Trabalho 2

Compressão e Codificação de Dados

*MESTRADO EM ENGENHARIA INFORMÁTICA E DE COMPUTADORES*

*Janeiro de 2021*

João Pedro Vitorino

Nº 45232

João Silva

Nº 42086

Índice

[Introdução 3](#_Toc61554754)

[Considerações Iniciais 4](#_Toc61554755)

[Conjunto inicial de ficheiros 4](#_Toc61554756)

[Implementações 4](#_Toc61554757)

[Compressão do conjunto com o dicionário de origem 5](#_Toc61554758)

[Referências 7](#_Toc61554759)

# Introdução

O LZ77, assim como as suas variantes, trata-se de um algoritmo de compressão baseado em técnicas de dicionário, como tal é necessário que exista uma sequência inicial que possa ser usada como base para codificação das seguintes de forma eficiente. No entanto, como é como acontecer em cenários de aplicações modernas baseadas na internet as comunicações entre clientes e servidores, ou servidores componentes destas aplicações, são baseadas na troca de mensagens de dimensão muito reduzida, sendo no enanto o volume total num canal ainda significativo. Este tipo de situação, em particular quando é possível construir um modelo preditivo do conteúdo das mensagens a trocar permite a utilização de um dicionário inicial, pré-definido e distribuído para compressão eficaz.

Um exemplo direto deste modelo e a definição, por parte da equipa do Facebook, de uma API para construção do referido dicionário inicial a partir de um conjunto de mensagens tipo para o modelo de uma aplicação.

No presente trabalho pretende-se estudar estes mecanismos em diferentes algoritmos de compressão baseados no LZ77, nomeadamente o DEFLATE, LZ4 e Zstandart, nas suas implementações para a linguagem Python3.

# 1 - Considerações Iniciais

## Conjunto inicial de ficheiros

De maneira a testar o funcionamento da compressão são necessários ficheiros de teste adequados. Privilegia-se o conceito de compressão de sequências reduzidas com auxílio de um dicionário pré-definido

Com recurso ao gerador de sequências por aproximação de 2ª Ordem apresentado no trabalho anterior, e utilizando o ficheiro *“34767-0.txt”* do conjunto *corpus* fornecido como base foram obtidas sequencias com várias dimensões. Nomeadamente 0.1, 0.2, 0.3, 0.4, 0.5, 1, 2, 3, 4, 5, 6, 10, 20 e 32 KiBytes. Estando a dimensão do dicionário limitado a 32 KiB, de maneira a corresponder a dimensão da janela de buffer usada pelo LZ77, foi criado uma versão foi ficheiro original, “dict\_reduces\_32kiB.txt”, reduzida aos primeiros 32 kiBytes com o comando:

***head --bytes 32768 33767-9.txt >> dict\_reduced\_32kiB.txt***

Todos os ficheiros criados são incluídos em anexo, juntamente com o código correspondente. Nota-se que o código incluído, nesta e outras secções, para alem do acesso a um ambiente da linguagem e uma shell *Bash* requere a instalação das bibliotecas de compressão incluídas nos ficheiros, com a ferramenta *pip3*.

## Implementações

Optou-se por uma implementação baseada na linguagem *Python3,* com auxílio da Shell *Bash* para execução automatizada dos scripts desenvolvidas. Para cada algoritmo foi escolhida uma implementação indicada como de referência para a linguagem, nomeadamente:

* **DEFLATE:** Biblioteca *standart* zlib do Pyhton: <https://docs.python.org/3/library/zlib.html>;
* **LZ4:** python-lz4: <https://github.com/python-lz4/python-lz4>;
* **Zstandart:** python-zstandard: <https://github.com/indygreg/python-zstandard>;

Nos testes realizados foram respeitadas as decisões de design tomadas na definição dos algoritmos, mantendo os parâmetros pré-definidos para os métodos de compressão.

De maneira a similar o comportamento pretendido, uso de dicionários, foi, para as implementações que careçam de um método adequado utiliza a aproximação:

Onde *d* é o ficheiro de dicionário e *f* é o ficheiro comprimido. Conforme indicado ao longo do relatório foi ainda feito recurso a métodos próprios para o efeito. Nomeadamente a passagem de uma sequência de bytes inicial dita *zdict,* em Zlib, indicada por um cabeçalho específico incluído no *stream* de símbolos. Assim como a analise em particular do processo *ZSTD\_createCDict(),* para construção de um dicionário a partir de varias fontes.

# 2 - Compressão do conjunto com o dicionário de origem

Para efeitos de comparação das duas versões de compressão define-se os fatores:

onde ganho é lido como *n* bytes com dicionário por cada byte sem.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dimensão Sequencia Gerada [kiB] | Ganho [bytes/byte] | | | | |
| **DEFLATE(Zlib)** | | | **LZ4**  **(Aproximação)** | **Zstandart**  **(Aproximação)** |
| **Aproximação** | | **zdict** |
| 0,1 | 1,0963 | 1 | | 1,1000 | 1,1046 |
| 0,2 | 1,0583 | 1 | | 1,0531 | 1,0633 |
| 0,3 | 1.0414 | 1 | | 1,0362 | 1,0436 |
| 0,4 | 1.0306 | 1 | | 1,0272 | 1,0307 |
| 0,5 | 1.0235 | 1 | | 1,0221 | 1,0238 |
| 1 | 1,0137 | 1 | | 1,0121 | 1,0142 |
| 2 | 1,0061 | 1 | | 1,0060 | 1,0068 |
| 3 | 1,0042 | 1 | | 1,0041 | 1,0046 |
| 4 | 1,0033 | 1 | | 1,0032 | 1,0037 |
| 5 | 1.0028 | 1 | | 1,0026 | 1,0030 |
| 6 | 1.0024 | 1 | | 1,0022 | 1,0026 |
| 10 | 1,0016 | 1 | | 1,0014 | 1,0018 |
| 20 | 1,0008 | 1 | | 1,0007 | 1,0009 |
| 32 | 1,0006 | 1 | | 1,0005 | 1,0006 |

Tabela 1 - Compressão de sequências usando o ficheiro de origem como dicionário.

