Experiments with Persian Text Compression for Web

Farhad Oroumchian
University of Wollongong in Dubai
POBOX 20183, Dubai, UAE
+971-503585420
FarhadOroumchian@uowdubai.ac.ae

Ehsan Darrudi
Dept. of ECE,
University of Tehran
POBOX 14395-515, North Kargar Ave, Tehran, IRAN
+98-21-8020403-3381
e.darody@ece.ut.ac.ir

Fattane Taghiyareh
Dept. of ECE,
University of Tehran
POBOX 14395-515, North Kargar Ave, Tehran, IRAN
+98-21-8020403-3381
ftaghiyar@ut.ac.ir

Neeyaz Angoshtari
Computer Science Dept.
University of Southern California
Los Angeles, CA 90089, USA
neeyazan@usc.edu

ABSTRACT

The increasing importance of Unicode for text encoding implies a possible doubling of data storage space and data transmission time, with a corresponding need for data compression. The approach presented in this paper aims to reduce the storage and the transmission time for Persian text files in web-based applications and Internet. The basic idea here is to compute the most repetitive n-grams in the Persian text and replace them by a single character in the user-defined sections of the Unicode. The compression will be done on the server side once and the decompression process is eliminated completely. The rendering process in the browser will do the decompression. There is no need for any additional program or add-ins for decompression to be installed on the browser or client side. The user needs only to download the proper Unicode font once. A genetic algorithm is utilized to select the most appropriate n-grams. In the best case, we have achieved 52.26 % reduction of the file size. The method is general, and applies equally well to English and other languages.

Categories and Subject Descriptors

E.4 [**Data**]: Coding and information theory --- *Data compaction and compression.*

General Terms

Languages, Algorithms, Measurement

Keywords

N-gram Compression, Farsi, Unicode, Genetic Algorithm

1. INTRODUCTION

With the increase in the amount of non-English or Non-Latin text in the Internet, there is a growing need for the compression algorithms that can use the characteristics of these languages. Most of the algorithms that have been developed for text compression are adaptive dictionary-based compression algorithms [5]. In spite of their good compression ratios, they require considerable amount of computing resources at the decompression end.

Unicode [4] is an encoding system that provides a unique code for every character, used in all the major languages written today. The original goal is to use a single 16-bit encoding that provides code points for more than 65,000 characters. There are about 6,700 unused code points for future expansions.

WWW 2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-912-8/04/0005.

We have utilized an n-gram based algorithm and experimented with Persian language. By using a genetic algorithm (GA), we have selected the right combination of strings that should be condensed. We have assigned codes and font glyphs for these strings from the user-defined section of the Unicode. In this way, the reduced files are still directly viewable because newly assigned codes have glyphs associated with them. In fact this approach is more suitable for corporate intranets that share large volumes of text in their internal networks.

2. N-GRAM FREQUENCIES

These experiments are done on a Persian text collection that contains laws and regulations passed by Iran Parliament. In the first step the frequencies of 2-grams, 3-grams, 4-grams and 5-grams were calculated. In addition to Persian characters, numbers and punctuation marks and blank character (space) were also considered (totally 57 characters). We used a simple n-gram table to count frequency of each combination. We didn't take in to account sequences larger than 5-grams for time considerations.

In general, replacing 2-grams produces the most reduction, and replacing 5-grams results in the least reduction. Figure 1 shows the real effect of replacing n-grams using 500 most frequent n-grams in each group. The sample file size was 46 MB of text encoded in Unicode. As it can be seen in the figure, shorter n-grams lend themselves to compression better than longer n-grams. For example to reduce the file size to 35MB, we only need 73 2-grams. However, we need 144 3-grams or 260 4-grams or 440 5-grams in order to achieve the same level of reduction.

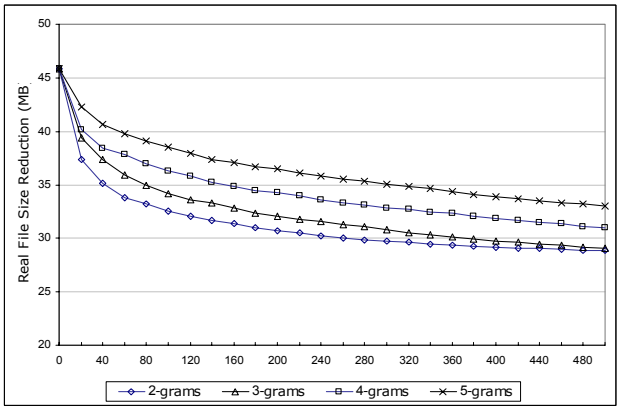

Fig. 1- Accumulative reduction of file, after practically replacing n-gram terms with new codes

3. N-GRAM SELECTION BY GA

Finding the best set of n-grams for substitution (best dictionary) is NP-hard [3], and many heuristics have been proposed [5]. The novelty of the approach presented here is in using a Genetic Algorithm (GA) [2] to select best combination of n-grams to achieve optimal (or near optimal) compression ratio with the minimum number of n-grams for replacement.

