MÔ HÌNH NÉN CHỈ MỤC TỆP ĐẢO TRONG THƯ VIỆN SỐ

Đỗ Quang Vinh

Tóm tắt: Bài báo khảo sát và đánh giá các mô hình nén chỉ mục tệp đảo tài liệu văn bản trong thư viện số, nhấn mạnh đến các mô hình Bernoulli cục bộ.

1. ĐẶT VẤN ĐỀ

Các tệp đảo (IF) nén là phương pháp chỉ mục hữu ích nhất một cơ sở dữ liệu (CSDL) lớn các tài liệu văn bản có độ dài có thể thay đổi trong thư viện số. Kích thước của một IF có thể được giảm đáng kể bằng cách nén. Ở đây, chúng tôi khảo sát các mô hình và phương pháp mã hoá để nén chỉ mục tệp đảo (IFID) CSDL tài liệu trong thư viện số.

Chìa khoá của bài toán nén là nhận xét mỗi một danh sách đảo (IL) có thể được lưu trữ như một dãy số nguyên tăng dần, không mất tính tổng quát. Chẳng hạn, giả sử thuật ngữ nào đó xuất hiện ở 8 tài liệu của một CSDL – gồm có 3, 5, 20, 21, 23, 76, 77, 78. Thuật ngữ được mô tả ở IF bằng một danh sách: <8; 3, 5, 20, 21, 23, 76, 77, 78>, địa chỉ của nó được chứa trong từ vựng. Tổng quát hơn, danh sách đối với một thuật ngữ t lưu trữ số tài liệu *ft* trong đó thuật ngữ xuất hiện và sau đó, một danh sách của số tài liệu f_t: <f_t; d_1, d_2, ... d_{f_t}>, trong đó $d_k < d_{k+1}$. Bởi vì danh sách số tài liệu bên trong mỗi một IL được sắp tăng dần và tất cả xử lý là tuần tự từ đầu danh sách, danh sách có thể được lưu trữ như một vị trí ban đầu tiếp theo bởi một danh sách của d-gap, hiệu $d_{k+1} - d_k$. Tức là, danh sách đối với thuật ngữ ở trên có thể được lưu trữ dễ dàng như: <8; 3, 2, 15, 1, 2, 53, 1, 1>. Không thông tin nào bị mất, vì số tài liệu gốc thường nhận được bằng cách tính tổng tích luỹ của d-gap.

Hai dạng là tương đương, nhưng không rõ ràng bất kỳ sự tiết kiệm đạt được d-gap lớn nhất ở biểu diễn thứ hai là có khả năng giống như số tài liệu lớn nhất ở biểu diễn thứ nhất và như vậy, nếu có N tài liệu trong CSDL và một mã hoá nhị phân phẳng được dùng để biểu diễn kích thước gap, cả hai phương pháp đòi hỏi [logN] bit cho mỗi con trỏ lưu trữ. Tuy nhiên, xem xét mỗi một IL như một danh sách của d-gap, tổng của nó bị giới hạn bởi N, cho phép cải thiện biểu diễn và có thể mã hoá các IL thực chất dùng trung bình nhỏ hơn [logN] bit cho mỗi con trỏ.

Nhiều mô hình riêng biệt được đề xuất để mô tả phân bố xác suất của kích thước d-gap. Chúng được nhóm lại thành hai lớp chính: phương pháp toàn cục, trong đó mọi IL được nén dùng mô hình thông thường giống nhau và phương pháp cục bộ, trong đó mô hình nén đối với mỗi một danh sách thuật ngữ được điều chỉnh theo tham số lưu trữ nào đó, thường là tần suất của thuật ngữ. Các mô hình toàn

cục tự phân chia thành tham số và không tham số. Các mô hình cục bộ luôn được tham số hoá.

2. CÁC MÔ HÌNH NÉN TOÀN CỤC

2.1 Mô hình không tham số

Mã toàn cục đơn giản nhất là biểu diễn cố định của các số nguyên dương. Chẳng hạn, như đã được xem xét, nếu có N tài liệu trong CSDL, một mã hoá nhị phân phẳng có thể được sử dụng, yêu cầu [logN] bit đối với mỗi một con trỏ.