Nota-se, ao contrário dos restantes algoritmos, o uso de LZ4 (cuja definição privilegiou a velocidade de operação, por sacrifício da compressão) teve situações onde a dimensão das mensagens mais pequenas piorou ligeiramente. Nas situações assinaladas a vermelho a compressão foi pior para ambos os casos, ao passo que nas assinaladas a laranja foi pior no caso sem dicionário. Uma vez que o LZ4 não possui uma fase de codificação de entropia, recorrendo unicamente a substituição por dicionário e que os ficheiros criados usam apenas aproximação de 2ª Ordem, isto é, a relação entre símbolos (neste caso, bytes) adjacentes, o comportamento é esperado.

O ganho real com a Zlib, com recurso ao parâmetro *zdict* do método de compressão, é nulo para todos os casos. Numa situação em que se recorra a esta implementação para comprimir as mensagens não so não existe benefício em termos de mensagens individuais, como na globalidade das trocas iria-se piorar a situação, resultante da transferência eventual do dicionário.

Para termo de comparação, assumiu-se apenas os valores resultantes dos métodos por aproximação na comparação dos valores obtidos. Pode-se observar um conjunto de comportamentos:

* Tendencialmente as mensagens com maior benefício, pelo uso de dicionário, em termos de compressão são as de dimensão mais reduzida, representando, por exemplo, troca de mensagens entre utilizadores de uma aplicação. Tal deve-se que será nestas situações onde, para o método sem dicionário, existem menos possibilidades de escolha para substituição de subsequências;
* Para ficheiros de dimensão superior o ganho tende a diminuir, refletindo a redução do benefício apresentado pelo dicionário;
* O ganho para as três variantes é muito semelhante, sendo mais significativo para o Zstandart e menos para o LZ4. Juntamente com o aumento de dimensão registado já notado, pode-se concluir que o LZ4 não é adequado para compressão deste tipo de sequências (curtas com relações de 2ª ordem).

# 3 - Compressão baseada em estimativa do dicionário

Recorrendo ao mesmo conjunto de ficheiros anteriormente mencionados, do conjunto inicial, e ao estimador de 2ª Ordem, foi obtido um novo dicionário estimado a partir do dicionário anterior e calculados os ganhos para a sua utilização sobre o conjunto.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dimensão Sequencia Gerada [kiB] | Ganho [bytes/byte] | | | | |
| **DEFLATE(Zlib)** | | | **LZ4**  **(Aproximação)** | **Zstandart**  **(Aproximação)** |
| **Aproximação** | | **zdict** |
| 0,1 | 1,0976 | 1 | | 1,0973 | 1,1047 |
| 0,2 | 1,0593 | 1 | | 1,0534 | 1,0629 |
| 0,3 | 1,0406 | 1 | | 1,0372 | 1,0422 |
| 0,4 | 1,0305 | 1 | | 1,0274 | 1,0323 |
| 0,5 | 1,0244 | 1 | | 1,0221 | 1,0259 |
| 1 | 1,0135 | 1 | | 1,0120 | 1,0144 |
| 2 | 1,0062 | 1 | | 1,0061 | 1,0068 |
| 3 | 1,0042 | 1 | | 1,0041 | 1,0046 |
| 4 | 1,0034 | 1 | | 1,0032 | 1,0038 |
| 5 | 1,0028 | 1 | | 1,0026 | 1,0031 |
| 6 | 1,0024 | 1 | | 1,0022 | 1,0026 |
| 10 | 1,0016 | 1 | | 1,0014 | 1,0018 |
| 20 | 1,0009 | 1 | | 1,0007 | 1,0009 |
| 32 | 1,0006 | 1 | | 1,0004 | 1,0006 |

Tabela 2 - Compressão de sequências usando o dicionário estimado.

Observa-se que as valores obtidos, globalmente apresentam os mesmos padrões que anteriormente. No entanto surge um novo padrão contraditório. O ganho melhorou para o DEFLATE e piorou para o LZ4 e para Zstandart, para dimensões inferiores a 1 kiB. Para todos os casos o ganho piorou, ou, permaneceu igual para as restantes dimensões.

Aparenta-se que este mecanismo não obtém bons dicionários para algoritmos mais fortemente dependentes de substituição de sequências, ao passo que métodos que implementam codificação de entropia com base a distribuição dos símbolos obtém melhores resultados.

No caso do método utilizado em Zstandard, tANS (*table variant Asymetric Numeral System*) nota-se que este se baseia na codificação das relações aritméticas de sequências de símbolos. Neste método utiliza-se as distribuições de probabilidade para decidir um fator de escalonamento dos sucessivos estados

# Referências

|  |  |
| --- | --- |
| [1] | Python Software Foundation, “zlib — Compression compatible with gzip,” [Online]. Available: https://docs.python.org/3/library/zlib.html. [Acedido em janeiro 2021]. |
| [2] | “python-lz4,” [Online]. Available: https://github.com/python-lz4/python-lz4. [Acedido em janeiro 2021]. |
| [3] | G. Szorc, “python-zstandard,” [Online]. Available: https://github.com/indygreg/python-zstandard. [Acedido em janeiro 2021]. |