Codificador Universal Via Recorrência de Padrões usando Alfabeto Expandido

André C. G. C. Reis
IME
reis@epq.ime.eb.br

Weiler A. Finamore
PUC-Rio
weiler@cetuc.puc-rio.br

Marcelo S. Pinho
UNESP
mpinho@feg.unesp.br

Resumo— **Este trabalho apresenta a descrição de uma variação do algoritmo de Lempel-Ziv baseado em um alfabeto expandido, que utiliza a idéia de Tunstall [7] para construção de alfabetos. Utilizando esta nova técnica sobre a variação do introduzida por Welch [4], conhecida como algoritmo *LZW*, é proposta neste trabalho uma nova versão, *tLZW*. Testes desta nova versão foram realizados através da compressão de arquivos componentes do *Canterbury* e *Calgary Corpi*. Os resultados de desempenho, modestamente superiores, quando comparado com o *LZW*, são indicam a potencialidade do novo conceito.**

Palavras-chave— **Codificação de fonte, compressão de dados, codificação universal de fonte, códigos de Tunstall.**

I. Introdução

Seja $x_i^j = x_i, x_{1+i}, \ldots, x_j$ uma seqüência de símbolos pertencentes a um conjunto finito $\mathcal{A}$, com cardinalidade α. Seja $\mathcal{A}^*$ o conjunto de todas as seqüências finitas formadas por símbolos de $\mathcal{A}$; $\mathcal{A}^\infty$ o conjunto de todas as seqüências inifinitas formadas por símbolos de $\mathcal{A}$; e $\mathcal{A}^n$ o conjunto de todas as seqüências formadas por n símbolos de $\mathcal{A}$. Seja ainda p uma medida de probabilidade em $\mathcal{A}^\infty$ e X_1^n uma variável aleatória definida a partir da medida p. A probabilidade de $X_1^n = x_1^n$, medida através de p, é representada por $Pr[X_1^n = x_1^n]$. Uma fonte de informação seqüencial é definida por um par $(\mathcal{A}, p)$, onde $\mathcal{A}$ é denominado alfabeto da fonte.

Um codificador sem perda de informação é definido como sendo um mapa biunívoco $C : \mathcal{A}^* \rightarrow \mathcal{B}^*$, onde $\mathcal{B}$ é o alfabeto de saída do codificador. Assim sendo, para cada seqüência $u \in \mathcal{A}^*$, o codificador associa um elemento distinto $C(u) \in \mathcal{B}^*$. Este trabalho assume que $\mathcal{B} = \{0, 1\}$, sem qualquer perda de generalidade.

Se u é uma seqüência de símbolos de algum alfabeto, o comprimento de u, representado por $|u|$, é definido como sendo o número de símbolos de u, i.e., $|x_1^n| = n$. O problema básico da teoria da codificação é minimizar o valor esperado do comprimento da palavra código, i.e., $E[|C(X_1^n)|]$ para uma fonte $(\mathcal{A}, p)$ dada. Um resultado fundamental da teoria da codificação de fonte diz que para todo o codificador sem perda $E[|C(x_1^n)|] \geq H(X_1^n) \geq nH(\mathcal{A}, p)$, onde $H(X_1^n)$ é a entropia de X_1^n definida por

$$H(X_1^n) = \sum_{x_1^n \in \mathcal{A}^n} -Pr[X_1^n = x_1^n] \log_2 Pr[X_1^n = x_1^n], \tag{1}$$

e $H(\mathcal{A}, p)$ é a entropia da fonte, definida por

$$H(\mathcal{A}, p) = \lim_{n \rightarrow \infty} \frac{H(X_1^n)}{n}. \tag{2}$$

A utilização da técnica de recorrência de padrões ("string matching") na compressão de dados teve sua origem em [3]. A partir dos resultados obtidos em [3], foram desenvolvidos diferentes codificadores universais, entre eles o $LZ78$, proposto em [6]. Devido à sua baixa complexidade computacional e ao seu bom desempenho, este codificador se tornou extremamente popular. Após a publicação do artigo original [6], surgiram diversas variações deste codificador e algumas destas versões se tornaram padrões em diferentes aplicações (por exemplo o padrão V.42bis da UIT). Uma das versões mais populares do $LZ78$ é o codificador LZW proposto em [4].

