Improving Run Length Encoding (RLE) on bit level by preprocessing

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

Universität Trier
FB IV - Informatikwissenschaften
Lehrstuhl für Informatik I

Gutachter:	Prof. Dr. Ingo J. Timm
	xxxxxxxxxx
Betreuer:	xxxxxxxxxx

Vorgelegt am xx.xx.xxxx von:

Sven Fiergolla
Am Deimelberg 30
54295 Trier
sven.fiergolla@gmail.com
Matr.-Nr. 1252732

Zusammenfassung

Hier steht eine Kurzzusammenfassung (Abstract) der Arbeit. Stellen Sie kurz und präzise Ziel und Gegenstand der Arbeit, die angewendeten Methoden, sowie die Ergebnisse der Arbeit dar. Halten Sie dabei die ersten Punkten eher kurz und fokussieren Sie die Ergebnisse. Bewerten Sie auch die Ergebnissen und ordnen Sie diese in den Kontext ein.

Die Kurzzusammenfassung sollte maximal 1 Seite lang sein.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Introduction

Die Einleitung besteht aus der Motivation, der Problemstellung, der Zielsetzung und einem erster Überblick über den Aufbau der Arbeit.
TODO:
- explain compression ratio
- define unit of compression to quantify question and results

1.1 Motivation

In the last decades, digital data transfer became available everywhere and to everyone. This rise of digital data urges the need for data compression techniques or improvements on existing ones. Run-length encoding [1] (abbreviated as RLE) is simple coding schemes that performs lossless data compression. RLE compression simply represents the consecutive, identical symbols of a string with a run, usually denoted by $\sigma\ i$, where σ is an alphabet symbol and i is its number of repetitions. To give an example, the string aaaabbaaabbbba can be compressed into RLE format as $a^4b^2a^3b^4a^1$. Its simplicity and efficiency make run-length encoding is still used in several areas like fax transmission, where RLE compression is combined with other techniques into Modified Huffman Coding [2]. Most fax documents are typically simple texts on a white background, RLE compression is particularly suitable for fax and often achieves good compression ratios. Another appliance of RLE is optical character recognition, in which the inputs are usually images of large scales of identically valued pixels. Other applications appear in bioinformatics, where RLE compression is employed to speed up the comparison of two biological sequences.

1.2 Problem statement

Some strings like aaaabbbb archive a very good compression rate because the string only has two different characters and they repeat at least twice. Therefore it can be compressed to a^4b^4 so from 8 byte down to 4 bytes if you encode it properly. On the other hand, if the input is highly mixed characters with few or no repetitions at all like abababab, the run length encoding of the string is $a^1b^1a^1b^1a^1b^1a^1b^1a^1b^1$ which needs at least 16 bytes.

So the inherent problem with run length encoding is obviously the possible explosion in size, due to missing repetitions in the input string. Expanding the string to twice the original size is not really a good compression so one has to make sure the input data is fitted for RLE as compression scheme. One goal is to minimize the increase in size in the worst case scenario.

Also it should improve the compression ratio on data suited for run length encoding and perform better than the originally proposed RLE.

1.3 Main Objective

Was ist das Ziel der Arbeit. Wie soll das Problem gelöst werden?

- bessere kompressionsrate im vergleich zu konventionellem RLE
- gleiche oder ähnliche decoding zeit ?
- ansatz beschreiben

The main objectives that derives from the problem statement is to archive an improved compression ratio compared to regular run length encoding. To unify the measurements, the compression ratio is calculated by encoding all files listed in the Galgary Corpus and then normalize the results.

Since most improvements like permutations on the input, for example a revertable Burros-Wheeler transformation to increase the number of consecutive Symbols or a different way of reading the bytestream take quite some time, encoding speed will increase. A secod objective might be to keep decoding speed close to the original run legth encoding.

1.4 Structure of this work

Was enthalten die weiteren Kapitel? Wie ist die Arbeit aufgebaut? Welche Methodik wird verfolgt?