Mối quan hệ của Shannon giữa độ dài mã lý tưởng l_x và xác suất P [x] như sau:

$$l_x = - \log P [x] \qquad (1)$$

cho phép phân bố xác suất hàm ý bởi phương pháp mã hoá riêng biệt được xác định. Mô hình xác suất ẩn kết hợp với một mã nhị phân phẳng là mỗi một kích thước d-gap trong mỗi một IL đều là ngẫu nhiên trong 1 ... N, không phản ánh chính xác thực tế.

Ý tưởng về một mã trong phạm vi phân bố xác suất hàm ý là một cách tốt để truy cập bằng trực giác liệu có thể làm tốt và khi xem xét theo quan điểm này, tất cả kích thước gap có xác suất như nhau dường như không thể xảy ra. Chẳng hạn, các từ thông thường có khả năng có gap nhỏ giữa hai lần xuất hiện – nói cách khác, chúng có thể không kết thúc xuất hiện thường xuyên. Tương tự, các từ không thường xuyên có khả năng có gap là rất lớn, dù cho nếu các tài liệu được lưu trữ theo thứ tự thời gian hoặc thứ tự logic khác nào đó. Như vậy, các biểu diễn có độ dài thay đổi nên được xem xét, trong đó các giá trị nhỏ được xem xét có khả năng hơn và mã hoá kinh tế hơn so với các giá trị lớn.

Bảng 1 - Các mã mẫu đối với các số nguyên.

Gap x	Phương pháp mã hoá				
	Đơn nguyên	γ	δ	Golomb	
				b = 3	b = 6
1	0	0	0	00	000
2	10	100	1000	010	001
3	110	101	1001	011	0100
4	1110	11000	10100	100	0101
5	11110	11001	10101	1010	0110
6	111110	11010	10110	1011	0111
7	1111110	11011	10111	1100	1000
8	11111110	1110000	11000000	11010	1001
9	111111110	1110001	11000001	11011	10100
10	1111111110	1110010	11000010	11100	10101

Một mã như thế là mã đơn nguyên. Ở mã này, một số nguyên $x \geq 1$ được mã hoá thành x-1 một bit tiếp theo bằng một bit 0, như vậy, mã đối với số nguyên 3 là 110. Cột thứ hai của bảng 1 trình bày một số mã đơn nguyên. Mặc dù mã hóa đơn nguyên nhất định có khuynh hướng thiên về gap hẹp, xu hướng thường là quá mức. Một IL mã hoá bằng đơn nguyên đòi hỏi dft bit, vì mã đối với một gap của x đòi hỏi x bit và ở mỗi một IL tổng kích thước gap là số tài liệu df của lần xuất hiện cuối cùng của từ tương ứng. Như vậy, tổng cộng, một IF mã hoá bằng đơn nguyên có thể tốn bằng N.n bit và nói chung đây là số cực lớn.

Xét xác suất, rõ ràng mã đơn nguyên tương đương với gán một xác suất $P[x] = 2^{-x}$ vào gap có độ dài x và đây là quá nhỏ.

Nhiều mã có phân bố xác suất hàm ý nằm ở chỗ nào đó giữa phân bố đều giả thiết bằng một mã đơn nguyên và sự phân rã hàm mũ nhị phân hàm ý bằng mã đơn nguyên. Một là mã γ biểu diễn số x như một mã đơn nguyên đối với $1 + [\log x]$ tiếp theo bằng một mã của $[\log x]$ bit biểu diễn giá trị của $x - 2^{[\log x]}$ thành nhị phân. Phần đơn nguyên định rõ bao nhiêu bit được đòi hỏi để mã hoá x và sau đó, phần nhị phân thực sự mã hoá x thành nhiều bit đó. Chẳng hạn, xét x = 9. Sau đó, $[\log x] = 3$ và như vậy, $4 = 1 + 3$ được mã hoá thành đơn nguyên (mã 1110) tiếp theo bằng $1 = 9 - 8$ thành một số nhị phân 3-bit (mã 001), tổ hợp nó để cho một từ mã của 1110001.

