Łódź, 13 of January, 2009

Review of the doctoral dissertation of Mgr. Jan Lánský, entitled *Syllable-based Compression*, supervised by Prof. RNDr. Jaroslav Pokorný

by Szymon Grabowski, PhD
Technical University of Łódź, Poland
Computer Engineering Dept.

Text compression is an established research area, with more applications appearing in the last 5–10 years due to interesting discoveries allowing to combine compression with efficient pattern matching.

It is well known that in text compression much depends on the alphabet representation of textual data, and the compression ratios and range of supported functionalities usually change if we once perceive the data as a sequence of characters, then a sequence of words, or q-grams, or XML entities etc.

Mr Lánský in his research rather consequently (he started for his MSc, defended in 2005) explores yet another method of text parsing, which considers syllables as the atomic tokens, an idea initiated by Üçoluk and Toroslu in 1997. His list of publications, included in the thesis, contains 20 refereed papers (but only one solo paper among them), mostly in English and mostly conference ones (incl. DCC posters, DATESO papers, one SOFSEM paper).

The thesis consists of 123 pages, 9 chapters (including Introduction and Conclusion), has bibliography of 118 entries, including in that number the Author's publications given in a separate list.

In a short introduction Mr. Lánský mentions the research topics, summarizes own achievements and lists the papers they were presented in (prior to the thesis) and then discusses contents of the thesis, chapter by chapter. The following two chapters, *Lossless Compression* and *Related Works*, present the fundamentals of lossless data compression, and then focus on the state of the art in text compression in various flavors and scenarios (XML, language-dependent methods, small text files, and so on). Chapter 3 (*Related Works*) is however too short, esp. in Section 3.4 (*XML Compression*), where some relevant papers are not cited (more on this later in the review).

The notion of a syllable, although respected by and useful in most European languages, is rather vague. How to parse words into syllables is the topic of Chapter 4, partly based on the Authors' MSc thesis. There are four word decomposition algorithms presented. There is a plausible motivation given in Sect. 4.2: *One of the reasons why we selected syllables is that documents contain less unique syllables than unique words*, supported with distinct word counts and syllable counts for exemplary files in English and Czech (plus some related information in Table 7.5), but it would be even better to show, how the number of unique tokens in the syllable model scales in those languages. I do not know if a Zipfian law holds also in this model (and if so, with what empirical parameters), and do not even know if such measurements have been done so far, but they should be a start for any serious research in this direction.

The next two chapters are dedicated to text compression of small files, general (plain text) and XML ones. Chapter 7 deals with dictionary compression, where the items are either words or syllables. The subject of Chapter 8 is compression of large text files. In the experiments, texts in English, Czech, Slovenian and German are used. I am going to sum up the Author's achievements from individual chapters, one by one, with comments.

The chapter *Small Text Files Compression* describes two syllable-based compressors, LZWL and HuffSyllable, being adoptions of the classic methods to the syllable setting. However, those algorithms were introduced yet in Author's MSc thesis (what Mr. Lánský admits, of course), and it is not specified in what aspect (if any) the current implementations are modified. In order to improve the compression (on small files chiefly), the Author initialized the dictionary with the most frequent syllables (and their estimated frequencies) for a given language. This works not only for LZWL, but also for Huffman, as an adaptive Huffman was used in HuffSyllable (together with two helper static Huffman models). Different classes of syllables are considered: lower-case, mixed-case, numeric, etc., and the previously encoded syllable is a predictor of the class of the next syllable, which is reported to improve compression. Experiments were carried out on English and Czech plain texts of sizes varying from a few KB to a few MB, and although (as expected) syllables fare better than characters, they are beaten by words, with the exception of the syllable-based LZW variant for Czech. Unfortunately, LZWL on syllables compresses worse than e.g. order-0 Huffman on words, so on the overall I find those results disappointing. One extra conclusion could be that there are seemingly strong inter-syllable correlations in words, which are obviously lost in order-0 syllable model. Some other compared compressors include bzip2 (consistently best results) and WRT (by Skibiński), in mode -0, which unfortunately was run with *compress* (LZC compressor) as its backend, while this mode was designed for zip-like compressors from LZ77 family (on the other hand, this is consistent with the LZW-based algorithm proposed by the Author, but what is the point of preferring LZ78 over LZ77-style compressors?).

