PDF Compression

Shangxing Jiang

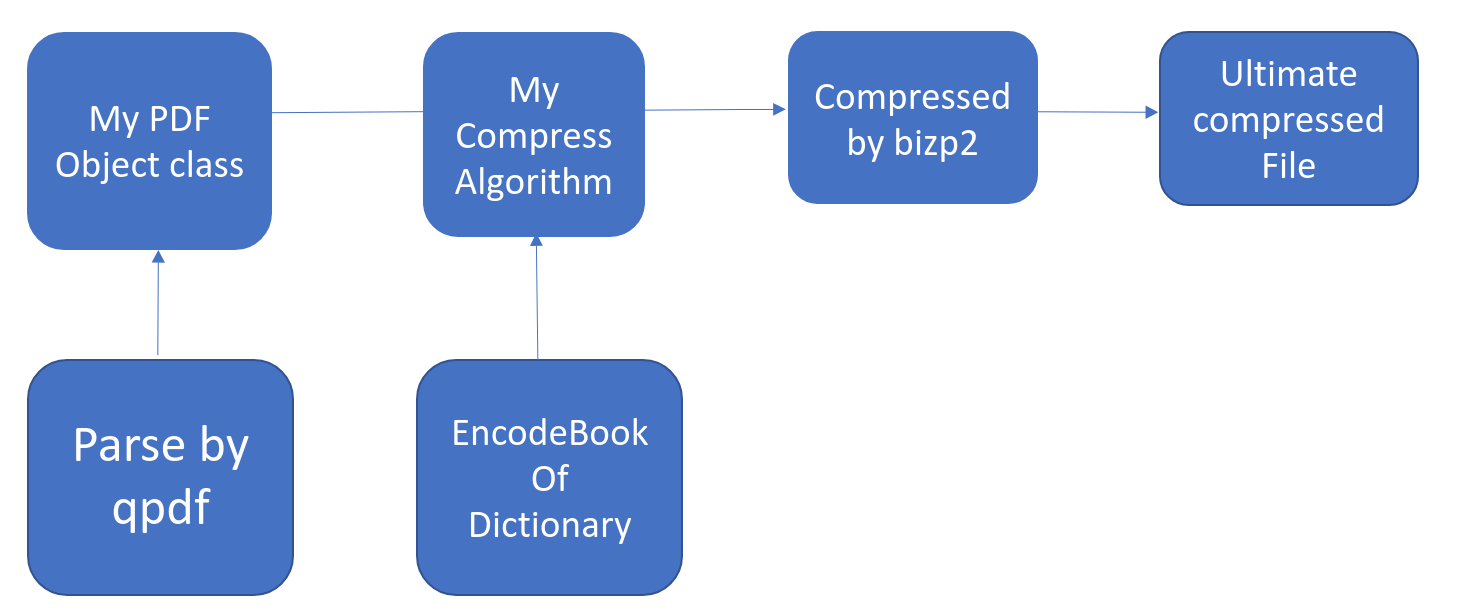
Abstract

PDF (Portable Document Format) is one of the most widely used file formats in today’s life. However, popular compression softwares, such as zip, can only slightly compress PDF file, because the text and pictures of PDF file are already encrypted. In the project, a software is designed to accomplish compressing PDF file to a much smaller size.

Introduction

If we go through the internal structure of PDF, we will find PDF file contains more information than the pure text plus picture pixels, which causes the size are much larger. It brings much inconvenience for devices with limited memory disk and costs longer time for uploading and downloading. Meanwhile, due to the unavailability of directly compressing PDF file, a new way of compression should be considered. In this article, I will only focus on text compression of PDF file.

