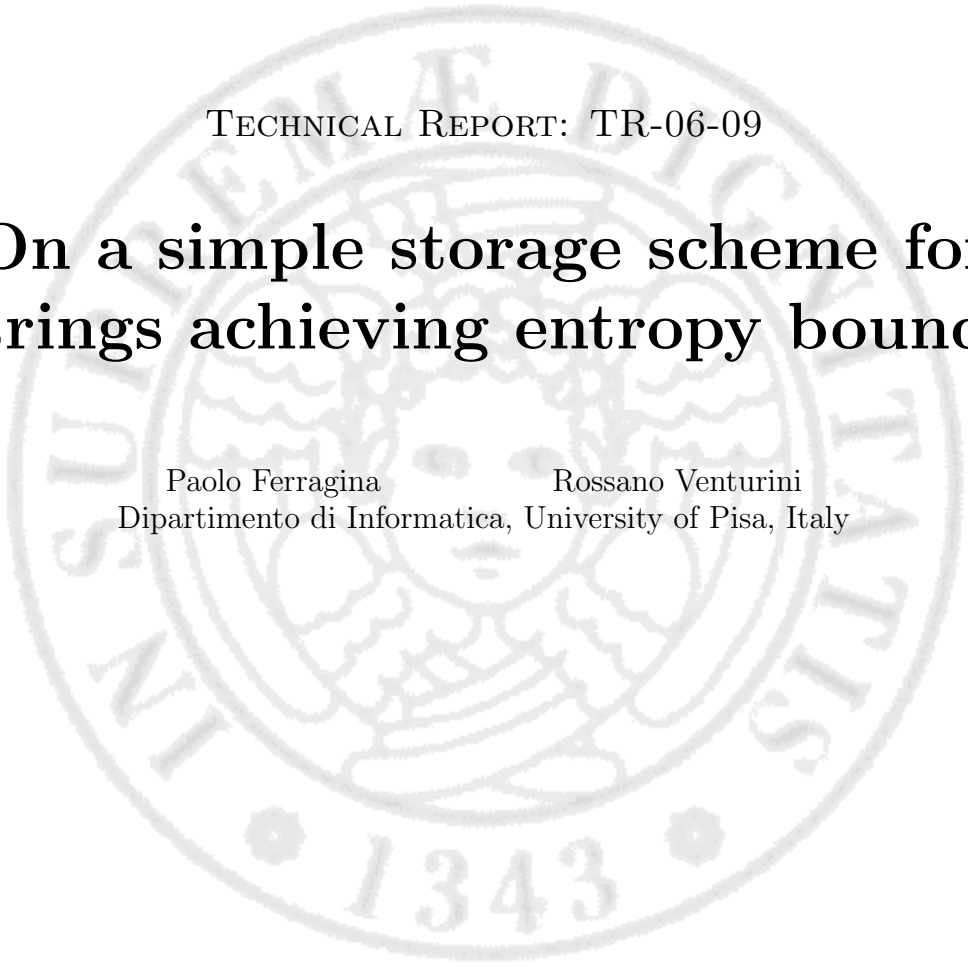
Università di Pisa
Dipartimento di Informatica



Technical Report: TR-06-09

On a simple storage scheme for strings achieving entropy bounds

Paolo Ferragina Rossano Venturini
Dipartimento di Informatica, University of Pisa, Italy

June 2006

Address: via F. Buonarroti 2, 56127 Pisa, Italy. Tel: +39 050 2212700 Fax: +39 050 2212726

On a simple storage scheme for strings achieving entropy bounds*

Paolo Ferragina Rossano Venturini
Dipartimento di Informatica, University of Pisa, Italy

June 2006

Abstract

In this note we propose a storage scheme for a string $S[1,n]$, drawn from an alphabet Σ, that requires space close to the k-th order empirical entropy of S, and allows to retrieve any ℓ-long substring of S in optimal $O(1+\frac{\ell}{\log_{|\Sigma|} n})$ time. This matches the best known bounds [11, 5], via the use of binary encodings and tables only.

1 Introduction

Starting from [3], the design of compressed (self-)indexes for strings became an active field of research (see [9]). The key problem addressed in these papers consists of representing a string $S[1,n]$ drawn from an alphabet Σ within compressed space, and still be able to answer various types of queries (e.g. substring, approximate, ...) in efficient time, without incurring in the whole decompression of the compressed data. In these results, compressed space usually means space close to the k-th order empirical entropy of S,[1] and efficient time means something depending on the length of the searched string and as much independent as possible of S's length. Various trade-offs are known, and for them we refer the reader to [9].

Recently, Sadakane and Grossi [11] addressed the foundational problem of designing a compressed storage scheme for a string S which is *provably better* than storing S as a plain array of symbols. Here, the query operation to be supported is the retrieval of any ℓ-long substring of S in optimal $O(1+\frac{\ell}{\log_{|\Sigma|} n})$ time. Previous solutions [9], based on compressed (self-)indexes, incurred in an additional sub-logarithmic time overhead. Conversely, the Sadakane-Grossi's storage scheme is able to achieve the optimal time bound by occupying a number of bits upper bounded by the following function[2]

$$nH_k(S) + O(\frac{n}{\log_{|\Sigma|} n}\,(k\log|\Sigma| + \log\log n)\,) \tag{1}$$

This storage scheme is based on a sophisticated combination of various techniques: Ziv-Lempel's string encoding [13], succinct dictionaries [10], and some novel succinct data structures for supporting navigation and path-decodings in Lz-tries. Since storing S by means of a *plain* array of symbols takes $\Theta(n\log_{|\Sigma|} n)$ bits, the scheme in [11] is effective when $k = o(\log_{|\Sigma|} n)$.

More recently, González and Navarro [5] proposed a simpler storage scheme achieving the same time and space bounds above, but exploiting a statistical encoder (namely, Arithmetic) on some of S's substrings. Unlike [11], this storage scheme requires to fix the order k of the entropy bound in advance.

In what follows, we propose a very simple storage scheme that: (1) drops the use of any compressor (either statistical or Lz-like), and deploys only binary encodings and tables; (2) matches the space bound in Eqn. (1) simultaneously over all $k = o(\log_{|\Sigma|} n)$.

*Emails: ferragina@di.unipi.it and rossano.venturini@swissinfo.org. Partially supported by Italian MIUR grants Italy-Israel FIRB "Pattern Discovery Algorithms in Discrete Structures, with Applications to Bioinformatics", PRIN "Algorithms for the Next Generation Internet and Web" (ALGO-NEXT), and by Yahoo! Research grant on "Data compression and indexing in hierarchical memories".

[1] This is a lower bound to the space achieved by any k-th order compressor.

[2] As stated in [5], the term $k\log|\Sigma|$ appears erroneously as k in [11]. We therefore use the correct bound in this note.

2 Some background

Let $S[1,n]$ be a string of length n drawn from the alphabet $\Sigma = \{a_1, \ldots, a_h\}$. For each $a_i \in \Sigma$, we let n_i be the number of occurrences of a_i in S. Let $\{P_i = n_i/n\}_{i=1}^{h}$ be the empirical probability distribution for the string S. The *zero-th order empirical entropy* of the string S is defined as[3]

$$H_0(S) = -\sum_{i=1}^{h} P_i \log P_i \tag{2}$$

For any length-k string w, we denote by $\vec{w}_S$ the string of single symbols following the occurrences of w in S, taken from left to right. For example, if $S = \texttt{mississippi}$ and $w = \texttt{si}$, we have $\vec{w}_S = \texttt{sp}$ since the two occurrences of `si` in S are followed by the symbols `s` and `p`, respectively.
The k-th order empirical entropy of S is defined as:

$$H_k(S) = \frac{1}{n} \sum_{w \in \Sigma^k} |\vec{w}_S| \, H_0(\vec{w}_S). \tag{3}$$

Not surprisingly, for any $k \geq 0$ we have $H_k(S) \geq H_{k+1}(S)$. The value $|S|H_k(S)$ is a lower bound to the output size of any compressor that encodes each symbol with a code that only depends on the symbol itself and on the k immediately preceding symbols.

For simplicity of exposition, we will use $\mathcal{B} = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$ to denote the infinite set of binary strings *canonically* ordered, where ϵ is the empty string.

3 Our storage scheme for strings

Let $S[1,n]$ be a string drawn from an alphabet Σ, and assume that n is a multiple of $b = \lfloor \frac{1}{2} \log_{|\Sigma|} n \rfloor$. If this is not the case, we append to S the missing characters taking them as the special *null symbol*.[4] We partition S into blocks S_i of size b each. Let $\mathcal{S}$ be the set of distinct blocks of S. The number of all blocks is $\frac{n}{b}$; the number of distinct blocks is $|\mathcal{S}| = O(|\Sigma|^b) = O(n^{1/2})$.

The encoding scheme. We sort the elements of $\mathcal{S}$ per decreasing frequency of occurrence in S's partition. Let $r(S_i)$ be the rank of the block S_i in this ordering, and let $r^{-1}(j)$ be its inverse function (namely, the one that returns the block having the given rank j). The storage scheme for S consists of the following information.

- Each block S_i is assigned a codeword $\texttt{enc}(i)$ consisting of the binary string that has rank $r(S_i)$ in $\mathcal{B}$. Of course, $\texttt{enc}(i)$ is not a *uniquely decodable code*, but the additional tables we build below will allow us to decode it in constant time and within a space bounded by Eqn. (1).
- We build a bit sequence V obtained by juxtaposing the binary encodings of all S's blocks in the order they appear in S. Namely $V = \texttt{enc}(1) \cdots \texttt{enc}(\frac{n}{b})$.
- We store r^{-1} as a table of $O(|\Sigma|^b)$ entries, taking $O(|\Sigma|^b \log n) = o(n)$ bits.
- To guarantee constant-time access to the encodings of S's blocks and to ensure their decodings, we use a *two-level* storage scheme for the starting positions of `enc`s (see [8]). Specifically, we *logically* group every $c = \Theta(\log n)$ contiguous blocks into one *superblock*, having thus size $bc \log |\Sigma| = \Theta(\log^2 n)$ bits. Table $T_{Sblk}[1, \frac{n}{bc}]$ stores the starting position of the encoding of every super-block in V, and table $T_{blk}[1, \frac{n}{b}]$ stores the starting position in V of the encoding of every block *relative* to beginning of its enclosing super-block. Note that the starting position of each super-block is no more than $|V| = O(\frac{n}{b} \log n) = O(n \log |\Sigma|)$, whereas the relative position of each block is $O(\log^2 n)$. Consequently, tables T_{Sblk} and T_{blk} occupy $O(\frac{n}{bc} \log |V| + \frac{n}{b} \log \log n) = O(\frac{n \log \log n}{\log_{|\Sigma|} n})$ bits overall, and guarantee a constant-time access to every codeword $\texttt{enc}(i)$ and its length.[5]

[3] Throughout this paper we assume that all logarithms are taken to the base 2, whenever not explicitly indicated, and we assume $0 \log 0 = 0$.

[4] This will add to the entropy estimation a negligible additive term equal to $O(\log |\Sigma| \ \log_{|\Sigma|} n) = O(\log n)$ bits.

[5] It suffices to compute the starting position of $\texttt{enc}(i)$ and $\texttt{enc}(i+1)$, if any.

Theorem 1 *Our storage scheme encodes $S[1,n]$ in $|V| + O(\frac{n \log\log n}{\log_{|\Sigma|} n})$ bits, which is upper bounded by Eqn. (1), simultaneously over all $k = o(\log_{|\Sigma|} n)$.*

Proof: Our proof consists of showing that V is shorter than the compressed string of [5]. The storage scheme in [5] proposes to encode each block S_i by writing its first k symbols explicitly, i.e. $k \log |\Sigma|$ bits, and by then encoding the other $b-k$ symbols via a kth order statistical compressor E (which is then ensured to have k characters to encode the next one). In the case that this encoding is longer than $\lfloor (1/2) \log n \rfloor$ bits, block S_i is written explicitly without any compression. To distinguish between these two cases, [5] keeps some extra tables and data structures.

Note that E's codewords are a subset of $\mathcal{B}$, and note that `enc` encodes the strings in $\mathcal{S}$ with the first $|\mathcal{S}|$ binary strings of $\mathcal{B}$. Given that $\mathcal{B}$ is the set of shortest codewords assignable to $\mathcal{S}$'s strings, our encoding `enc` is better than E because it follows the *golden rule* of data compression: it assigns shorter codewords to more frequent symbols. Consequently, V is shorter than the string encoded by E, which was shown in [5] to achieve the bound stated in Eqn. (1). For completeness, that proof is reported in Appendix A. ∎

We now show how to decode in constant time a generic block S_k. This will be enough to prove the result for any l-long substring of S.

We first derive the starting position $p(k)$ of the string $\texttt{enc}(k)$ that encodes S_k in V. Namely, we compute the super-block number $h = \lceil k/c \rceil$ containing $\texttt{enc}(k)$, and its starting bit-position $y = T_{Sblk}[h]$ within V. Then, we compute $x = T_b[k]$ as the relative bit-position of $\texttt{enc}(k)$ within its enclosing super-block. Thus $p(k) = x + y$.

Similarly, we derive the starting position $p(k+1)$ of $\texttt{enc}(k+1)$ in V (if any, otherwise we set $p(k+1) = |V| + 1$). We can thus fetch $\texttt{enc}(k) = V[p(k), p(k+1) - 1]$ in constant time since $|\texttt{enc}(k)| = p(k+1) - p(k) = O(\log n)$.

We finally decode $\texttt{enc}(k)$ as follows. Let v be the integer value represented by the binary string $\texttt{enc}(k)$, where $v = 0$ if $\texttt{enc}(k) = \epsilon$. Because of the canonical ordering of $\mathcal{S}$, S_k is computed as the block having rank $z = 2^{|\texttt{enc}(k)|} + v$. That is, $S_k = r^{-1}(z)$.

Theorem 2 *Our storage scheme retrieves any ℓ-long substring of S in optimal $O(1 + \frac{\ell}{\log_{|\Sigma|} n})$ time.*

Proof: The algorithm described above allows to retrieve any block S_k in constant time. The theorem follows by observing that any l-long substring $S[j, j+l-1]$ spans $O(1 + \frac{l}{\log_{|\Sigma|} n})$ blocks of S. ∎

4 Conclusions

The simplification we have proposed in this paper to the results of [11, 5] drives us to two possible considerations. One is that we now have a *class-note* solution for the string storage problem that, as deeply illustrated in [11], may find successful applications into many other interesting contexts: e.g. it may turn succinct or 0-th order entropy data structures into k-th order entropy data structures. The second consideration refers to future research. Namely, all known solutions are far from being usable in practice because of the additive term which usually *dominates* the kth order entropy term. More research is therefore needed to either achieve a space bound close to the one attainable with the k-th order compressors of the Bzip-family [7, 6, 2], for which the additive term is $O(|\Sigma|^k \log n)$ bits, or to show a lower bound related to kth order entropy, in the vein of [1, 4]. Since our storage scheme, unlike [11, 5], does not use any sophisticated data compression machinery, we are led to think that there is room for improvement!

References

[1] P. Bro Miltersen. Lower bounds on the size of selection and rank indexes. *ACM-SIAM symposium on Discrete Algorithms* (SODA), 11-12, 2005.

[2] P. Ferragina, R. Giancarlo, G. Manzini, and M. Sciortino. Boosting textual compression in optimal linear time. *Journal of the ACM*, 52:688–713, 2005.

[3] P. Ferragina and G. Manzini. Opportunistic data structures with applications. In *Proc. of the 41st IEEE Symposium on Foundations of Computer Science*, pages 390–398, 2000.
Now, *Journal of the ACM*, 52(4):552–581, 2005.

[4] A. Golynski. Optimal lower bounds for rank and select indexes. *International Collowuium on Automata, Languages, and Programming* (ICALP), 2006.

[5] R. González and G. Navarro. Statistical encoding of succinct data structures. In *Symposium on Combinatorial Pattern Matching (CPM)*, LNCS 4009, pages 295–306, 2006.

[6] R. Grossi, A. Gupta, and J. Vitter. High-order entropy-compressed text indexes. In *Proc. 14th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA '03)*, pages 841–850, 2003.

[7] G. Manzini. An analysis of the Burrows-Wheeler transform. *Journal of the ACM*, 48(3):407–430, 2001.

[8] I. Munro. Tables. In *Proceeding of the 16th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 37–42. Springer-Verlag LNCS n. 1180, 1996.

[9] G. Navarro and V. Mäkinen. Compressed full-text indexes. Technical Report TR/DCC-2006-6, Dept. of Computer Science, University of Chile, April 2006. Available at: `ftp://ftp.dcc.uchile.cl/pub/users/gnavarro/survcompr2.ps.gz`.

[10] R. Raman, V. Raman, and S. Srinivasa Rao. Succinct indexable dictionaries with applications to encoding k-ary trees and multisets. In *ACM-SIAM Symposium on Discrete Algorithms (SODA '02)*, pages 233–242, 2002.

[11] K. Sadakane and R. Grossi. Squeezing succinct data structures into entropy bounds. In *ACM-SIAM Symposium on Discrete Algorithm (SODA)*, pages 1230–1239, 2006.

[12] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, second edition, 1999.

[13] J. Ziv and A. Lempel. Compression of individual sequences via variable length coding. *IEEE Transaction on Information Theory*, 24:530–536, 1978.

A Bounding the space of [5]'s scheme

For completeness of exposition we report here the proof of [5] that shows that their storage scheme actually achieves the space bound stated in Eqn. (1).

Let us denote by f_i the frequency of occurrence of each symbol $S[i]$ given its preceding kth order context $S[i-k, i-1]$. Note that $n \times f_i$ is the number of times symbol $S[i]$ occurs after $S[i-k, i-1]$. According to the notation in Section 2, $n \times f_i$ is the number of times symbol $S[i]$ occurs within $\overrightarrow{w}_s$, where $w = S[i-k, i-1]$. It is easy to see that a (semi-static) k-order modeler can compute all the frequencies f_i via two passes over S, hence in $O(n)$ time.

Arithmetic encoding is one of the most effective statistical encoders [12]. Given the f_is, it represents the string S with a range of size $F = f_1 \times f_2 \times \cdots \times f_n$. It is well known [12] that $2 + \log(1/F) = 2 + \sum_{i=1}^{n} \log(1/f_i)$ bits are enough to distinguish a number within that range. The binary representation of this number is the Arithmetic compression of S. If we compute $\sum_{i=k+1}^{n} f_i \log(1/f_i)$, and then we group all the terms with the same kth order context, we obtain a summation upper bounded by $H_k(S)$. Additionally, since $f_i \geq 1/n$, we have that $\sum_{i=1}^{k} f_i \log(1/f_i) = O(k \log n)$. As a result, a (semi-static) kth order Arithmetic encoder compresses the whole S within $nH_k(S) + 2 + O(k \log n)$ bits.

The storage scheme of [5] compresses the blocks S_i of S individually: the first k symbols of S_i are represented explicitly, the remaining $b - k$ symbols of S_i are compressed via the k-order Arithmetic

encoder (hence using their kth order frequencies fs). This blocking approach increases the Arithmetic compression cost of the whole S, shown above, by $O((n/b)k \log |\Sigma|)$ bits, which accounts for the cost of explicitly storing $S_i[1, k]$.

To decode in constant time every block S_i, [5] uses a table indexed by the pair $\mathcal{P}[i] = \langle\ S_i[1, k]$, Arithmetic encoding of $S_i[k + 1, b]\ \rangle$. It is easy to observe that $\mathcal{P}[i]$ uniquely identifies S_i. A small technical point is that, if the encoding of S_i is longer than $\lfloor (1/2) \log n \rfloor$ bits, then block S_i is written explicitly without any compression. This table uses $O(2^{k \log |\Sigma| + (1/2) \log n} \log n) = O(|\Sigma|^k\ n^{1/2} \log n)$ bits.

Summing up the cost of the block's encodings and the space occupancy of the decoding table, we get the space bound of Eqn. (1), whenever $k = o(\log_{|\Sigma|} n)$.