Yet another experiment along these lines was to compare character, syllable, word and q-gram models with BWT-based compression, which was treated separately since (as opposed to the previous experiment) this method has no static model initialization. Now, up to (roughly) 200 KB the character model wins, then syllables take the lead. The relatively poor performance of the word models (for all three languages in these tests: Czech, English and German) is a little surprise for me, but of course the dictionary encoding cost must be an important factor. The range of tested compressors is surprisingly small, and some algorithms from the literature have not been even cited. Interesting word-based results with a semi-static dictionary were reported in Kadach's PhD (1997). For small text files, a known and quite successful idea (e.g. (Skibiński et al., 2005), (Cleary and Teahan, 1996), Teahan's PhD, (Abel and Teahan, 2005)) is replacing frequent q-grams (q=2...4, or perhaps only q=2, to simplify things) in a given language with 1-byte codes in preprocessing, I would be glad to see such comparisons also (partitioning the whole text into 3-grams or 5-grams, which was checked in experiments in this chapter, is a different and obviously worse idea). Yet another suggestion for extending comparisons is Shkarin's Durilca, which is extremely competitive on English texts, due to a built-in statistical model for its PPM engine (it can be pointed out that the Durilca's "hot start" idea is somehow related to the use of characteristic syllables in this thesis – i.e., they both use static initialization).

The next chapter, on XML compression of small files, presents two algorithms, XMLSyl and XMillSyl. The former makes use of a widely applied idea in XML compression, of encoding XML tokens with short (1-byte, if possible) codewords. Then also a syllable model is used (the backend coder used is LZWL). The latter is based on the well-

known XMill compressor, again with LZWL as a backend for most XML element containers. The encoding of the XML structure (extracted via Expat, an open-source SAX parser) is presented in detail, with an example, which I find clear and instructive. Alas, the achieved compression results are again not convincing. Of course, syllable-based algorithms (and specialized XML compression algorithms) do win with general purpose compressors but this was obvious. In comparison with XMill (with Deflate as the backend coder), the presented algorithms compress worse (although in some cases the loss is relatively little, about 7% loss for XML.Syl on textual XML documents). It was not mentioned that XMill is not the last word in XML compression, to name only XML.PPM (this compressor is cited), SCMPPM, XBzip, and XWRT variants. No time measurements were presented here.

Chapter 7 is dedicated to the problem of compressing the dictionary. This is perhaps the most interesting part of the thesis, which does not mean it is without flaws. Mr. Lánský notices that compressing small text files in a large alphabet model (e.g., words or syllables) is constantly threatened by the model encoding overhead, hence efficient representation of the model (i.e., the dictionary of used symbols / tokens) is crucial.

Alas, Subsect. 7.4.1 (*Existing methods*) is quite disappointing. The Author claims that two approaches for dictionary compression are widely used, but it is unclear to me what he means by the approach "based on coding of a succession of strings (words or syllables) contained in it". The second mentioned approach is very trivial. Other algorithms (which are described in Sect. 3.5, where the reader is referred to) are now commented with a remark that they have not been used "for the compression of the used alphabet" – I understand this means those algorithms (perhaps some of them should be named here directly again) have only been tested on a large (presumably rather complete) dictionary for a given language, not on vocabularies of small or medium-size files. I find this excuse only partly convincing. The net effect is that the Author has not verified experimentally any non-trivial algorithms known from the literature. Now, let me point a claim from Sect. 3.5: *Complex methods like applying combination of prefix and suffix compression (like in [21]) seems to be useful especially for dictionaries with long elements (what is not the case of syllables)* – how can we be sure about it (for syllables), without experiments? A simple reference dictionary encoding scheme could be the combination of front coding (FC) and some general purpose compressor (e.g., zip, PPMd); the Author presents the results of FC but without any backend coder (only the prefix length and suffix length of the string are Elias gamma encoded).

