A Survey of Grammar based Compression

Easwaran R

IISc-CSA-TR-2003-5

http://archive.csa.iisc.ernet.in/TR/2003/5/

Computer Science and Automation
Indian Institute of Science, India

June 2003

A Survey on Grammar based compression

Easwaran R.
Department of Computer Science and Automation
Indian Institute of Science
Bangalore 560012

Abstract

Recently, text compression has been viewed as the problem of getting a compact grammar for the given string. Decompression involves getting the string from the grammar. Most such approaches focus on getting a grammar which produces exactly one string, the string which is compressed. We survey a set of papers on this subject. Four of them analyze the problem from an information theory point of view, by computing the redundancy of these codes with other classes of codes. Two other papers looks at whether the grammar based schemes exploit the full power of the grammar model

Introduction

Grammar based compression involves obtaining a grammar for a given string, and decompression is achieved by deriving the string from the grammar. Since grammars are well studied, it is possible that several string operations can be directly done in the compressed text. Though popular compression schemes like LZ78[1] can be viewed as producing a grammar as the compressed text, only recent works use the term *grammar based compression* and view the schemes in this perspective. The survey has six sections, each of them summarizing the work of a particular work in this field. The final section concisely summarizes the work in these papers.

Grammar-Based Codes : A new class of Universal Lossless Source Codes

The paper by Kieffer and Yang [2] describes the notion of grammar-based codes, wherein the string to be compressed is represented by a Context Free Grammar (CFG), which generates only the given string. The idea of using a grammar based compression by itself is not new and previous works are cited by this paper. But the authors claim that this is the first paper which examines the problem from the information theoretic point of view and describes a systematic way of generating such codes.

After describing the well known terminologies of CFG, the authors define a subclass of CFG called Admissible Grammars. Admissible grammars are cycle-free, deterministic CFGs, which doesn't have any useless symbols and which are non-empty. A string is compressed by obtaining an admissible grammar, which represents the string.

Then the paper talks of Grammar Transforms, which is the process of uniquely mapping a given string to an admissible grammar. Two types of grammar transforms, Asymptotically Compact (AC) Grammar Transforms and Irreducible Grammar Transforms, are described.

Asymptotically compact transforms are those that transform a string x into an admissible grammar G_x satisfying the condition $lim_{n \to \inf} \max_{x \in \mathcal{A}^n} \frac{|G_x|}{|x|} = 0$, where $\mathcal{A}$ is the alphabet set and n is the length of the string x. The paper shows that the Ziv Lempel (LZ78) transform and the bisection grammar are both asymptotically compact.

Irreducible Grammars are the grammars which cannot be 'reduced' by applying a set of reduction rules. These reduction rules transform a grammar into an equivalent grammar producing the same language, but with a reduction in the size of the grammar. There are 5 reduction rules given in the paper. They are

- If a string of symbols of length more than 2 appear at two or more non-overlapping places in a production rule, replace that string with a non-terminal
- If a string of symbols of length more than 2 appear in two different production rules, replace the string with a non-terminal
- If a non-terminal appears only once in the grammar, replace the non-terminal with its expansion
- If a non-terminal is redundant, remove the non-terminal
- If two non-terminals have the same derivation, replace all occurrences of one by the other non-terminal

Thus an irreducible grammar can be thought of as a locally optimal grammar, with respect to the set of five reduction rules given in the paper.

Finally the paper proves certain theorems which gives the asymptotic performance of the two grammar transforms proposed in the paper. The asymptotic performance of the codes is measured for k^{th} order finite state sources with alphabet $\mathcal{A}$ denoted by $\Lambda_{fs}^{k}(\mathcal{A})$. A notion of maximal point-wise redundancy is used to quantify the performance of these transforms. By proving that the maximal-point-wise redundancy tends to 0 as the string length nears infinity, they prove that the codes are asymptotically optimal. The theorems are