Ví dụ khác của mã γ được trình bày ở cột thứ ba của bảng 1. Mặc dù chúng có độ dài khác nhau, các từ mã có thể được giải mã rõ ràng. Tất cả bộ giải mã phải làm đầu tiên là trích lọc một mã đơn nguyên c_u và sau đó xem bit $c_u - 1$ tiếp theo như một mã nhị phân để nhận được một giá trị thứ hai c_b. Giá trị x được trả lại, sau đó, dễ dàng được tính bằng $2^{c_u - 1} + c_b$. Đối với mã 1110001, $c_u = 4$ và $c_b = 1$ là giá trị của 3 bit tiếp theo và như vậy giá trị $x = 9 = 2^3 + 1$ được trả lại. Mặc dù nó có thể làm tốt hơn bằng một số phương pháp mô tả dưới đây, tuy nhiên mã γ tốt hơn nhiều nhằm mã hoá gap IF so với cả một mã hoá nhị phân lẫn một mã hoá đơn nguyên và nó đúng là dễ mã hoá và giải mã. Nó biểu diễn một gap x bằng $l_x \approx 1 + 2\log x$ bit, như vậy, xác suất hàm ý về một gap của x là

$$P[x] = 2^{-lx} \approx 2^{-(1+2\log x)} = \frac{1}{2x^2} \qquad (2)$$

cho một mối quan hệ đảo bình phương giữa kích thước gap và xác suất.

Tổng quát hơn, xem xét mã γ là chia nó thành hai thành phần: một mã đơn nguyên biểu diễn một giá trị k + 1 có liên quan đến vector nào đó $V = \langle v_i \rangle$ như là

$$\sum_{i=1}^{k} v_i < x \leq \sum_{i=1}^{k+1} v_i$$

tiếp theo bằng một mã nhị phân của $[\log v_k]$ bit biểu diễn giá trị dư

$$r = x - \sum_{i=1}^{k} v_i - 1$$

Theo khuôn khổ này, mã γ sử dụng vector

$V\gamma = <1, 2, 4, 8, 16, ...>$

và x = 9 được mã hoá với k = 3 và r = 1. Tương tự, mã đơn nguyên có quan hệ hồi quy đến mức độ nào đó với vector

$V_u = <1, 1, 1, 1, 1, ...>$

Sự phát triển sâu hơn là mã δ, trong đó tiền tố chỉ thị số bit hậu tố nhị phân được biểu diễn bằng mã γ đúng hơn mã đơn nguyên. Lấy chính mẫu của x = 9, tiền tố đơn nguyên của 4 mã hoá 110 được thay bằng 11000, mã γ đối với 4. Tức là, mã δ đối với x = 9 là 11000001.

Nói chung, mã δ đối với một số nguyên tuỳ ý đòi hỏi

$$l_x = 1 + 2[\log(1 + [\log x])] + [\log x] = 1 + 2[\log\log 2x] + [\log x]$$

bit. Đảo công thức, phân bố hàm ý được xấp xỉ bằng

$$P[x] \approx 2^{-(1 + 2\log\log x + \log x)} = \frac{1}{2x(\log x)^2}$$

(3)

Bảng 1 cho các mẫu của mã δ đối với các giá trị khác nhau x. Mặc dù đối với các giá trị x nhỏ chứng tỏ mã δ dài hơn mã γ, trong giới hạn, vì x trở nên lớn, trạng thái bị đảo. Đối với một giá trị x như 1000000, mã δ tốt hơn, đòi hỏi 28 bit so với 39 bit của γ.

2.2. Mô hình Bernoulli toàn cục

Một cách hiển nhiên tham số hoá mô hình và có thể nhận được nén tốt hơn là sử dụng mật độ thực của con trỏ trong IF. Giả thiết tổng số con trỏ *f* được lưu trữ biết trước. Chia *f* cho số thuật ngữ chỉ mục và sau đó cho số tài liệu, coi một xác suất của *f*/(N.n) là bất kỳ tài liệu lựa chọn ngẫu nhiên chứa bất kỳ thuật ngữ lựa chọn ngẫu nhiên. Sau đó, sự xuất hiện con trỏ có thể được mô hình hoá như một quá trình Bernoulli với xác suất này, bằng giả thiết các con trỏ *f* của IF được lựa chọn ngẫu nhiên từ n.N cặp tài liệu-từ có thể trong CSDL.