- describe following structure
- use references
- try to keep idea of a recurrent theme

2. Principles of compression

To understand compression, one first has to understand some basic principles of information theory like Entropy and different approaches to compress different types of data with different encoding and entropy. I will also show the key differences between probability coding and dictionary coding and a few comments on lossy compression.

2.1 Compression and Encoding fundamentals

TBD
...

2.1.1 Information Theory and Entropy

As Shannon described his analysis about the English language [3], he used the term Entropy closely aligned with its Definition in classical physics, where it is defined as the disorder of a system. Specifically speaking, it is assumed that a system has a set of states it can be in and it exists a probability distribution over those states. Shannon then defines the Entropy as:

$$H(S) = \sum_{s \in S} p(s)\, log_2 \frac{1}{p(s)}$$

where S describes all possible States, $p(s)$ is the likelihood of $s \in S$. So generally speaking it means that evenly distributed probabilities imply a higher Entropy and vice versa.

In addition to that, Shannon defined the term self information $i(s)$ as:

$$i(S) = log_2 \frac{1}{p(s)}$$

indicating that the higher the probability of a state, less information can be contained. As an example, the statement „The criminal is smaller than 2 meters.“ is very likely but doesn't contain much information, whereas the statement „The criminal is larger than 2 meters.“ is not very likely but contains lots of information. With these definitions in mind, we can analyze some properties for the English language from an information theory perspective of view.

2.1.2 General Analysis

To evaluate the efficiency of a specific compression technique, we have to determine how much information the raw data contains. In this case for textual compression at first, we are talking about the English language.

- upscale analysis for UTF-8 encoded data not 96 ASCII Chars
- comment reference

2.1.3 Probability Coding

The general idea behind Probability Coding is to analyze probabilities for messages and the encode them in bit strings according to their probability. This way, messages that are more likely and will repeat more often can be encoded in a smaller bit string representation. The generation of those probability distributions is considered part of the Analyzer module by the algorithm and will be discussed later on (chap: design, chap: impl). Probability coding compression can be discerned into fixed unique coding, which will represent each message with a bit string with the amount of bits being an integer, for example Huffman coding as type of prefix coding. In contrast to Huffman coding there are also arithmetic codes which can represent a message with a floating point number of bits in the corresponding bit string. By doing so they can „fuse“ messages together and need less space to represent a set of messages.
...

2.1.4 Dictionary Coding

...

2.1.5 Irreversible Compression

Irreversible Compression or also called „Lossy Compression“ is a type of compression which looses information in the compression process and is therefore not completely reversible, hence the name. There are a lot of use cases for these type of compression algorithms, mostly used for images, video and audio files. These files typically contain information almost not perceptible for an average human like really small color differences between two pixels of an image or very high frequencies in an audio file. By using approximations to store the information and accept the loss of some information while retaining most of it, lossy image compression algorithms can get twice the compression performance compared to lossless algorithms. Due to the fact that most lossy compression techniques are not suited for text compression which is the main use case for this work, we wont elaborate any further on this topic.

2.2 Run Length Encoding

...

2.3 Huffman Coding

...

2.4 State of the Art

Die Literaturrecherche soll so vollständig wie möglich sein und bereits existierende relevante Ansätze (Verwandte Arbeiten / State of the Art / Stand der Technik) beschreiben bzw. kurz vorstellen. Es soll aufgezeigt werden, wo diese Ansätze Defizite aufweisen oder nicht anwendbar sind, z.B. weil sie von anderen Umgebungen oder Voraussetzungen ausgehen.

Je nach Art der Abschlussarbeit kann es auch sinnvoll sein, diesen Abschnitt in die Einleitung zu integrieren oder als eigenes Kapitel aufzuführen.