Theorem 1 *Let ϕ be a grammar-based code from the class of AC codes, and $x \to G_x$ be the underlying grammar transform. Let $\{\nu_n\}$ be a sequence of positive numbers converging to zero such that*

$$\max_{x \in \mathcal{A}^n} \frac{|G_x|}{|x|} = O(\nu_n)$$

Then, for every positive integer k

$$Red_n(\phi, \Lambda^k_{fs}(\mathcal{A})) = O(\gamma(\nu_n))$$

where γ is the function defined by

$$\gamma(x) \triangleq x \log(1/x), x > 0$$

Theorem 2 *The class of irreducible grammar codes $\mathcal{C}_{irr}(\mathcal{A}$ is a subset of the class of AC codes. Furthermore, for every positive integer k*

$$\max_{\phi \in \mathcal{C}_{irr}(\mathcal{A})} Red_n(\phi, \Lambda^k_{fs}(\mathcal{A})) = O(\frac{\log \log n}{\log n})$$

Algorithms for irreducible grammar transforms

This second paper by the same authors [3] is based on their work on irreducible grammar transforms. The authors combine arithmetic coding with grammar transforms to achieve better compression. They discuss three discuss schemes which create a string into an irreducible grammar. The first scheme is called a hierarchical scheme, the second is an online scheme and the third scheme improves upon the second using some special context information.

The paper introduces a *greedy sequential transform* which forms the basis of all the three algorithms. The proposed transform parses the input string into a sequence of non-overlapping substrings and incrementally builds an irreducible grammar for the concatenation of these substrings. In other words, given a parsing of the string x into substrings $s_1, s_2, \ldots s_t$, given an irreducible grammar transform G_i for $s_1 s_2 \ldots s_i$, and the substring s_{i+1}, the greedy transforms obtains G_{i+1}, the irreducible grammar for $s_1 s_2 \ldots s_i s_i + 1$. The substring s_{i+1} is the longest substring in the unprocessed portion of the given string x which can be represented by a symbol of the irreducible grammar G_i.

Using the irreducible transform, three different algorithms are discussed. The **Hierarchical algorithm** constructs the irreducible grammar for the given string using the above mentioned transform. Once the grammar for the whole string is obtained, the symbols of the grammar are encoded using a zero-order arithmetic coder. The **Sequential algorithm** is an online algorithm which arithmetic codes the partial grammar obtained at any intermediate stage. A dynamic alphabet is used in the arithmetic coding process as the alphabet set (to be specific, the set of Non-terminals) is increasing as the grammar gets constructed. When a symbol is seen, the decoder updates the grammar so that it is in sync with the encoder. The third algorithm proposed is an improvement over the sequential algorithm. Since both the encoder and the decoder know the structure of the irreducible grammar G_i when the next token is being encoded, this grammar G_i can be used as a context. Essentially, the algorithm uses the fact only certain substrings can result in a pair of non-overlapping substrings, thereby generating a new non-terminal and so if such a reduction takes place, then the substring can be encoded with a higher probability.

Then the paper analyzes the three algorithms. The analysis of the sequential and the improved sequential algorithms are similar to that of the hierarchical algorithm. The analysis uses combinatorial arguments to bound the no. of grammar symbols. They derive the worst case redundancy of the hierarchical algorithm against the *k-context empirical entropy*. The worst case redundancy is the difference bits per symbol between the hierarchical algorithm and the k-context empirical entropy for the best probability model. This value is given by

$$R_{n,k} \leq d_k \frac{\log \log n}{\log n}$$

, where the constant d_k depends only on the alphabet size and k. Similar results are proved for the sequential and the improved sequential algorithms. Then the authors show simulation results to prove the efficiency of the compression schemes discussed. For binary strings of different lengths, they compare the bits per symbol of their schemes, *UNIX compress* and *gzip*. The strings are generated by three different sources - memoryless binary source, first order Markov sources and second order Markov sources. The results show that the grammar based schemes perform better than the other schemes. *But the performance of zeroth order arithmetic coding is better than the grammar based schemes for the sources and string lengths mentioned and the paper has not compared its performance with plain arithmetic coding*

Multilevel Pattern Matching algorithm

The third in the series of papers[4] describes a new universal grammar based compression scheme called Multilevel Pattern Matching (MPM). The idea of the scheme is to partition the given string into a fixed number of substrings, then partition the substrings and so on till the substrings' size becomes 1. The substrings are given a non-terminal. If there are multiple substrings of a particular length, then all of them are represented by the same non-terminal, thereby achieving compression. The paper explores how such a partitioning can be efficiently made and then analyzes the performance of the scheme asymptotically.

