Insert here your thesis' task.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Framework for Extraction of Wikipedia Articles Content

Oleksandr Husiev

Department of theoretical computer science
Supervisor: Ing. Milan Dojčinovski, Ph. D.

July 7, 2020

Acknowledgements

I would like to thank my family and friends for support during writing this thesis.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on July 7, 2020 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Husiev, Oleksandr. *Framework for Extraction of Wikipedia Articles Content.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstrakt

Klíčová slova NIF, RDF, propojená data, poškrábání webu.

Abstract

This thesis is concerned with the extraction of Wikipedia content for a DBpedia, a crowd-sourced community effort. The main goal of this thesis is to develop a framework for extraction of Wikipedia articles content, structure and annotations. The framework should process large Wikipedia XML dumps in several popular languages.

Keywords NIF, RDF, linked data, web scraping, knowledge graph.

Contents

List of Figures

List of Tables

Introduction

Motivation and objectives

Knowledge bases are growing up in importance as Web and enterprise search engine. At the moment knowledge bases cover only specific niches, and are not useful outside of their main purpose. This thesis, as part of a broader DBpedia initiative, has an objective to structure information, store it in a machine-readable form and provide better ways for information to be collected, organized, searched and utilized.

A DBpedia is a knowledge base, which information is organized as an open knowledge graph. DBpedia data is served as a Linked Data, which opens a new way to access Web for applications: via browser, automated crawlers or complex SQL-like queries. For example, current technologies do not allow to combine information about cities, criminal rate, climate and open job postings into one search. The goal of DBpedia is to allow such queries to happen.

Problem statements

The main sight of the thesis is to extract structured content from the Wikipedia articles. This content can be divided into several main parts: context, structure and links. Context is the text itself, structure is the way the text is organized and split into sections, subsections and paragraphs, and links are either links to other Wikipedia articles or external websites. Additionally, it is important to take care about the dates of article publication, clean up the non-standard articles and sections and cover other Wikipedia languages.

The main problem, however, is the current size of the Wikipedia. Only full English Wikipedia dump that includes only text and XML structures takes around 16 GB of space. Therefore, a thesis should research a design and implementation of not only functional, but an efficient parser.

Chapter 1

Data compression

This thesis was submitted at Czech Technical University in Prague (see Figure 1.1).

1.1 Notions and definitions

The source *message* consists of *source units*, which can be defined as *alphabet symbols* or sequence of alphabet symbols *(word, string, phrase)*, where alphabet S is a finite and non-empty set of symbols. The *code unit* is defined as a sequence of bits. The empty sequence of symbols is called *empty string* and it is represented by ε. The set of all symbols from alphabet S, free of empty string, is represented by S^+. The *concatenation* of two phrases $x, y \in S$ is represented by $x.y$.

FAKULTA
INFORMAČN
TECHNOLOG
ČVUT V PRAZ

Figure 1.1: CTU logo

Code is a triple $K = (S, C, f)$, where

- S is a finite set of source units,
- C is a finite set of codewords (code units),
- f is an injective mapping from S to C^+.

The mapping f does not map two different source units from S to the same codeword from C, as shown Formula 1.1.

$$\forall a_1, a_2 \in S, a_1 \neq a_2 \Rightarrow f(a_1) \neq f(a_2) \tag{1.1}$$

The string $x \in C^+$ is *uniquely decodable* with f, when Formula 1.2 is true.

$$\forall y_1, y_2 \in S^+, f(y_1) = f(y_2) = x \Rightarrow y_1 = y_2 \tag{1.2}$$

The code $K = (S, C, f)$ is *uniquely decodable*, when all strings $x \in C^+$ are uniquely decodable in f. The code K is called *prefix code*, when none of codewords is a prefix of another codeword. If all codewords are exactly n symbols length in code K, the code K is called *block code*. The prefix codes and block codes are often used by compression algorithms because of their unique decode-ability during the left-to-right reading (decoding).

$$\textit{Compression ratio} = \frac{\textit{Length of compressed data}}{\textit{Length of original data}} \tag{1.3}$$

The compression efficiency can be expressed by many units of measure. The amount of data reduction gained by the compression process is *compression ratio*. This compression ratio is a ratio of the length of compressed data to the original size of data (Formula 1.3). For example, the compression ratio is measured in *bpb* (bits per bit), *bpc* (bits per character) or *bpp* (bits per pixel).

The compression algorithms use specific *compression models* to encode the data. For example, these models could follows:

- The algorithm assigns code to each source unit irrespective to its position (statistical compression methods).
- The Markov's model of n-th order look at previous n source units to assign code. The simplist of this codes, 0th order, are mentioned above.
- The models based on finite automata.

Table 1.1: RLE example

Method	Data to encode	Encoded data
RLE 1	*babaaaaabbbaabbbba11aaa*	1*b*1*a*1*b*5*a* 3*b*2*a*4*b*1*a* 2*1*3*a*
RLE 2	*babaaaaabbbaabbbba11aaa*	*bab*@5*a*@3 *baa*@4*ba1 1*@3*a*

1.1.1 Entropy

The entropy is only theoretical minimal length but it is possible to reach this border in some special cases. It is very difficult to measure real entropy of source message in common usage, because not only statistical model of 0th order (context of source units with length 1) exists. For example, the probabilities of appearances of source units pairs (context of source units with length 2) are considered in 1st order statistical model.

