Data Compression Using Grammar-Based Codes

Michal Vašinek

Faculty of Electrical Engineering and Computer Science
Department of Computer Science
VŠB - Technical University of Ostrava

February 1, 2019

Grammar-Based Codes

The main objective of a grammar-based data compression algorithms is to find the smallest grammar that replaces raw representation of input message.

Example

Let $m = abcdabcdab$. The CFG representation of m:

$$
\begin{aligned}
S &\rightarrow 220 \\
0 &\rightarrow ab \ (m_1 = 0cd0cd0) \\
1 &\rightarrow 0c \ (m_2 = 1d1d0) \\
2 &\rightarrow 1d
\end{aligned}
$$

- Competitive compression ratio, fast decompression, compressed pattern matching.

The Smallest Grammar Problem

The smallest grammar problem is NP-hard (Charikar 2005)

Find a grammar so that the sum of the number of symbols on the right side of production rules is minimal.

- Usually not possible to find an optimal solution $\Rightarrow$ use of heuristics.
- Framework of Re-Pair algorithm
  1. Find a pair of symbols ab so that the frequency $f(ab)$ is maximal.
  2. Replace all occurences of ab for some yet unused symbol γ.
  3. Repeat Steps 1. and 2. until for all pairs p: $f(p) < 2$.

Objectives

Main idea

Application of a production rule leads to the change of the message length and zero-order entropy. Can we predict this change?

Main objective

Quantify how much the resulting compressed size will change before the production rule is applied.

- Identify quantities that are modified by grammar transformation.
- Describe how these quantities influence subsequent application of statistical coder.
- Propose algorithms (strategies) exploiting this knowledge.

Measuring size of strings

Two approaches how to measure the size of the message m:

- The number of symbols in m: $|m|$
- Entropic size of messages:

$$\begin{aligned} |m|^H &= |m| H_0(X) \\ &= -|m| \sum_{x \in X} p(x) \log p(x) \end{aligned} \quad (1)$$

MinEnt Strategy: The largest entropic size reduction first. (Vasinek 2017)

$$\Delta |m|^H = |m_0|^H - |m_1|^H \quad (2)$$

Select and replace repeated string s so that $\Delta |m|^H$ is maximal.

Entropic Size Change

Computation of $\Delta|m|^H$ for $ab \to \gamma$ replacement (Vasinek 2017)

$$\Delta|m|^H = \left[|m_0| - \sum_{x\in\Sigma\setminus\Sigma^T} f_0(x)\right]\log c_1 + \sum_{x\in\Sigma^T} f_0(x)\log c_2(x) + \Delta f(x)\log c_2(x) + \Delta f(x)\log p_0(x) + \Delta f(\gamma)[\log \Delta f(\gamma) - \log(|m_0| + \Delta m)] \tag{3}$$

- Symbols whose frequency do not change: $\Sigma \setminus \{a, b\}$.
- Symbols participating in production rule, i.e. their frequency changes, but their initial frequency is non-zero: $\Sigma_T = \{a, b\}$.
- Symbols introduced into the message: $\{\gamma\}$.

Paradoxes of Recursive Pairing

Example - Reduction Paradox

Let $m_0 = abbaabacbd$. Only pair **ab** has frequency greater than 1:

$$m_1 \rightarrow 0ba0acbd$$
$$0 \rightarrow ab$$

- $|m_1| < |m_0|$
- $\Delta|m|^H = |m_0|^H - |m_1|^H = -0.78$ bits.

- Reduction paradox: decreasing the length of the message not necessarily leads to decrease of the entropic size!
- Expansion paradox: direct consequence of the reduction paradox.

Comparison of $\Delta|m|^H$ evolution - Re-Pair and MinEnt

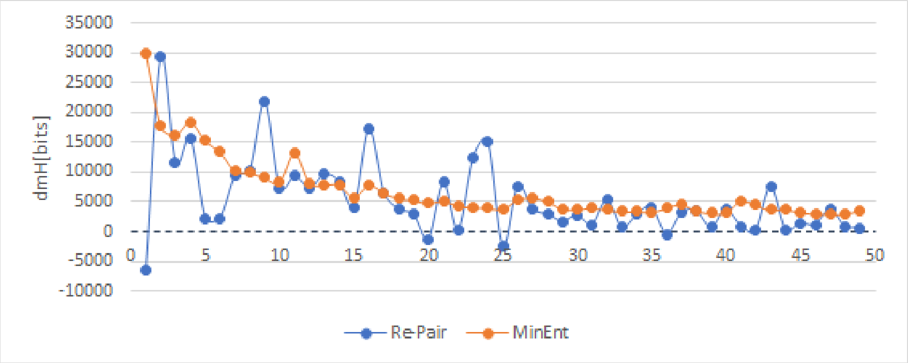

Figure: calgary/book1

The Smallest Grammar Problem Revisited

- Standard grammar encoding:
  - Right side of start nonterminal is encoded by zero order statistical coder.
  - The set of remaining production rules is encoded by differential codes: $\{ab, ac\} \Rightarrow_{enc} a, b, \gamma_0(a - a), \gamma_1(c - b) \rightarrow a, b, \gamma_0(0), \gamma_1(1)$.
- The smallest grammar problem is not necessarily equal to **the smallest compressed grammar problem.**

	$\|m\|$	$\Delta\|m\|^H$
Reduction paradox is present	true	false
Allows $f(p) = 1$ replacements	false	true

Entropic Size Change - Algorithms

- Context Transformations: verification of $\Delta|m|^H > 0$:
  - Context Transformations (CT, Vasinek 2014) - replacement rule $\alpha\beta \rightarrow \alpha\gamma$, if $p_0(\alpha\gamma) = 0$.
  - Generalized CT (GCT, Vasinek 2015) - any replacement rule $\alpha\beta \leftrightarrow \alpha\gamma$.
  - Higher-Order CT (HOCT, Vasinek 2016) - replacement rule $w\beta \leftrightarrow w\gamma$, for $|w| \geq 1$.
- MinEnt strategy:
  - MinEnt algorithm (Vasinek 2017) - replacement rule $\alpha\beta \rightarrow \gamma$.
  - Context Dependent Re-Pair (CD-Re-Pair) - replacement of patterns with $f(p) = 1$.

Comparison of Re-Pair and MinEnt - bible.txt

- Re-Pair - the most frequent pair of symbols first.
- MinEnt - the pair of symbols with the highest $\Delta|m|^H$ first.

Algorithm	$\|\Sigma_n\|$	H_0	$\|m_n\|$	bpB	$\|G\|$
Re-Pair	81,246	14.88	386,517	1.85	548,883
MinEnt	84,880	**14.72**	**372,663**	**1.80**	**542,297**

Comparison of Re-Pair and MinEnt - paper1

- paper1 is relatively small file (52kB).
- Re-Pair achieves smaller bpB ratio even through H_0 and $|m|$ are larger.
- The encoding of the set of production rules is prevailing factor.

Algorithm	$\lvert\Sigma_n\rvert$	H_0	$\lvert m_n\rvert$	bpB	$\lvert G\rvert$
Re-Pair	3,650	10.76	8,792	**2.67**	**15,902**
MinEnt	4,231	**10.53**	**8,728**	2.77	17,000

Comparison of Re-Pair and MinEnt - E.coli

- E.Coli genome exhibits 0-order Markov I.I.D. source like behaviour, i.e. $H_0 \approx H_1 \approx \ldots H_k$.
- Re-Pair will try to compress but finishes with additional bits needed for storage of production rules.
- MinEnt won't produce any production rule.

Algorithm	$\lvert\Sigma_n\rvert$	H_0	$\lvert m_n\rvert$	bpB	$\lvert G\rvert$
Re-Pair	67,040	13.72	**651,012**	2.31	**785,084**
MinEnt	4	1.99	4,638,690	**2**	**4,638,690**

Summary

- Identification of main quantities responsible for the change of entropic size.
- Formulation of equation for computation of $\Delta|m|^H$.
- Proposal of algorithms GCT, HOCT, MinEnt and CD-Re-Pair selecting grammar production rules based on $\Delta|m|^H$.
- Proposal of other grammar-based compression algorithms: DBC, DBCR .

Future Research

- Derivation of simple rules so that we don't have to compute $\Delta|m|^H$ directly.
- Selection of rules should not only count for $\Delta|m|^H$, but the resulting measure should also take into account storage of production rules.
- Utilization of pattern p replacement with $f(p) = 1$.
- Entropy of the set of production rules. Upper bounded by $d \log d + 0.557d$ (Tabei et al. 2016).

References

- Vasinek, M., Platos, J., Entropy Reduction Using Context Transformations, *Data Compression Conference (DCC)*, 2014.
- Vasinek, M., Platos, J., Generalized Context Transformations - Enhanced Entropy Reduction, *DCC*, 2015.
- Vasinek, M., Platos, J., Parallel Approach to Context Transformations, *Dateso Workshop*, 2015.
- Vasinek, M., Platos, J., Higher Order Context Transformations, *arXiv preprint arXiv:1701.01326*, 2016.
- Vasinek, M., Platos, J., Delimiter-Based Grammar Compression, *ISCAMI*, 2016.
- Vasinek, M., Platos, J., Prediction and evaluation of zero order entropy changes in grammar-based codes, *Entropy*, 19(5), 2017.

Thank you for your attention!
Questions and/or discussion?

Context Transformations

Question: $H_0(Y) = H_k(X)$?

- $Y_0 Y_1 Y_2 = t(X_1, X_2, \dots)$, t corresponds to CT.
- Assuming k-order Markov process and that $H_0(Y) = H_k(X)$.
- Assume that $p_Y(y|x) \neq p_Y(y) \Rightarrow$ we can compress Y below H_0 using conditional probability, **contradiction** with $H_0(Y) = H_k(X)$.
- Conclusion is that $p_Y(y|x) = p_Y(y)$. All conditional distributions must be equal to distribution of symbols.
- It is possible to achieve $H_k(X)$ if there is a sequence of context transformations producing equal distributions.

Context Transformations cont.

Example $H_0(Y) = H_1(X)$

$p(y\|x)$	a	b
a	0	1
b	1	0

- The sequence of letters: $s = abababab\dots$.
- $H_1(X) = 0$ but $H_0(X) = 1$.
- $GCT(ab \leftrightarrow aa, s) = aaaaaaaa\dots \Rightarrow H_0(Y) = H_1(X)$.

Context Transformations cont.

Example $H_0(Y) \neq H_1(X)$

$p(x,y)$	a	b	c
a	$p(aa)$	$p(ab)$	$p(ac)$
b	$p(ba)$	$p(bb)$	$p(bc)$
c	$p(ca)$	$p(cb)$	0

- Assume $p(a) > p(b) > p(c)$ and $p(cb) > p(ca) \Rightarrow GCT(cb \leftrightarrow ca)$.
- If $H_0(Y) = H_k(X)$ then $p_Y(c|c) = p_Y(c)$, but $p_Y(c) \neq 0$ and $p_Y(c|c) = 0$.

Context Transformations cont.

Example $H_0(Y) \neq H_1(X)$

$p(x,y)$	a	b	c
a	$p(aa)$	$p(ab)$	$p(ac)$
b	$p(ba)$	$p(bb)$	$p(bc)$
c	$p(ca)$	$p(cb)$	$p(cc)$

- $p_Y(a|a) = \frac{p_Y(aa)}{p_Y(a)} = p_Y(a)$
- $p_Y(aa) = p_X(aa) - p_X(caa) + p_X(cba)$
- $p_Y(a) = p_X(a) + p_X(cb) - p_X(ca)$
- There are σ^{k+1} such equations for k order processes..

Context Transformations - No Fixed Point

- Let $m = 1111$
- Apply $GCT_{\rightarrow}(11 \leftrightarrow 01)$

$$
\begin{array}{c}
1111 \\
0001 \\
0011 \\
0101 \\
1111
\end{array}
$$

Context Transformations in Comparison with MinEnt

- Context Transformations (CT) preserves the size of the alphabet and the message length $\Rightarrow$ formula for $\Delta|m|^H$ simplifies.
- Re-Pair and MinEnt alphabet size is increasing and the message length is reduced.
- Language produced by CT grammar contains more than one message $\Rightarrow$ rules must be applied in the reversed order.

Context Transformations in Comparison with MinEnt cont.

Filename	GCT	HOCT	MinEnt
book1	3.848	3.001	2.282
paper1	4.197	2.316	1.978
progc	4.335	2.346	1.886
alice29.txt	-	2.608	1.939
bible.txt	-	2.662	1.461
world192.txt	-	2.617	1.314

Sources of Inefficiency

- Large output alphabet.
- Frequencies of symbols ranges from 1 to $\sigma + 1$.

Filename	$\lceil \log \sigma \rceil$	H_0	R
book1	15	13.417	1.583
paper1	12	10.765	1.235
progc	12	10.496	1.504
alice29.txt	13	11.860	1.140
bible.txt	17	14.887	2.113
world192.txt	16	14.444	1.556

MinEnt - The Smallest Grammar Problem

Approximation Ratio $a(n)$

$$a(n) = \max_{x \in \Sigma^n} \left(\frac{\text{grammar size for } x \text{ produced by } A}{\text{size of the smallest grammar for } x} \right)$$

- Charikar et.al showed the class of strings σ_k for which Re-Pair has $a(n) = \Omega(\sqrt{\log n})$.

$$\sigma_k = \prod_{w=\sqrt{k}}^{2\sqrt{k}} \prod_{i=0}^{w-1} (x^{b_{w,i}}|)$$

- Using the same class of strings MinEnt won't infer any production rule and it has $a(n) = \Omega(\frac{n}{\sqrt{\log n}})$.

DBC Comparison

- DBC - Delimiter Based Compression
- DBCR - DBC followed by Re-Pair
- HuffW - HuffWord algorithm
- WLZW - Word based LZW
- WLZ77 - Word based LZ77

Filename	*DBC*	$DBCR_{ph}$	$DBCR_{th}$	*HuffW*	*WLZW*	*WLZ77*
bible	1.932	**1.692**	1.557	2.274	1.923	1.712
world192	2.365	1.589	1.475	2.220	1.698	**1.433**

MinEnt - Time Complexity

Suppose a message $m = (\prod_{x \in \Sigma} x)^2$. Example: $m = (abcd)^2 = abcdabcd$.

- Message length: $n = |m| = 2|\Sigma| = 2\sigma$
- Number of candidate rules: $d = \sigma = n/2$.
- Number of $\Delta|m|^H$ computations c, when $\Delta|m|^H$ is recomputed in each iteration of the algorithm:

$$c = d + d - 1 + \cdots + 1 = \frac{d(d+1)}{2} = \frac{n^2}{8} + \frac{n}{4} = O(n^2)$$

- If we recompute $\Delta|m|^H$ per $\frac{n}{\log n}$ iterations:

$$c = O(n \log n)$$

- If we compute $\Delta|m|^H$ once in the beginning and once when all $\Delta|m|^H$ are negative:

$$c = O(n)$$

MinEnt vs. Re-Pair - Execution Times

- k - per how many iterations of MinEnt we recompute $\Delta|m|^H$ of all candidate pairs.
- $k = SE$ - $\Delta|m|^H$ computed once in the beginning and when all pairs have $\Delta|m|^H$ negative.

		MinEnt		
Filename	Re-Pair	$k = SE$	$k = \frac{n}{\log n}$	$k = 1$
alice29.txt	0.060	0.062	0.068	5.929
bible.txt	1.815	1.750	1.784	1238.288
world192.txt	1.008	0.982	0.993	516.038

Speeding up MinEnt

- Do we have to recompute all $\Delta|m|^H$ in each iteration of the algorithm and still obtain the same or at least approximately the same order of pairs?
- The change of $\Delta|m|^H$ of symbols whose frequency doesn't change is less significant than the entropic size change of symbols participating in the rule.
- When all symbols have very low frequency (below f_{lim} given in Proposition 1 in thesis) we can switch to Re-Pair processing and $\Delta|m|^H$ will be always positive.
- Compute $\Delta|m|^H$ only to avoid reduction paradox in Re-Pair.

Extra Comments

$$H_0(S) = \sum_{x \in \Sigma} \frac{n_x}{n} \log \frac{n}{n_x} = \log n - \frac{1}{n} \sum_{x \in \Sigma} n_x \log n_x$$

Is it sufficient to find a pair (a, b) that maximizes the following formula to mimic behaviour of MinEnt?

$$n_a \log n_a + n_b \log n_b - n_{ab} \log n_{ab}$$