The MPM scheme has two parameters, r and I. The restrictions on these parameters are, $r \geq 2$ and $r^I \geq n$, where n is the length of the string to be compressed. The given string is partitioned into non-overlapping, contiguous substrings of length r^I and a residual portion. Each of the substrings of length r^I are given Non-Terminals and are again partitioned to r different substrings and so on till the size of the substring becomes 1. The residual string is split when its size equals or exceeds the substring at some stage. Since substrings are represented by Non-Terminals, repeated patterns of different lengths are represented by the same NT, thereby achieving compression. After getting the NTs (or tokens), the tokens are encoded using an arithmetic coder.

The paper then attempts to analyze the performance of the MPM code with the aim of finding the best values for the parameter I to get a good compression. The first result gives a bound on the no. of tokens generated in the partitioning process. The size of tokens is upper bounded by

$$C(r, I, |\Lambda|, n) = 2nr^{-1} + 2Ir + 2r|\Lambda|^r + 4r|\Lambda|^5 + 8\log|\Lambda|r^2\frac{n}{\log n}$$

, where $|\Lambda|$ is the cardinality of the alphabet set. Using this result, it is shown that the size of the compressed string is close to the s^{th} order empirical entropy. The bound is given by

$$L(x|r, I) \leq H^s(x) + 2\lceil\log(n+1)\rceil + 2r|\Lambda|^{r+1} + (2 + \log s)C(r, I, |\Lambda|)$$

, where x is the string to be compressed and I and r are parameters of the MPM code.

Finally the paper derives the redundancy of the codes. The redundancy of the code is calculated relative to a class of codes (arithmetic coder with a finite context) and also relative to a class of finite state sources. As in other grammar based codes, the redundancy of MPM also approaches 0 asymptotically. But the rate at which it drops to 0 depends on the value of I chosen as the parameter. A main result of this paper is in showing that the best value of I is given by $C\log\log n \leq I \leq \log_r n$, where C is some positive constant. The paper also contains some experimental results on the performance of this scheme on image compression.

Multilevel pattern matching with side information

This paper [5]is on improving the performance of the MPM scheme using context information. The context is known to both the encoder and the decoder. The paper doesn't talk about how to choose the context. It explains how to improve the MPM scheme, given some context information. This scheme, called the Conditional MPM (CMPM) scheme has linear time and space complexity and it asymptotically reaches the conditional entropy rate for stationary ergodic sources. The context information is also given as a string, of the same length as the string to be compressed, and the string uses a different context alphabet.

The paper reviews the MPM scheme without the conditional information. It then proposes a *bottom-to-top* version of the MPM scheme, where the non-terminals at the bottom level (i.e non-terminals whose expansion length is lesser) are created before the non-terminals at the top level. They have proposed an algorithm for this bottom-to-top method. Given the tokens (non -terminals) T_i at some level i, the algorithm performs two steps. The first step is to create the tokens T_{i+1} at level $i+1$. This step involves clubbing r tokens at the level i and associating a token at level $i+1$ for every distinct set of r tokens. The next step is to update the token list T_i. This stage is needed because tokens which can be derived from the same token at the $i+1$ level, need not be repeated at two different places. An efficient implementation of such a method is discussed. Both the space and the time complexities of this algorithm are linear in both the alphabet size ($|\mathcal{A}|$) and the length of the string (n).

The paper then introduces the idea of Conditional MPM (CMPM). Here, along with the given string x of length n, another string y, also of length n, is given as input. The string y is from an alphabet set $\mathcal{A}_y$, the alphabet set of side information. The CMPM works in a similar way as MPM, and the bottom-to-top partitioning process is carried out both for the original string and the string of conditional symbols. While arithmetic coding the grammar symbols of the given string, the grammar symbols of the side information string is used as the context. Thus the probabilities of the grammar symbols increases, resulting in a better compression rate. Asymptotically, the compressed string's reaches the conditional entropy of x given y. They also show that the worst case redundancy of the scheme against a finite state arithmetic coder with side information is $O(\frac{1}{\log n})$. Finally, simulation results of this code for different models of sources are given.