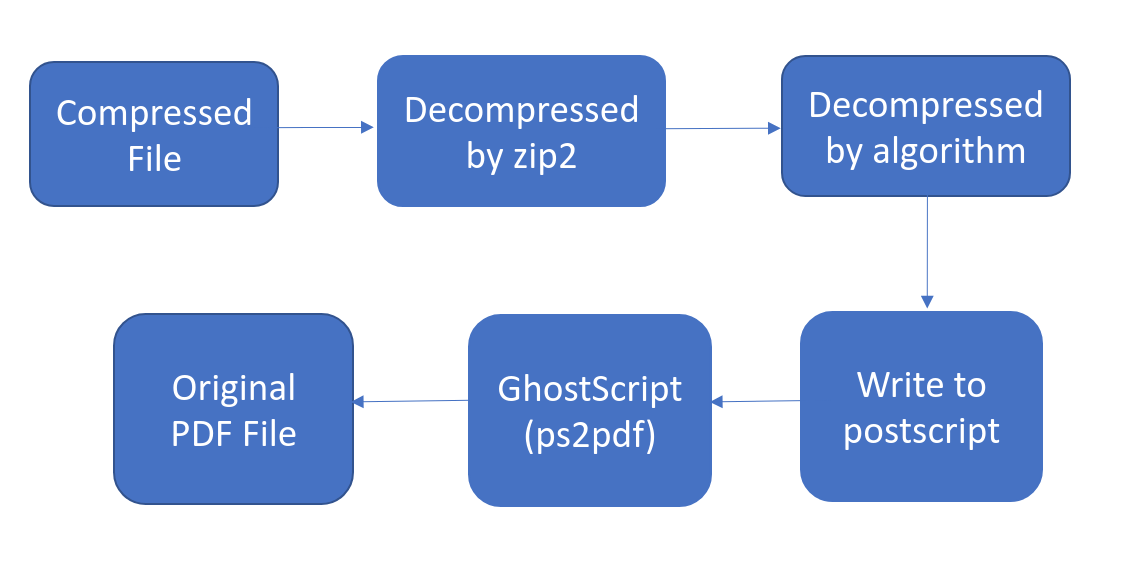
On the approach of compression, a PDF file is firstly parsed by qpdf\*. Then the plain contents are encoded by a compression algorithm. The algorithm is based on an English dictionary and some high-frequent patterns, which can encode every English word or pattern to 3 bytes. For ASCII code, the range of one byte is from 1 to 127 (‘\0’ is not used), which means 3 bytes have 127^3 = 2,048,383 permutations, sufficient for all English words and patterns. After this plain encoder, the encoded file is compressed by bzip2\*\* to get the ultimate compression file. The whole compression program design is shown as figure 1.

Figure 1

\*qpdf: a open source library, which can parse PDF file and give information, such as text contents, coordinates and image pixels.

\*\*bzip2: a open source software, which can compress a plain text file based on many mature compression algorithm.

On the approach of decompression, the ultimate compression file is firstly decompressed by bzip2, giving 3-byte encoded plain contents. Then the encode book is read in, so every code can match its corresponding English word or pattern. Till here, all the contents of PDF file have been got. The remaining work is to recover the PDF file. To do this, the contents should be written as postscript\* format, which could be converted to PDF file by ps2pdf\*\*. I have to mention that the recovered PDF file only has the same appearance with the original one, but the internal structure is not guaranteed to be identical. The whole decompression program design is shown as figure 2.

Figure 2

\*postscript: is a page description language in the electronic publishing, created at Adobe Systems. (cited from <https://en.wikipedia.org/wiki/PostScript>)

\*\*ps2pdf: is a script that comes bundled with ghostscript, a freeware PostScript interpreter. ps2pdf runs a PostScript file through GhostScript and outputs a PDF file.

Method

Compression Procedure:

(1) Parse PDF

Because of the messy format of PDF file and the encrypted streams, it is hard to get contents by myself. So I use qpdf library to implement this. Then these contents are stored in a PDFFile class. The hierarchy is shown as figure 3.

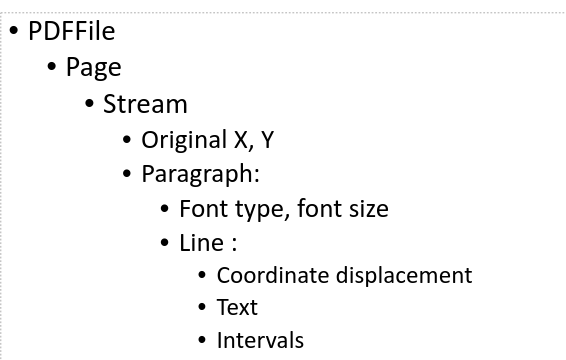
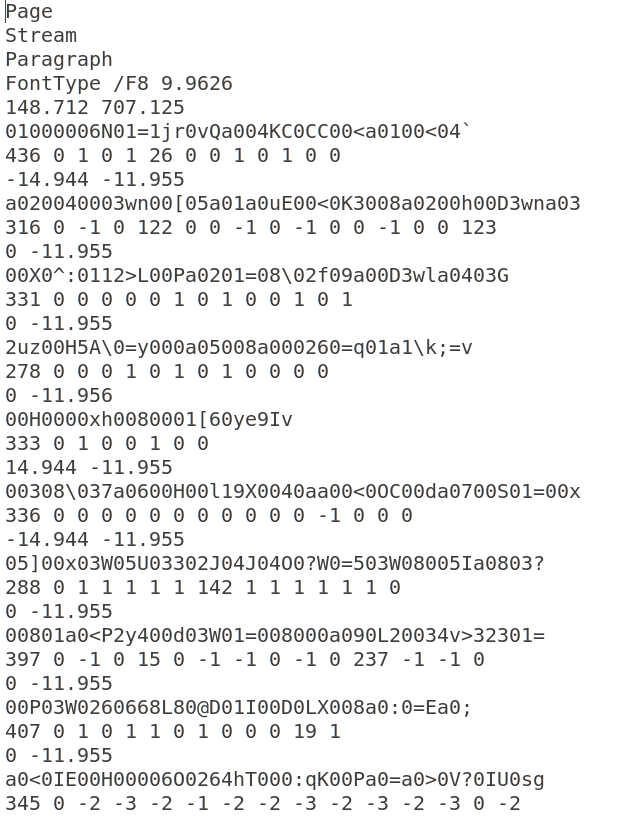
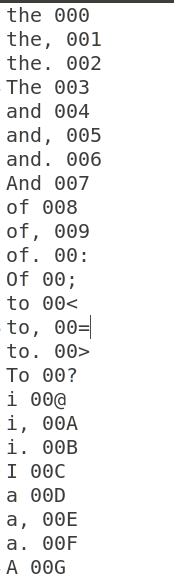


Figure 3

(2) Encode to 3 bytes

An encode book file is loaded into a hashmap (unordered\_map in c++) variable, whose key is one English word or pattern and value is its corresponding 3 ASCII characters. After that, I encode the text of PDF file based on the hashmap. There is a possibility that some words may not exist in the hashmap. At this time, I need create a dynamic zone to note these words with their dynamically generated code.

Due to the average O(1) complexity for finding a key in hashmap, the complexity for the whole encoding process is O(n).



(3) Compress encoded file by bzip2

I should have integrated bzip2’s API to my program. However, there’s some difficulty understanding the source code. So when I test it, I just use command “bzip2 filename” in the terminal to do this.

Decompression Procedure:

(1) Decompress by bzip2

The ultimate compression file is decompressed by bzip2 to get the encoded file.

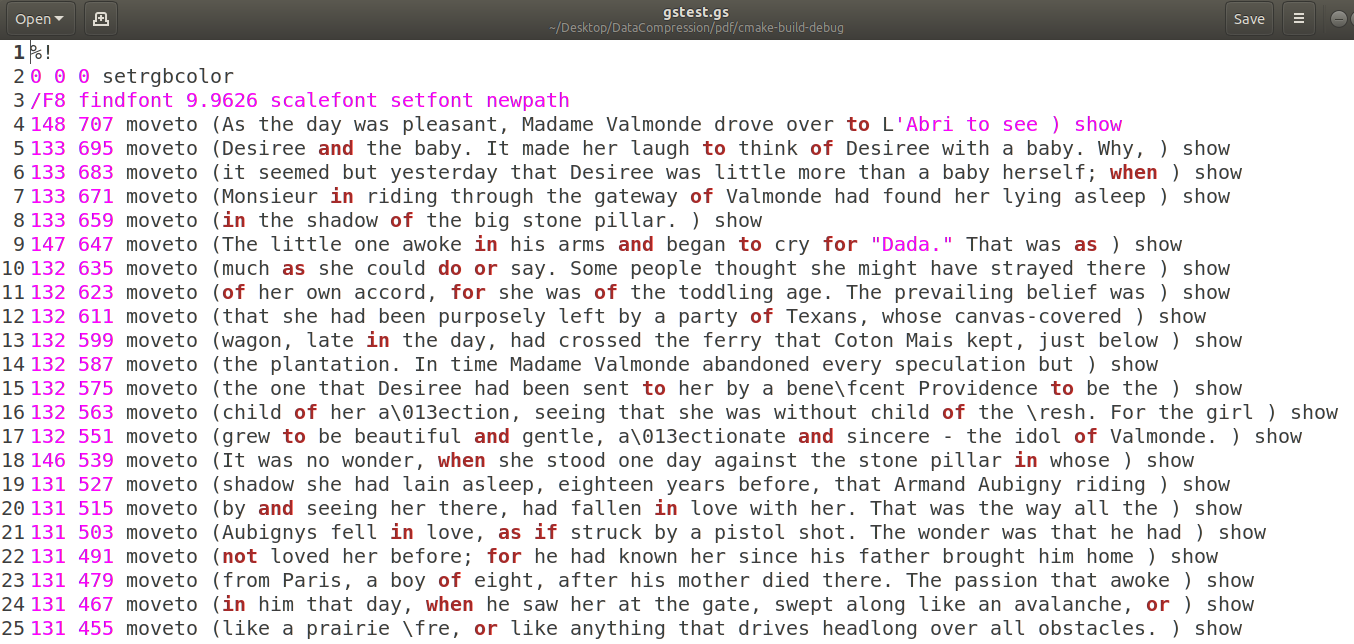
(2) Decode to plain contents

The encode book file is loaded into a hashmap (unordered\_map in c++) variable reversely, whose key is 3 ASCII characters and value is corresponding English word or pattern. Then the words in the dynamic zone are also loaded to the hashmap.

Due to the average O(1) complexity for finding a key in hashmap, the complexity for the whole decoding process is O(n).

(3) Generate postscript

All the text in PDF file with their coordinate displacement is written as postscript format, i.e. *x y moveto (text) show.*



(4) Convert postscript to PDF

I should have integrated ps2pdf API to my program. However, I only find the source code for Windows, instead of Linux. So when I test it, I just use command “ps2pdf filename” in the terminal.

Multithreading:

Because parsing, storing and loading dictionary are three independent steps, I run three threads concurrently to make the whole program faster. And the parsing thread and the storing thread are connected by a shared queue of data, so a mutex lock is inevitable. We can not execute next step - compressing, until all the three threads are done.

Result

(1) Linux Command Line Interface

(pdf is the executive file)

For compression: ./pdf -c input.pdf encodedFile

For decompression: ./pdf -d encodedFile original.pdf



(2) Performance

I choose a short article call shortStory.txt, whose size is 12.0 KB. Converted to PDF file, its size is 35 KB. After compressing, the ultimate file is 7.2 KB, which has a compression ration of 18.46% .

(3) Future Improvement

The encode book only includes high frequent English words, which causes the dynamic zone large. If the encode book includes the complete English dictionary, the dynamic zone would be much smaller.

Github repository link:

[https://github.com/StevensDeptECE/DataCompression](https://github.com/StevensDeptECE/DataCompression/tree/PDF)