Existem vários codificadores, o codificador de Huffman o mais conspícuo, para os quais $E[|C(x_1^n)|] \approx H(X_1^n)$ e cujas taxas de compressão (em bits por símbolo da fonte) $E[\frac{|C(x_1^n)|}{n}]$ convergem para a entropia da fonte $H(\mathcal{A}, p)$, quando $n \rightarrow \infty$.

Na realidade o o codificador de Huffman, C_{H_L}, atua sobre blocos $x_{n_i}^{L+n_i}$ de tamanho L (supor que n é múltiplo de L), tal que $\left[x_1^n = x_1^{L+1}, \ldots, x_{n_i}^{L+n_i}, \ldots, x_{n-L}^n\right]$ e a palavra-código resultante $C(x_1^n)$ corresponde a uma concatenação das palavras-códigos $C_{H_L}(x_{n_i}^{L+n_i})$ obtidas pela aplicação sucessiva do código de Huffman L-dimensional aos blocos de x_1^n. É fato conhecido que a eficiência do codificador de Huffman aproxima-se de 100% quando L cresce. Na maioria das aplicações prática, no entanto, usa-se $L = 1$, de tal forma que a complexidade do codificador não inviablize a implementação.

Se a cardinalidade do alfabeto da fonte for, por exemplo, $|\mathcal{A}| = 256$, usar $L = 2$ corresponde a implementar um codificador de Huffman bi-dimensional com um alfabeto de $|\mathcal{A}^2| = 256^2$ símbolos. Este pequeno aumento da dimensionalidade do codificador provoca uma enorme expansão do dicionário. Uma solução intermediária para expandir de forma controlada o dicionário foi proposta por Tunstall [7]. A proposição de Tunstall se aplica a codificadores de fontes não universais. O presente trabalho desenvolver um algoritmo de compressão sem perdas utilizando a idéia de Tunstall.

Este trabalho é dividido da seguinte forma. A seção 2 apresenta um modelo geral para os códigos seqüenciais via recorrência de padrões e utiliza este modelo para descrever o código LZW. A seção 3 descreve o algoritmo de Tunstall e a seção 4 apresenta o novo codificador $tLZW$ juntamente com um exemplo de aplicação. Os resultados obtidos são apresentados na Seção 5. Encerrando o trabalho, a Seção 6 apresenta as conclusões.

II. Codificadores Seqüenciais via Recorrência de Padrões

O princípio de um codificador seqüêncial via recorrência de padrões é utilizar um dicionário de padrões, criado a partir de uma parcela da seqüência que já foi codificada, para codificar a parcela que falta ser codificada. No início do processo, existe um dicionário inicial e a cada parcela da seqüência que é codificada, o dicionário é atualizado.

A Figura 1 ilustra um passo genérico i, de um codificador seqüencial que utiliza a técnica de recorrência de padrões. Nela, a sequência $x_1^{n_i}$ representa a parte da sequência que foi codificada antes do passo i. A sequência $x_{1+n_i}^n$ representa a parte da sequência que ainda falta ser codificada. O bloco "Dicionário" constrói o dicionário que será utilizado no passo i, utilizando a sequência $x_1^{n_i}$. O bloco "Encontra próximo bloco" encontra o bloco de símbolos $s_i = x_{1+n_i}^{n_{1+i}}$ que será codificado no passo i. O bloco "Código" codifica o bloco s_i, gerando a palavra código $C_{D_i}(s_i)$. Em geral, este código é bem simples, baseado apenas em uma indexação dos blocos pertencentes ao dicionário, e não utiliza diretamente uma medida estatística dos símbolos emitidos pela fonte.

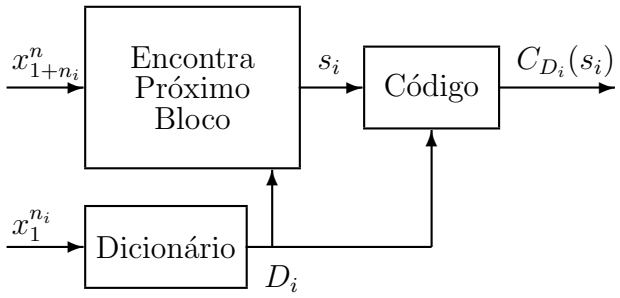

Fig. 1
Codificadores Seqüênciais via Recorrência de Padrões.

Seguindo o modelo geral, apresentado na Figura 1, é possível descrever qualquer codificador seqüêncial via recorrência de padrões. Para isso, basta estabelecer os seguintes procedimentos.

1. Procedimento para escolha de dicionário inicial D_0;
2. Procedimento para encontrar o próximo bloco, a partir do dicionário D_i;
3. Procedimento para construir o dicionário;
4. Procedimento para codificar os blocos do dicionário.

O algoritmo proposto tem como base a mais popular implementação de um algoritmo de Lempel-Ziv, o algoritmo LZW [4]. Seguindo a descrição geral, o LZW é apresentado.

LZW
1. Dicionário inicial: $D_0 = \mathcal{A}$;
2. Método para encontrar o próximo bloco: s_i é o maior prefixo de $x_{1+n_i}^n$ que pertence ao dicionário D_i;
3. Atualizar dicionário: o dicionário D_{1+i} é obtido

com a inclusão do menor prefixo de $x_{1+n_i}^n$, que não pertence ao dicionário D_i, i.e. $D_{1+i} = D_i \cup \{x_{1+n_i}^{1+n_{1+i}}\}$;
4. Código: codificar o bloco s_i, através da sua posição no dicionário.

III. Algoritmo de Tunstall

Tunstall descreveu em seu trabalho [1] um algoritmo muito simples para um dicionário ótimo unicamente percorrível com M entradas para uma fonte discreta sem memória com alfabeto de tamanho K. Este algoritmo é descrito a seguir, a título de ilustraç
ao.
1. Começar com cada simbolo da fonte como uma entrada do dicionário.
2. Se o número total de entradas é menor que M, então avançar par o passo 3, senão parar.
3. Escolher a entrada mais provável σ e trocar esta entrada com as K cadeias que são extensões de uma única letra de σ. Não alterar as outras entradas. Ir para o passo 2.

É interessante observar que o modelo geral da Figura 1 pode ser utilizado para descrever o Algoritmo de Tunstall ainda que ele não seja universal. Neste caso o dicionário D_i utilizado em cada passo é fixo.

IV. Codificador Proposto

tLZW
1. Dicionário e Alfabeto iniciais: A. $D_0 = \mathcal{B}_0 = \mathcal{A}$, $i = 1$ e $n_i = 1$
2. Método para encontrar o próximo bloco: s_i é o maior prefixo de $x_{n_i}^n$ que pertence ao dicionário D_{i-1}. $s_i = \pi_L\left(x_{n_i}^n | D_{i-1}\right)$ e $l_i = |s_i t|$
3. Se $l_i = 1$ então $b = x_{n_i}^{n_i+1}$ e, dicionário e alfabeto serão atualizados com a inclusão do elemento b, ou seja, $D_i = D_{i-1} \cup \{b\}$ e $\mathcal{B}_i = \mathcal{B}_{i-1} \cup \{b\}$. Senão $b = \pi_L\left(x_{n_i+l_i+1}^n | \mathcal{B}_{i-1}\right)$ e somente o dicionário será atualizado com $(s_i + b)$, ou seja, $D_i = D_{i-1} \cup \{(s_i, b)\}$ e $\mathcal{B}_i = \mathcal{B}_{i-1}$.
4. Código - codifica o bloco s_i, através da sua posição no dicionário. $s_i = d_k \in D_{i-1} \Rightarrow C_i[s_i] = \beta_{\lceil \log_2(|D_{i-1}|)\rceil}(k)$ onde $\beta_{\lceil \log_2(|D_{i-1}|)\rceil}(k)$ corresponde à representação binária com $\lceil \log_2(|D_{i-1}|)\rceil$ bits do inteiro k.
5. $n_{i+1} = n_i + l_i$.

O exemplo a seguir ilustra o funcionamento do *tLZW*.

Exemplo 1: Considerando o alfabeto quaternário, i.e., $A = \{a, b, c, d\}$ e a seqüência $x_1^n = aacabadababaacadabacabadadaaababaaba$, então o codificador funciona da seguinte forma. $D_0 = \mathcal{B}_0 = A$.

1. $n_1 = 1$
   $s_1 = \pi_L(x_1^n | D_0) = a$
   $l_1 = |s_1| = 1$
   $b = x_{n_1}^{n_1+1} = aa$
   $D_1 = D_0 \cup \{aa\}$ e $\mathcal{B}_1 = \mathcal{B}_0 \cup \{aa\}$
   $s_1 = d_0 \in D_0 \Rightarrow C_1[s_1] = \beta_{\lceil \log_2(|D_0|)\rceil}(0) = 00$
   $n_2 = n_1 + l_1 = 2$
2. $n_2 = 2$
   $s_2 = \pi_L(x_2^n | D_1) = a$
   $l_2 = |s_2| = 1$
   $b = x_{n_2}^{n_2+1} = ac$
   $D_2 = D_1 \cup \{ac\}$ e $\mathcal{B}_2 = \mathcal{B}_1 \cup \{ac\}$
   $s_2 = d_0 \in D_1 \Rightarrow C_2[s_2] = \beta_{\lceil \log_2(|D_1|)\rceil}(0) = 000$
   $n_3 = n_2 + l_2 = 3$
3. $n_3 = 3$
   $s_3 = \pi_L(x_3^n | D_2) = c$
   $l_3 = |s_3| = 1$
   $b = x_{n_3}^{n_3+1} = ca$
   $D_3 = D_2 \cup \{ca\}$ e $\mathcal{B}_3 = \mathcal{B}_2 \cup \{ca\}$
   $s_3 = d_2 \in D_2 \Rightarrow C_3[s_3] = \beta_{\lceil \log_2(|D_2|)\rceil}(2) = 010$
   $n_4 = n_3 + l_3 = 4$
4. $n_4 = 4$
   $s_4 = \pi_L(x_4^n | D_3) = a$
   $l_4 = |s_4| = 1$
   $b = x_{n_4}^{n_4+1} = ab$
   $D_4 = D_3 \cup \{ab\}$ e $\mathcal{B}_4 = \mathcal{B}_3 \cup \{ab\}$
   $s_4 = d_0 \in D_3 \Rightarrow C_4[s_4] = \beta_{\lceil \log_2(|D_3|)\rceil}(0) = 000$
   $n_5 = n_4 + l_4 = 5$
5. $n_5 = 5$
   $s_5 = \pi_L(x_5^n | D_4) = b$
   $l_5 = |s_5| = 1$
   $b = x_{n_5}^{n_5+1} = ba$
   $D_5 = D_4 \cup \{ba\}$ e $\mathcal{B}_5 = \mathcal{B}_4 \cup \{ba\}$
   $s_5 = d_1 \in D_4 \Rightarrow C_5[s_5] = \beta_{\lceil \log_2(|D_4|)\rceil}(1) = 001$
   $n_6 = n_5 + l_5 = 6$
6. $n_6 = 6$
   $s_6 = \pi_L(x_6^n | D_5) = a$
   $l_6 = |s_6| = 1$
   $b = x_{n_6}^{n_6+1} = ad$
   $D_6 = D_5 \cup \{ad\}$ e $\mathcal{B}_6 = \mathcal{B}_5 \cup \{ad\}$
   $s_6 = d_0 \in D_5 \Rightarrow C_6[s_6] = \beta_{\lceil \log_2(|D_5|)\rceil}(0) = 0000$
   $n_7 = n_6 + l_6 = 7$

7. $n_7 = 7$
   $s_7 = \pi_L(x_7^n|D_6) = d$
   $l_7 = |s_7| = 1$
   $b = x_{n_7}^{n_7+1} = da$
   $D_7 = D_6 \cup \{da\}$ e $\mathcal{B}_7 = \mathcal{B}_6 \cup \{da\}$
   $s_7 = d_3 \in D_6 \Rightarrow C_7[s_7] = \beta_{\lceil \log_2(|D_6|) \rceil}(3) = 0011$
   $n_8 = n_7 + l_7 = 8$
8. $n_8 = 8$
   $s_8 = \pi_L(x_8^n|D_7) = ab$
   $l_8 = |s_8| = 2$
   $b = \pi_L(x_{8+2+1}^n|\mathcal{B}_7)$
   $D_8 = D_7 \cup \{abab\}$ e $\mathcal{B}_7 = \mathcal{B}_6$
   $s_8 = d_7 \in D_7 \Rightarrow C_8[s_8] = \beta_{\lceil \log_2(|D_7|) \rceil}(7) = 0111$
   $n_9 = n_8 + l_8 = 10$