Giả thiết trường hợp ngẫu nhiên của một gap có kích thước x có xác suất x - 1 lần không xuất hiện của từ riêng biệt đó, mỗi một của xác suất (1 – p), tiếp theo bằng một lần xuất hiện có xác suất p, là $P[x] = (1 - p)^{x-1}p$. Đây chính là phân bố hình học và tương đương với mô hình hoá mỗi một cặp tài liệu-thuật ngữ có thể

như xuất hiện độc lập với xác suất p. Nếu mã hoá số học được sử dụng, các xác suất tích luỹ yêu cầu có thể được tính bằng cách lấy tổng phân bố này:

$$\text{cận\_dưới} = \sum_{i=1}^{x-1}(1-p)^{i-1}p = 1 - (1-p)^{x-1} \qquad (4)$$

$$\text{cận\_trên} = \sum_{i=1}^{x}(1-p)^{i-1}p = 1 - (1-p)^{x}$$

Khi giải mã, công thức xác suất tích luỹ $1 - (1 - p)^x$ phải được đảo để xác định x và đảo chính xác theo thứ tự đối với bộ giải mã để thực hiện đúng. Hàm nghịch đảo $x = 1 + [(\log(1 - p)) /(\log(1 - v))]$, trong đó v là giá trị phân số của đích mã hoá số học, sinh ra giá trị giải mã x.

Các xác suất sinh ra bằng phân bố hình học có thể được biểu diễn bởi một mã kiểu Huffman đặc biệt hiệu quả và thành ra là một sự lựa chọn có ích hơn để mã hoá số học. Phương pháp tiếp theo được gọi là mã Golomb. Đối với tham số b nào đó, bất kỳ số $x > 0$ được mã hoá thành hai phần: thứ nhất, $q + 1$ thành đơn nguyên, trong đó số thương $q = [(x - 1)/ b]$; sau đó, số dư $r = x - qb - 1$ mã hoá thành nhị phân, đòi hỏi cả [log b] lẫn [log b] bit.

Gallager và Van Voorhis chỉ ra nếu b được chọn để thoả mãn

$$(1-p)^b + (1-p)^{b+1} \le 1 < (1-p)^{b-1} + (1-p)^b \qquad (5)$$

phương pháp mã hoá này sinh ra một mã tiền tố tự do tối ưu đối với phân bố hình học tương đương với các phép thử Bernoulli với xác suất thành công cho trước bằng p. Theo một nghĩa nào đó, sự xây dựng của Golomb là một phương pháp 1-bước nhằm tính mã Huffman đối với tập xác suất vô hạn riêng biệt này, rõ ràng không thể sử dụng giải thuật Huffman truyền thống. Giải phương trình 1 đối với b cho

$$b^A = \left\lceil \frac{\log(2-p)}{-\log(1-p)} \right\rceil \qquad (6)$$

trong đó chỉ số trên A chỉ thị số bit trung bình đòi hỏi để mã hoá IF được cực tiểu hoá.

Giả thiết $p = f / (N.n) << 1$, một trường hợp đơn giản hoá hữu ích là

$$b^A \approx \frac{\log_e 2}{p} \approx 0.69.\frac{N.n}{f} \qquad (7)$$

Đối với CSDL TREC, tham số được dùng là b = 2039.

Mã Golomb có quan hệ gần với mã đơn nguyên và giống như mã đơn nguyên gán xác suất giảm theo hàm mũ. Tuy nhiên, ở mã Golomb cơ sở của sự phân rã theo hàm mũ là một hàm của b và thường rất gần tới 1. Thực vậy, nó là các xác suất giảm theo hàm mũ trả lại mã Golomb phù hợp để sử dụng khi phân bố cơ bản là hình học. Trong phạm vi của biểu diễn vector, mã Golomb là một mã có quan hệ với vector: V_G = <b, b, b, b, ...>.

Mã hoá Golomb cho các kết quả trong khoảng vài phần trăm nén nhận được bởi một mô hình Bernoulli với mã hoá số học nếu p << 1 và B >> 1, là trường hợp chuẩn về nén IF. Chỉ khi nhiều thuật ngữ xuất hiện với xác suất rất cao thực hiện tối ưu mã hoá số học dẫn đến một sự cải thiện đáng kể và đối với hầu hết ứng dụng bao hàm một mô hình Bernoulli, thực tế là cách tiếp cận Golomb sinh ra phương pháp lựa chọn giải mã nhanh hơn nhiều thực hiện nó. Chú ý nếu p vượt quá 0.5, mã Golomb hiệu quả hơn nếu IF được bù trước khi nén – tức là, nếu các số tài liệu không chứa thuật ngữ được lưu trữ hơn là các số tài liệu chứa. Lý do là ở trường hợp này một số đầu ra cần được mã hoá thành nhỏ hơn 1 bit, là không thể được với mã Golomb.

