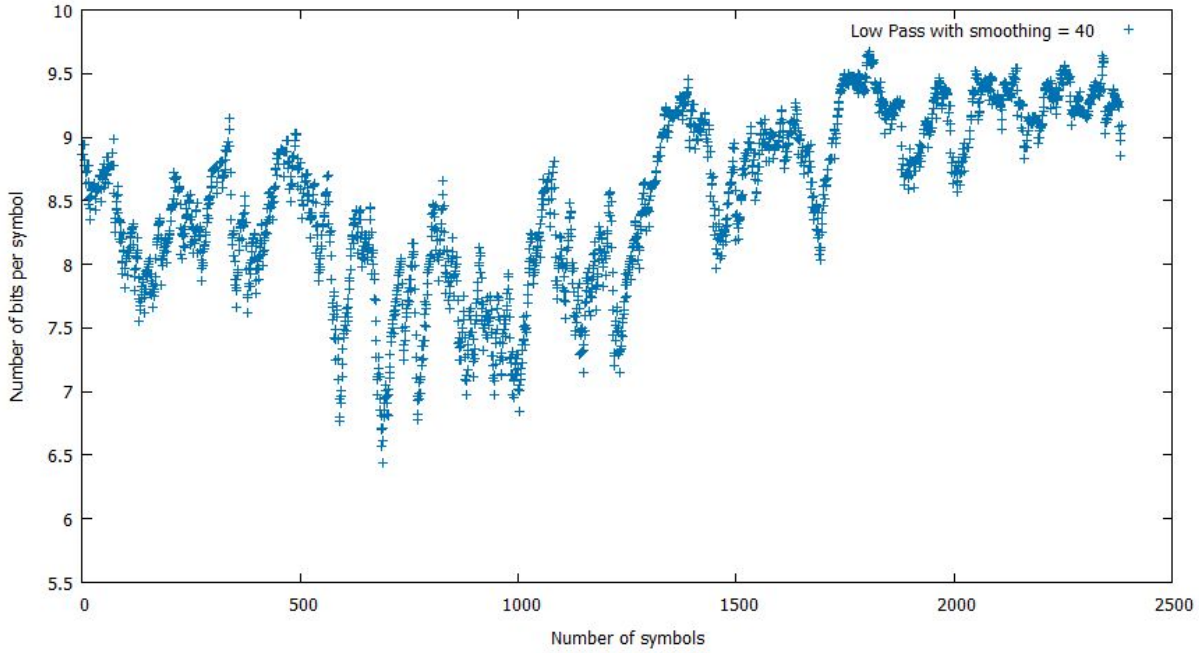
Anexo B - Gráficos Low Pass Filter

Gráficos que demonstram o efeito do Low Pass Filter para vários valores do parâmetro de *smoothing*.

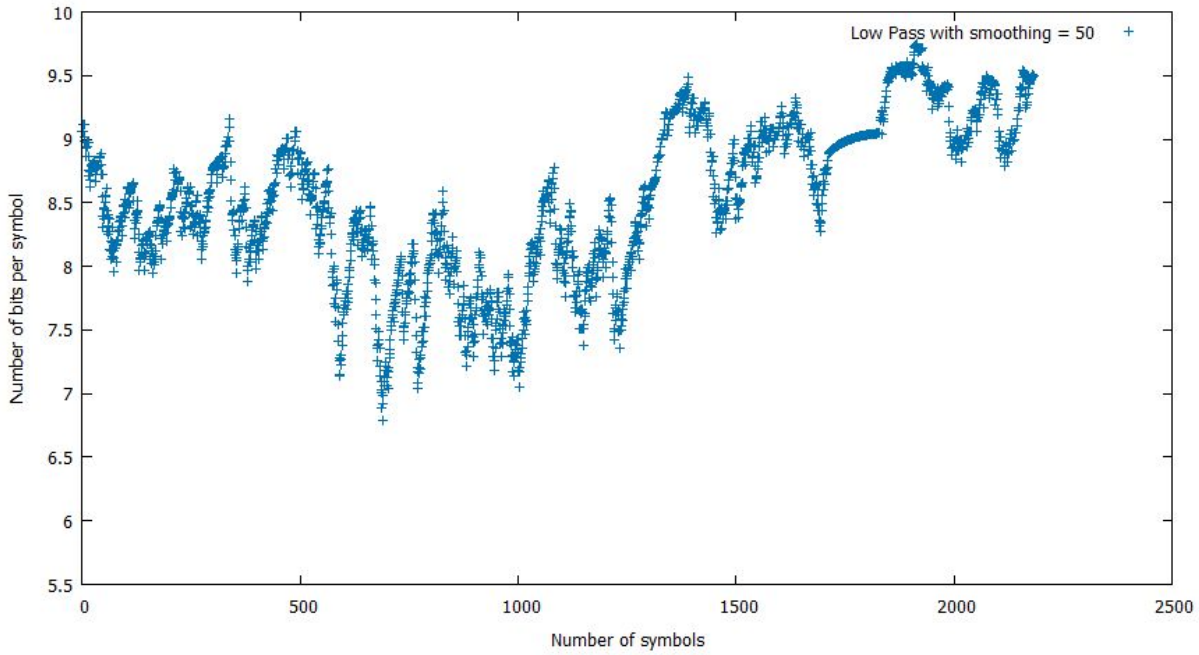
Parâmetro de *smoothing* = 25:



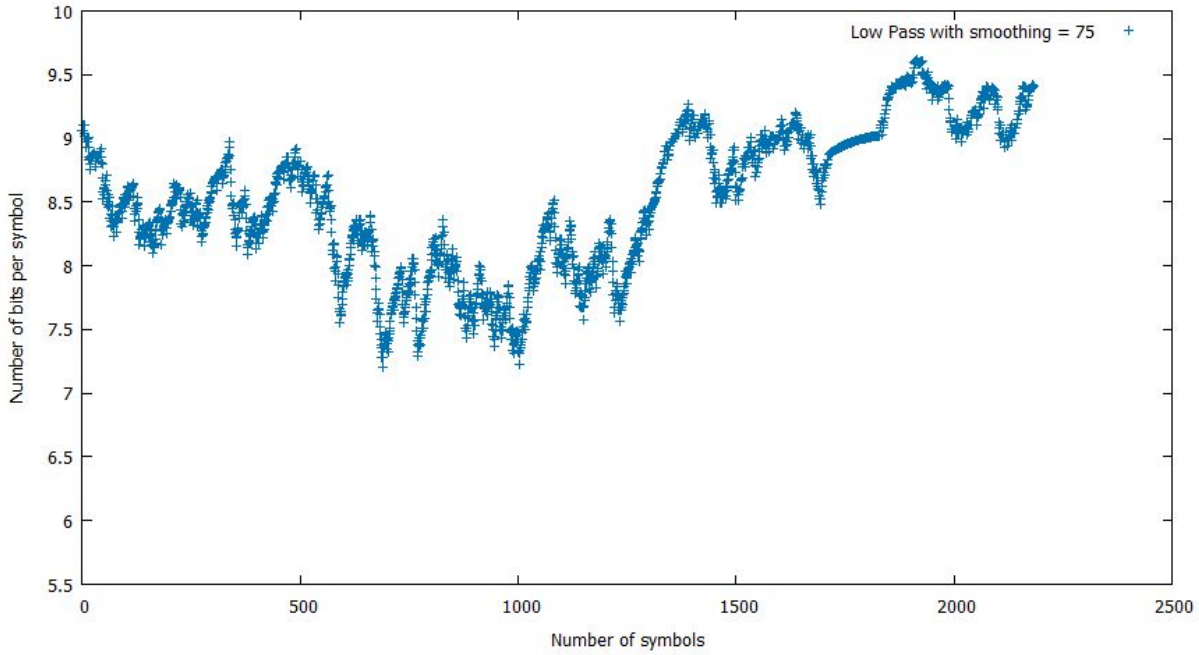
Parâmetro de *smoothing* = 40:



Parâmetro de *smoothing* = 50:



Parâmetro de *smoothing* = 75:



Parâmetro de *smoothing* = 100: