IS TEXT COMPRESSION BY PREFIXES AND SUFFIXES PRACTICAL?[1]

A.S. Fraenkel[2,4,5] M. Mor[2] and Y. Perl[3,4]

ABSTRACT.

One approach to text compression is to replace high-frequency variable-length fragments of words by fixed-length codes pointing to a compression table containing these high-frequency fragments. It is shown that the problem of optimal fragment compression is NP-hard even if the fragments are restricted to prefixes and suffixes. This seems to be a simplest fragment compression problem which is NP-hard, since a polynomial algorithm for compressing by prefixes only (or suffixes only) has been found recently. Various compression heuristics based on using both prefixes and suffixes have been tested on large Hebrew and English texts. The best of these heuristics produce a net compression of some 37% for Hebrew and 45% for English using a prefix/suffix compression table of size 256.

---

1 This work was done within the Responsa Retrieval Project, developed initially at the Weizmann Institute of Science and Bar-Ilan University, now located at the Institute for Information Retrieval and Computational Linguistics (IRCOL), Bar-Ilan University, Ramat Gan, Israel. The work reported herein was done at the Weizmann Institute.

2 Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel 76100.

3 Department of Mathematics and Computer Science, Bar-Ilan University, Ramat Gan, Israel.

4 Partial affiliation with IRCOL.

5 Supported in part by a grant of Bank Leumi Le'Israel.

1. INTRODUCTION

One of the approaches to text compression is variable-to-fixed-length encoding. In this approach, high-frequency variable-length fragments of words in the text are replaced by fixed-length codes pointing to a compression table containing these high-frequency fragments. See e.g. Rubin [9] for heuristic algorithms for selecting the fragments and Cooper and Lynch [2] for a review of this approach. Radhakrishnan [8] considered the special case of fragments which are prefixes (or suffixes, but not both). See also Gotlieb et al. [4].

Recently, a dynamic programming polynomial algorithm for constructing an optimal all-prefix (or all-suffix) compression table for a text was given [1]. Can we expect a polynomial algorithm even for the general problem of optimal compression table construction containing arbitrary fragments? Storer and Szymanski [11] and Storer [10] addressed themselves to this question. They used a model where both the text and the compression fragments were represented by a long string. For this model they showed that finding an optimal compression string is an NP-hard problem. This suggests that the similar problem for a compression table may also be NP-hard. However, there may not be a simple transformation between the two models. It may thus be more profitable to give a separate proof for the table model problem.

In view of this situation we decided to consider the simultaneous prefix-suffix compression problem, that is, the question of constructing an optimal compression table containing affixes but no other fragments. By an affix we mean any fragment which is either a prefix or a suffix, as in Peterson [7]. An optimal affix compression table clearly yields a compression which is at least as good as that based on an all-prefix (or all-suffix) compression table, but the extent of the expected improvement is unknown. This problem seems to be the simplest unsolved case of the general problem of compression table construction using fragments, as well as the simplest generalization of the recently solved problem of compression by an all-prefix (or all-suffix) compression table mentioned above.

In Sect. 2 we show that even this problem is NP-hard, even for a ternary alphabet. In fact, the result holds even when every compressed word is compressed by only a single table-entry, and even when the compression table constitutes an affix code, that is, no table-entry is the prefix or suffix of any other table-entry. It is therefore natural to consider heuristics for the construction of an affix compression table. For this purpose we first constructed an all-prefix compression table and an all-suffix compression table separately, for English and Hebrew texts. In Sect. 3 we describe several heuristics for doing this, which seem computationally more efficient than the polynomial algorithm of [1]. One of them is similar to a heuristic considered in [8]. The others take into account the overlaps between dif-