State of the art compression
- techniques
- use cases

2.5 Limits of compression

- existence of not compressable strings
-https://www.quora.com/Is-there-a-theoretical-limit-to-data-compression-If-so-how-was-it-found
- unable to compress random data
- Kolmogorow-Komplexität

3. Analysis

In diesem Kapitel sollen zunächst das zu lösende Problem sowie die Anforderungen und die Randbedingungen einer Lösung beschrieben werden (eine präzisierte Aufgabenstellung).

The following chapter contains a detailed analysis of the problem and some fundamental requirements for the algorithm.

...

3.1 Prerequisites

- prerequisites ...

3.2 Initial Findings

- no matrix representation but chunks of type byteArray
- static encoding difficulties

...

3.3 Improvements by Preprocessing

- First improvements due to byte remapping
- burrows wheeler transformation

...

3.4 Further Improvements

- combining different compression techniques

...

3.5 Summarization

Am Ende sollten ggf. die wichtigsten Ergebnisse nochmal in *einem* kurzen Absatz zusammengefasst werden.

4. Conceptual Design

In diesem Kapitel erfolgt die ausführliche Beschreibung des eigenen Lösungsansatzes. Dabei sollten Lösungsalternativen diskutiert und Entwurfsentscheidungen dargelegt werden.

To archive the main objective a few ideas to improve run length encoding where found. The first real difference is the conception of reading the data in chunks and then encode it in parallel, meaning all the most significant bits in one chunk, then all second most significant bits and so on. The second change was to switch between the encoding of runs of characters bytes to count runs of ones and zeros so basically the same mechanism but on bit level.

4.1 Parallel Byte Reading

...

4.1.1 First Ideas

...

4.1.2 New Perspective

...

4.1.3 Performance Improvements

...

4.2 Preprocessing

...

4.2.1 Burrows Wheeler Transformation

...

4.2.2 Byte Mapping to reduce Input space

...

4.2.3 Dynamic Encoding

...

4.3 Alternative Compression for partial data

...

4.3.1 Huffman Coding

...

4.4 Summarization

Am Ende sollten ggf. die wichtigsten Ergebnisse nochmal in *einem* kurzen Absatz zusammengefasst werden.

5. Implementation

- some detailed implementation details
...

5.1 Implementation Decisions

- why kotlin
- performance improvements with the graalvm
- other decisions
...

5.2 Implementation Detail

- detailed information about specific modules and classes
...

5.2.1 Parsing

- explain universal parsing

5.2.2 Burrows Wheeler Transformation

- TBD

5.2.3 Byte Remapping

- show use case

5.2.4 Dynamic Encoding

- different sizes and optima for different files and chunk sizes

5.3 Implementation Evaluation

- evaluation of implementation choices made ...

6. Evaluation

Hier erfolgt der Nachweis, dass das in Kapitel 4 entworfene Konzept funktioniert. Leistungsmessungen einer Implementierung werden immer gerne gesehen.

6.1 Functional Evaluation

- Mathematical Comparison
- Comparison of encoded file sizes
- comparing to regular REL and Huffman coding
...

6.2 Benchmarks

- Benchmark with the Galgary corpus
http://www.data-compression.info/Corpora/
...

6.3 Conclusion

Am Ende sollten ggf. die wichtigsten Ergebnisse nochmal in *einem* kurzen Absatz zusammengefasst werden.

7. Discussion

(Keine Untergliederung mehr)

Literaturverzeichnis

[1] *Results of a prototype television bandwidth compression scheme, Proceedings of the IEEE 3*, volume 55, March 1967.

[2] A. Harry Robinson Roy Hunter, editor. *International digital facsimile coding standards, Proceedings of the IEEE 68*, 1980.

[3] C. E. Shannon. *Prediction and entropy of printed English*, volume 30. Jan 1951.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich diese Bachelor-/Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die aus fremden Quellen direkt oder indirekt übernommenen Gedanken als solche kenntlich gemacht habe. Die Arbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vor-gelegt. Sie wurde bisher auch nicht veröffentlicht.

Trier, den xx. Monat 20xx