V. Resultados

Um conjunto de arquivos de teste bastante utilizado para medir o desempenho de codificadores universais foi proposto em [2], e é denominado *conjunto de Canterbury*. Este trabalho aplicou os codificadores LZW (largamente usado na implementação do *compress* do $UNIX$) e $tLZW$ no *conjunto de Canterbury* e também no *conjunto de Calgary*. Os resultados obtidos estão apresentados na Tabela 1 e 2, respectivamente. Os resultados apresentados para os algoritmos LZW e do $tLZW$ indicam os tamanhos dos arquivos gerados *bytes*.

Arquivos - x_1^n	$\|x_1^n\|$	LZW	$tLZW$
alice29.txt	152089	61573	61946
asyoulik.txt	125179	54990	55916
cp.html	24603	11317	11082
fields.c	11150	4964	4761
grammar.lsp	3721	1813	1757
sum	38240	20102	20025
xargs.1	4227	2339	2243

TABELA I
Resultados para o *Conjunto de Canterbury*

Os resultados da Tabela 1 e 2 mostram que o $tLZW$ produz um ganho de 4% a 5% em arquivos com menos de $50Kbytes$ sobre o LZW. Para arquivos maiores, como o alfabeto desta versão implementada cresce sem limites, o desempenho do codificador proposto é inferior; em torno de 2%.

Arquivos - x_1^n	$\|x_1^n\|$	LZW	$tLZW$
bib	111261	46528	46134
trans	93695	38240	35565
progc	39611	19143	18766
obj1	21504	14048	13760
paper1	53161	25077	24732
progl	71646	27148	26306
paper2	82199	36161	35989

TABELA II
Resultados para o *Conjunto de Calgary*

VI. Conclusões

Este trabalho apresentou um algoritmo para compressão de dados, sem perda, universal, baseado na técnica de recorrência de padrões. Diferentemente dos algoritmos descritos na literatura, que, para construção do seus dicionários, usam como alfabeto do codificador, o alfabeto da fonte $\mathcal{A}$, o presente algoritmo (designado por $tLZW$) expande o seu alfabeto $\mathcal{B}$ recursivamente de tal forma que , $\mathcal{B} \subset \mathcal{A} \cup \mathcal{A}^2$.

Os resultados preliminares apresentados indicam que o desempenho do $tLZW$ é comparável ao desempenho do LZW. A contribuição significativa deste trabalho consistiu em aplicar a idéia de Tunstall aos codificadores universais tendo demonstardo a validade do novo conceito. O estudo realizado indica ainda que existem varias formas que podem ser exploradas para melhorar o desempenho do $tLZW$, tais como restrição do alfabeto, podagem da arvore do dicionário entre outras. Estes novos aspectos estão sendo examinados.

References

[1] B. P.Tunstall, "Synthesis of noiseless compression codes," Ph.D. dissertation, Georgia Inst. Technol., Atlanta, GA, 1967.

[2] R. Arnold, & T. Bell, "A corpus for the evaluation of lossless compression algorithms," Proc. of IEEE Data Compress Conference, pp. 201-210, UT, 1997.

[3] A. Lempel & J. Ziv, "On the complexity of finite sequences," IEEE Trans. Inform. Theory, vol. IT-22, pp. 75-81, 1976.

[4] T. A. Welch, "A technique for high-performance data compression," IEEE Computer, vol. 17, no 6, pp. 8-19, 1984.

[5] J. Ziv & A. Lempel, "A universal algorithm for data compression," IEEE Trans. Inform. Theory, vol. IT-23, pp. 337-343, 1977.

[6] J. Ziv & A. Lempel, "Compression of individual sequences via variable-rate coding," IEEE Trans. Inform. Theory, vol. IT-24, pp. 530-536, 1978.

[7] S. A. Savari, "Renewal Theory and Source Coding," Proc. of the IEEE, pp. 1692-1702, vol.8 No.11, November 2000.