Efficiency of existing grammar based schemes in exploiting the grammar based model

This paper by Lehman and Shelat [6] differs from the previous ones that it looks at how existing grammar based strings exploit the underlying grammar model for finite length strings rather than looking at the asymptotic case. Since obtaining the optimal grammar for a string is a hard problem, the paper first gives certain results with regard to the hardness of getting an approximate solution. Then the paper proceeds to analyze the performance of four existing grammar based compression schemes by giving upper and lower bounds on the approximation ratio of these grammars with respect to the optimal grammar. The main contribution of this work is in showing that the none of the existing grammar based schemes exploit the performance of the grammar model by showing that their approximation ratios are farther away from the hardness results shown earlier.

The first result of this paper is in showing that it is impossible to approximate the size of the smallest grammar to within a constant factor (8569/8568) of the optimal grammar unless P=NP. To prove this, they have reduced a restricted form of vertex cover problem, which has a known constant factor approximation ratio, to the grammar problem. Next they proceed to show that this constant factor approximation is unlikely to be realized by showing that to better the ratio of $o(\log n/\log\log n)$ implies progress on a well studied problem. This problem is that of finding the shortest addition chain containing a sequence of p integers , $k_1, k_2 \dots k_p$. Even when the problem is given in unary it has only a o(log n/ log log n) approximation ratio

After showing the inapproximability results, they consider four existing schemes, which can be thought of as grammar based schemes, and derive their approximation ratios. Such an analysis proves to be difficult and they are able

LZW	$O((n/\log n)^(2/3))$	$\Omega(n^(2/3)/\log n)$
BISECTION	$O((n/\log n)^(1/2))$	$\Omega(n^(1/2)/\log n)$
SEQUENTIAL	$O((n/\log n)^(3/4))$	$\Omega(n^(1/3))$
GREEDY	$O((n/\log n)^(2/3))$	$> 1.37\ldots$

Table 1: Approximation ratios of grammar based schemes

to show only the range in which their approximation ratios lie by showing an upper and a lower bound for each of their schemes. The schemes they consider are LZ78, BISECTION (a particular case of MPM), SEQUENTIAL (a modification of SEQUITUR) and a greedy scheme. The approximation ratios of these schemes are tabulated in Table 1

The best proven approximate ratio is that of BISECTION. Only the greedy algorithm has a potential to get a constant factor ratio but its proven upper bound is worse than that of BISECTION. Thus the main contribution of this paper is that although many of the grammar based schemes are asymptotically optimal for certain classes of sources, for finite length strings they don't fully exploit the underlying grammar model. Another contribution is in analyzing the approximation ratios for hierarchical structures like grammars.

A new grammar-based compression algorithm with a better approximation ratio

This paper [7] develops a grammar-based compression scheme which gives a better approximation ratio than the schemes analyzed in the previous paper. The approximation ratio of this scheme is $O(\log(n/g^*))$, where n is the size of the string to be compressed and g^* is the size of the optimal grammar. The paper also looks at other grammar models like CFG with information on derivation tree and NDFAs and gives inapproximability results for these models.

The proposed grammar based scheme is based on the LZ77 scheme. The LZ77 scheme cannot be looked as a grammar based scheme. But the authors show that the compression of this scheme is a lower bound on the size of the optimal grammar. *However, while making this claim the authors consider the LZ77 scheme where any reference to an existing substring ...* This is shown by converting the optimal grammar to LZ77 form, whose size is of the same order of the optimal grammar.

The proposed algorithm converts a string into a CFG (an *admissible CFG*) which is called a *balanced-binary grammar.* A binary grammar is one in which the right hand side has exactly two symbols (terminals and non-terminals). A balanced binary grammar is one in which the ratio expansions of the right-hand sides is bounded above and below by a certain fraction. In other words, if $A \to \beta\gamma$ is a production in the grammar, then $\frac{\delta}{1-\delta} \le \frac{|\beta|}{|\gamma|} \le \frac{1-\delta}{\delta}$, where $|\delta|$ denotes the length of the *expansion* of δ. The grammar produced by the algorithm maintains the invariant of being a balanced-binary grammar.

The algorithm converts the substrings generated by the concatenation of successive LZ77 tuples into a balanced-binary grammar. After adding every LZ77 tuple, the resulting grammar is balanced. For this purpose, they employ three sub-routines. The first one is called **ADD-PAIR** which takes two Non-Terminals, whose right-hand sides are balanced, and creates a new Non-terminal whose expansion is the concatenation of these two non-terminals. The next routine is called **Add-SEQUENCE** which takes a sequence of symbols, the non-terminals in them are balanced-binary, to produce a single non-terminal whose expansion is the concatenation of the expansion of the sequences. The resulting grammar is balanced-binary. The third subroutine called **ADD-SUBSTRING** takes a string ω, which is the expansion of a non-terminal in a balanced binary grammar, and creates a non-terminal whose expansion is a given string ω_1, which is a substring of ω. As in the previous operations, here also the invariant is maintained. Given these routines it is easier to convert LZ77 tuples into a balanced-binary grammar. Given a grammar which represents the concatenation of a sequence of tuples, the string representing the next tuple can be added using the **ADD-SUBSTRING**. The size of the resulting grammar is $O(g^* \log(\frac{n}{g^*}))$. In order for these subroutines to work in the manner described (which ensures this approximation ratio), the value of the parameter δ described in the balanced-binary grammar should satisfy the condition that $0 < \alpha \leq 1 - \frac{1}{2}\sqrt{2}$.

The paper also talks about other means of measuring the complexity of a string using related models. They consider the NFA model, where the given string is represented by a NFA. Since an NFA can produce strings other than the given string, an *advice-string* is provided to choose the correct string. The size of the compressed string is the size of the NFA and the size of representing the advice. It is shown that, using this model, the given string can be approximated within a $O(\log^2 n)$ of the optimal grammar. Then they consider a model where there is a CFG which can derive many strings (including the string to be compressed) and an advice string which chooses the correct string. It has been shown that this model has the same complexity as the CFG which produce only one string (within a constant factor). Finally they consider another model, which has a CFG and an *edit-string* which edits (insert, delete or replacement of a character) the grammar to derive the given string. Even this model can approximate the optimal grammar only to a $O(\log n)$ factor.

Thus the main contribution of the paper is to give an algorithm for grammar based compression which guarantees a better approximation ration than the existing algorithms. But the paper does not show how effective this scheme in practice as it does not show an implementation of the scheme and its comparison with other schemes. The paper also discusses related models and concludes that none of them are more powerful than the simple model of generating a CFG which produces only the given string.

Summary

The first four papers analyze the notion of grammar-based codes from an information theory point of view. They show that the grammar-based codes they consider asymptotically reach the entropy of the string for certain class of sources. One of the potential advantages of the grammar-based codes is that they can be very amenable to searching without decompressing. But since most of the proposed schemes use arithmetic coding to encode the grammar symbols, it is not very clear how these codes can be leveraged to do searching without compression. Since arithmetic coding is used, they also talk about using side information to give context for the arithmetic coder.

The last two papers look at the grammar based compression from the point of view of exploiting the underlying grammar model. These schemes are also reach the entropy asymptotically. But since the new scheme proposed in the second paper does not talk about implementation aspects, there is no way to gauge its performance. The authors themselves state this fact. Efficient implementation of the scheme can be looked at to see its practicality. Conversely, better analysis of the existing schemes may prove that those codes also show a better proven approximation guarantee than what is shown in the first paper.

References

[1] J. Ziv and A. Lempel. "Compression of Individual Sequences via Variable-Rate Coding." *IEEE Transactions on Information Theory*, vol. 24 (1978)

[2] J.C.Kieffer and E-H. Yang, "Grammar based codes : A new class of universal lossless source codes", *IEEE Trans. on Information Theory*, May 2000

[3] E-H Yang and J.C. Kieffer, "Efficient universal lossless algorithms based on a greedy sequential grammar transform -Part one : Without context models", *IEEE Trans. on Information theory*, May 2000

[4] J.C. Kieffer and E-H Yang, "Universal lossless compression via Multilevel Pattern Matching", *IEEE Trans. on Information theory*, July 2000

[5] E.-H. Yang, A. Kaltchenko, and J. C. Kieffer, "Universal lossless data compression with side information by using a conditional MPM grammar transform ", *IEEE Trans. on Information theory*, September 2001

[6] Eric Lehman and Abhi Shelat, "Approximation Algorithms for Grammar-Based Compression" *SODA*, 2002

[7] Moses Charikar *et. al.*, "Approximating the Smallest Grammar: Kolmogorov Complexity in Natural Models", STOC 2002