As depicted in figure 1, small numbers of replacements can produce very good results. Therefore there is no need to examine all the n-grams of the groups in order to find the optimal solution. In this case we used 500 best n-grams of each group. That is, the biggest individual combination that would be generated and computed by genetic algorithm was 500-500-500-500 (5, 4, 3 and 2-grams, respectively). It is obvious that without any restriction, the best solution is the one that uses maximum number of n-grams because it causes most size reduction on the sample file. So, we didn't use file size reduction as fitness value. We needed a solution to produce good reduction with reasonable number of n-grams. We used the following formula (1) as our fitness function.

$$fitness = \frac{reduction}{size + \beta} \qquad (1)$$

Where, for each combination of 5, 4, 3 and 2-grams, *reduction* is the saving in the storage achieved by replacing its n-grams with new codes; *size* is the proportional size of the combination. It is computed by dividing the size of the combination (total number of 5, 4, 3 and 2-grams) by the maximum possible size of combinations (4×500 = 2000). Inverse relation between *fitness* and *size* restrains evolution from choosing very large combinations. Although, larger combinations may cause greater compression ratios but they need also more efforts to define new codes and create appropriate font glyphs. β is a constant (for each experiment) that controls genetic algorithm convergence. Indeed, β value determines how much the size of the solution is important for us. Small value of β implies that we need a solution with minimum number of n-grams but still reasonable compression ratio. In the other hand, big β weakens the influence of *size* and lets the evolution to choose larger combinations, thus more compression. We ran our genetic algorithm for different values of β. Table 1 shows best combinations achieved at the end of evolutions and their compression ratios.

Table 1- Final GA solutions for different β values with achieved compression ratios

β Value	Best Solution					Comp. Ratio (%)
	5-grams	4-grams	3-grams	2-grams	total	
0	0	0	0	2	2	4.30
0.25	9	6	3	17	35	24.35
0.5	23	6	23	60	112	34.25
1.5	45	16	43	181	285	42.35
2.5	74	26	50	260	410	45.44
3.5	132	16	90	287	525	47.42
4.5	164	26	173	385	748	50.06
5.5	289	62	188	489	1028	52.26

As it is apparent from table 1, there is a trade-off between the number of required new codes and compression ratio. β values 0.25, 0.5, 1.5 seems to be appropriate for most applications. However, higher β values can be considered if higher compression is required.

4. FONT CREATION

After choosing the correct solution with appropriate number of 5, 4, 3 and 2-grams from table 1, the new codes should be allocated in the user-defined section of Unicode. This section currently, does not contain any character assignments and covers E000-F8FF (6400 code points). Finally, new glyphs should be created and added to the current font file that supports Unicode [1]. In Persian (and Arabic), each letter may have different presentation forms according to its location in the word. For example, there are four forms for "غ": an initial "غـ", a medial "ـغـ", a final"ـغ" and an isolated "غ". Thus, for some n-grams it is required to create glyphs for all presentation forms. Most of the top n-grams begin or end with blank (space) character, so the number of excess presentation glyphs shouldn't be high.

5. CONCLUSION AND FUTURE WORK

We have presented the idea of compressing Persian language text by replacing carefully selected 2, 3, 4, and 5-grams and placing these n-grams in the user defined section of the Unicode to avoid decompression. After calculating the frequency of each n-gram, a genetic algorithm was employed to select the right n-grams of each group in order to get an optimal file reduction with the least possible number of n-grams. The highest percentage of file reduction was reached was 52.26 percent.

The drawback to this method could be its dependence on the text collection. Our experiments are based on a text collection that is about Iranian laws and regulations which implies that the list of high frequency n-grams (especially larger ones) may be biased. Some 5-grams such as "_ماده" (article), "_نامه" (letter), "قانون" (law), "وزارت" (ministry) appeared as good candidates for replacement while they are not very common in Persian general documents. However, this could also present an opportunity. For example different corporations can use different n-grams for compression of their internal text that maximizes their unique collection.

The approach presented here doesn't presuppose anything about the structures of the language. Thus it can be used for other languages with minimal modifications. We aim to test our method with languages that have structural similarities with Persian in terms of the number of the alphabets and the repetition of linguistic information inherent in the structures of the languages.

6. REFERENCES

[1] Bigelow, C. and Holmes, K. The design of a Unicode font, Electronic Publishing, Vol. 6, No. 3, 1993, pp. 289–305.

[2] Holland, J.H. Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.

[3] Storer, J.A. and Szymanski, T.G. Data compression via textual substitution, Journal of the ACM, Vol. 29, No. 4, October 1982, pp. 928-951.

[4] The Unicode Consortium, The Unicode Standard, Version 3.0, Addison-Wesley Press, Massachusetts, 2000.

[5] White, H.E. Printed English compression by dictionary encoding, Proceedings of the IEEE, Vol. 55, No 3, March 1967, pp 390-396.

[6] Ziv, J., and A. Lempel. A universal algorithm for sequential data compression, IEEE Transactions on Information Theory, 23 (1977) 337-343.