The Author's algorithms, TD1 and its refinements, are presented in Subsect. 7.4.2. I find their design principles logical, but some implementation issues could probably be improved. It is a pity that the modifications mentioned in *9.2 Future Work* and planned to be presented as TD4A up to TD4H variants (by the way, in Subsect. 7.4.5 the lists of newer variants is even longer, includes TDAR, ..., TD4HR), have not yet been already implemented (or published?), as they belong to obvious choices. It is hard to guess how practical TD3, the most efficient of the presented algorithms, is, as in the experiments no other specialized algorithms were tested (the non-efficient PS algorithm is also from Jan Lánský).

A substantial part of Chap. 7 is spent in quest of so-called characteristic syllables, useful in the syllable-based compression algorithms presented earlier. Six pages were needed to present a genetic algorithm finding those characteristic syllables, but its crops are mediocre: hardly better (if any) than a very simple heuristic called A20 (i.e., label as characteristic all the syllables that occur at least once in at least 20% of all documents).

The research presented in Chapter 8 was motivated by EGO-THOR full-text search system. The result of Mr. Lánský's research for this problem, dealing with large text files, is a flexible (modular) compression system, called XBW. The first parameter to choose is the alphabet type: characters, syllables or words. Then, the appropriate parser does its duty, if the user's choice were words or syllables, the dictionary of extracted items is created / encoded

(using a method from the previous chapter), and the main compression engine starts to work: BWT transform, followed by MTF step, then RLE / LZC / LZSS (variant of) / PPM modeler, and finally arithmetic or Huffman coding is performed. Combinations of all stages don't make sense (as post-BWT modeling must be very specific, which is known from the seminal works of Burrows and Wheeler (1994) and Fenwick (mid-1990s), and for example running LZSS on BWT or BWT+MTF sequence is likely to produce horrible results), but most of the components can be turned off (which is however not presented in Fig. 8.1). I wonder why the PPM component can only be used with A, B and C escapes; especially PPMA and PPMB are antique variants, but even C escapes were proposed in 1990! The BWT construction was achieved with Kao's and Kärkkäinen–Sanders' algorithms, which belong to efficient ones. Interestingly, Move-To-Front was implemented with splay trees, which is much faster for large alphabets (words, syllables) than the naive (but commonly used) approach (but it would be interesting to see how much this implementation is faster in practice, both for word and syllable models...). It should also be pointed out that MTF was a standard in the times of influential Fenwick's papers (which are not cited), i.e. in 1995–1996, but later many more sophisticated post-BWT modeling techniques were proposed; only in 8.6 *Future Work* two of such techniques are cited, but it is a pity they have not been implemented so far.

One of the crucial components of the system is an XML parser, which handles also non well-formed documents, which I find a practical design decision. Specialized XML compressors: XMLPPM and XMill, were not tested (note on p. 105) since they do not handle ill-formed XML files. Still, I would like to know what percentage of the tested files were well-formed ones and how the Author's algorithms compare to e.g. XMLPPM on them.

The Author reports that his implementation of PPM on words was unable to finish working within an hour; it must be due to serious algorithmic flaws in the implementation, and maybe also improper parameter choice (PPM on words should work in small order, e.g. 2). Later (Sect. 8.5.1) we read that that for a 5 MB excerpt of a given text collection, the PPM on characters needs one minute to compress, while on syllables it needs one hour, and on words even 2 hours. This is, in my conservative estimation, at least 2-3 orders of magnitude worse than what is possible (for the syllable and word model)! Also one minute for the character model starkly contrasts with less than 2 seconds (in my estimation) needed by PPMd -o5 (which is also much more sophisticated and thus compresses much better than PPM with C escapes).

Experiments include a number of third-party compressors, namely gzip, rar 3.7 -m5, ABC, szip, PPMonstr and bzip2, but it was not specified what compression methods (i.e., PPM or BWT, or LZ77 etc.) are used by those programs. Variant I of PPMonstr was tested but since May of 2006 var. J rev. 1 exists (with stronger but slower compression). Even worse, PPMonstr was tested using "-o1 -m256" switches while the minimum order with which PPMonstr works is 2 (i.e. -o2 switch)! (It seems that PPMonstr -o1 uses order-2 though. But for redundant data, like XML, order-2 is MUCH too little.) Also, the cited Durilca version [94] is outdated (v0.5 exists for a long time).

Experimental results seem quite good but again we cannot fairly tell, since a (big) part of XBW's advantage over e.g. bzip2 (Table 8.14) comes from a much larger block used than 900 KB of bzip2. PPMonstr is a very strong but rather impractical PPM implementation (and it was used with order-2 only, as mentioned above); adding PPMd (also from Shkarin), working at least in its default order-4 mode, would be more useful.

From those experiments it is again dubious if syllables are better than words: for Czech and Slovenian XBW on syllables was slightly better than on words, but for English the word model was slightly better. Let me note that the parser for syllables was significantly slower than for words (and also BWT on words works faster in Author's experiments than on syllables, due to decreased number of tokens). I don't like the idea of giving totals (should

rather be "average"...) in those tables (8.7–8.9 and 8.11–8.14) as calculating averages over the results for different (and arbitrarily chosen, in a way) languages makes little sense.

Chapter 9 concludes the thesis and points out avenues for future work. The (rough) conditions under which the syllable model may dominate over the word model are listed, but remembering about incomplete experimental comparisons, we cannot be so sure about it. The whole thesis lacks a broader (application-oriented) overview of research issues and possible scenarios of text compression. Since, of course, compression ratio is not everything, are there any particularly attractive features of the syllable-based model? Compression speed? Dynamism? Search capabilities? Those issues are not discussed. Also, I find the claim on XBW: "A very successful method" (p. 111) a really bold one.

Minor notes.

The thesis is overall tidy, its language mostly correct, still I spotted several typos, language mistakes or awkward phrases etc., which are listed below.

Typos: *worses* → worsens (p. 26), *charactersitc* (p. 85), *powerfull* (p. 106).

p. 16 *...is not usually very good.* → ...usually is not very good.

p. 17, *during the construction of the Huffman tree is not only way to go.* → ...is not the only way to go.

The use of articles (*a* / *the* / –) is dubious in many places throughout the whole thesis.

p. 19, *Dictionary-based methods are especially suitable for files with many repetitious long strings (max. 5-15 characters)*... It is true, but why the limitation (5-15)? Those methods are even more suitable for files with VERY LONG repeating strings...

p. 24, About the spaceless model by de Moura et al.: *The word is often followed by a special non-word space.* Why "special"? It is the most common case of a word followed by a space.

p. 26, *Extensive and closely described overview of word-based methods can be found in [97].* Exactly the same sentence was written 2 pages earlier.

p. 30, *There is proposed a new data structure, the burst trie [49],* ... Why "a new data structure" as the paper was published in 2002?

p. 51, there should be no period after a subsection title.

p. 71, the word *need* / *needs* used 5 times in the first 5 lines of Sect. 7.4.

p. 83, in the first two paragraphs of Subsect. 7.5.1 the word "characteristic" occurs 9 times.

Table style in Chapter 6 is different than in the rest of the thesis.

p. 101: *These results show effect of XML, which improves the compression ratio by*... Clumsy sentence (XML improves the compression ratio?).

It is time for a conclusion. I am sorry to say that in my opinion **the reviewed work by Jan Lánský does not fulfill the requirements for a Ph.D. thesis**. The elements missing, or existing in unsatisfactory forms, are: a thorough review of the literature, comprehensive and consistent experimental tests with in-depth discussions of their methodologies, and putting the work in wider (application-oriented) context. The thesis does not contain theoretical analyses (or even qualitative empirical observations) of the syllable-based model, for a deeper understanding under what conditions it can dominate in compression over both word- and character-based models. The proposed algorithms are not very original (quite mechanical combinations of known, often old, ideas), their results usually disappointing, and it is not clear from the thesis what could be their possible real-world applications, if competitive existing algorithms are also considered.