1.1.2 Classification

Data compression/decompression is classified by many factors. The first classification depends on information loss during the compression process. The data compression, as data compression algorithms, is divided into two main parts:

- *Lossy*—some information loss is possible. These compression methods achieve higher compression (better compression ratio) but they are useful only in special cases (images, video, voice...).
- *Lossless*—information is acquired in original form. These compression methods are best suited for data where loss is unacceptable (documents, programs, scripts...).

1.2 Elementary methods

1.2.1 RLE

1.3 Dictionary methods

1.3.1 LZ77

It is obvious that the value of offset and the match length have to be limited to some constant. The usually chosen value for the match length is 255 (8 bits) and the offset is commonly encoded on 12–16 bits, so the search buffer is limited to 4 095–65 535. In so far that there is no need to remember more than 65 535 already encoded symbols during compression process.

Chapter 2

Implementation and testing

2.1 Details of realised tests

The scaliness of implemented algorithms were tested on chosen files from Calgary and Canterbury corpus too. The framework to scale the files is different of the testing one. Each file is equally split to 1 000 parts (files with number of lines less than 1 000 are split to 100 parts) by number of lines—the n th part consists of $\frac{n}{1000}$ ($\frac{n}{100}$) lines. Each test runs only 100–10 times (depending up the n—the size of compressed data) because of time complexity. This cycle runs 10–100 times (10 was chosen for this tests) and the minimum time is taken. The file splitting by the number of lines was chosen because of the character of algorithms (word-based)—the splitting by the block of the same size is not so predicative.

There are parameters of the computer used for tests shown in Table 2.1.

2.2 Integers distribution and encoding

The integers encoding of indexes of phrases from the word (non-word) dictionary is possible cause of the only average results of compression ratio of implemented algorithms. The decision is to get the distribution of indexes during the encoding process (and decoding process too). The length of indexes located in shown graphs is only hypothetical—the binary code with minimal length.

Table 2.1: Testing computer parameters

Part	Description
CPU	2.2 GHz AMD Athlon(tm) 64 Processor 3200+
MEM	2.5 GB
OS	x86_64 GNU/Linux Fedora release 7 (Moonshine)

The graphs of index distribution also shows the differencies between the algorithms with sorted dictionaries (*WLZWS* and *WLZWES*) and the algorithms with unsorted dictionaries (*WLZW* and *WLZWE*). The most frequently used phrases are moved to the front of dictionary in algorithms with sorted dictionary so they get lower indexes. This feature is demonstrated by the growth of number of indexes at the beginning of the distribution. The compression process of algorithms with sorted dictionaries becomes more efficient when the code with variable length of code words (Fibonacci code) is used but the compression efficiency is supposed to be the same at the transition from the *WLZWE2* algorithm to the *WLZWES2* algorithm—the encoding by block code.

Conclusion

The word-based dictionary data compression algorithms (a part of lossless data compression) are the subject of this thesis. The lossless data compression is a very important field of research because the data compression allows to reduce the amount of space needed to store data.

The background of data compression field was presented in Chapter 1. There are basic notions and definitions followed by description of character-based dictionary algorithms. The word-based dictionary compression methods were investigated and discussed at the end of this chapter too.

The testing of memory used during compression and/or decompression process is one of the possibilities of further research. The experiments with files of greater size or multilingual files could be also good opportunity to gain new improvements of algorithms. The static part of dictionaries could improve the compression efficiency too.

The implemented methods achieve fairly good compression ratio (25–30% at large files) with acceptable compression and decompression time. There are possibilities of further improvements especially at semi-adaptive methods. However, the gain of these improvements is not good enough to top the compression efficiency of other lossless data compression methods (context methods from PPM family). The results of implemented algorithms were not as good as it was expected but the work on this thesis showed new ways of possible further research—word-based version of grammar-based compression algorithms and another possibilities in the field of word-based context methods of data compression.

The Gnuplot 4.2 utility was very useful for generation of graphs in this thesis. There was the drawing editor Ipe 6.0 used for figures creation.

Bibliography

[1] Powel, M. The Canterbury Corpus [online]. November 2001, [Cited 2011-10-12]. Available from: http://corpus.canterbury.ac.nz/index.html

Appendix A

Contents of CD

Visualise the contents of enclosed media. Use of `dirtree` is recommended. Note that directories src and text with appropriate contents are mandatory.

```
readme.txt ..... the file with CD contents description
data ..... the data files directory
    graphs ..... the directory of graphs of experiments
        *.eps ..... the B/W graphs
        *.png ..... the color graphs
        *.dat ..... the graphs data files
exe ..... the directory with executable WBDCM program
    wbdcm ..... the WBDCM program executable (UNIX)
    wbdcm.exe ..... the WBDCM program executable (Windows)
src ..... the directory of source codes
    wbdcm ..... the directory of WBDCM program
        Makefile ..... the makefile of WBDCM program (UNIX)
    thesis ..... the directory of LaTeX source codes of the thesis
        figures ..... the thesis figures directory
        *.tex ..... the LaTeX source code files of the thesis
text ..... the thesis text directory
    thesis.pdf ..... the Diploma thesis in PDF format
    thesis.ps ..... the Diploma thesis in PS format
```