3. CÁC MÔ HÌNH NÉN CỤC BỘ

3.1 Mô hình Bernoulli cục bộ

Nếu tần suất f_t của thuật ngữ t biết trước, một mô hình Bernoulli trên mỗi một IL riêng biệt có thể được sử dụng. Mã Golomb lại được đòi hỏi ít khắt khe hơn về mặt tính toán so với mã hoá số học và cho nén tương tự. Chẳng hạn, nếu IL được rút ra từ một CSDL có N = 78 tài liệu, phương trình 6 quy định $b_t^A = 6$.

Sử dụng phương pháp này, các từ thường gặp được mã hoá với giá trị b nhỏ, trong khi các từ thường gặp được mã hoá với giá trị lớn. Ở CSDL TREC, 1 từ chỉ dùng 1 lần – 1 từ chỉ xuất hiện 1 lần - được mã hoá với b ≈ 500000, bằng 19, 20 hoặc 21 bit. Mặt khác, 1 từ xuất hiện bằng 10% trong số tài liệu được mã hoá với b = 7 và ở trường hợp này, một gap của nó được biểu diễn đúng bằng 3 bit. Điều này so sánh có lợi với cực tiểu của 11 bit – 1 đối với một phần đơn nguyên cực tiểu cộng 10 đối với phần nhị phân cực tiểu - đòi hỏi dùng giá trị tối ưu toàn cục của b = 2036.

Các từ rất thông thường được mã hoá với b = 1. Khi b = 1 mã thoái hoá thành một tập mã đơn nguyên đối với kích thước gap không có thành phần nhị phân. Điều này tương đương với lưu trữ IL bằng một bitvector, tức là, vì một vector nhị phân với 1 bit cho mỗi một tài liệu, bit được cài đặt nếu thuật ngữ xuất hiện trong tài liệu đó. Như vậy, biểu diễn IF nén dùng mô hình Bernoulli mã hoá Huffman không bao giờ có thể xấu hơn so với một chỉ mục nén một bitvector cho mỗi thuật ngữ.

Để khai thác mô hình cục bộ này, cần lưu trữ tham số f_t với mỗi một IL, sao cho giá trị chính xác của b có thể được dùng trong khi giải mã. Tổng giá thực hiện

nhỏ. Mỗi một IL nén dễ dàng được tiếp đầu ngữ với một mã γ đối với f_t – mã γ là một lựa chọn tốt bởi vì hầu hết tần suất có thể được mong đợi nhỏ. Thực vậy, ở TREC, khoảng một nửa tần suất f_t bằng 1 và lưu trữ f_t có chi phí tương đối nhỏ.

3.2 Mô hình Bernoulli lệch

Như mã γ, vector đối với mã Golomb là V_G = <b, b, b, ...> và bởi vì kích thước *bucket* đều đã sử dụng, một lượng lớn đối xứng lệch của phân bố γ bị mất. Vì vậy, mã Golomb cục bộ chỉ thực hiện ở mép tốt hơn so với mã γ và δ toàn cục.

Thực tế, không hợp lý mong đợi mỗi một thuật ngữ riêng lẻ bị phân tán ngẫu nhiên trong suốt các tài liệu bao hàm một CSDL. Đúng hơn, có khả năng có nhiều giai đoạn dài kém hoạt động, đặt rải rác theo cụm tài liệu chứa một từ nhất định hoặc *cluster* – cluster được gom nhóm đồng thời trong CSDL có thể bởi vì chúng bắt nguồn từ chính tài nguyên hoặc có thể bởi vì chúng thảo luận tư liệu chủ đề nào đó và các tài liệu trong CSDL được chèn theo thứ tự thời gian. Hơn nữa, gom nhóm không bị hạn chế các tên thích hợp.

Một cách làm méo phân bố xác suất hình học cho phép nhằm gom nhóm là nâng các xác suất ẩn của d-gap nhỏ lên không gây bất lợi quá cho gap lớn. Để thực hiện, một sự pha tạp giữa các mã γ và Golomb có thể được sử dụng, dùng một vector mã có các bucket ban đầu nhỏ trở thành to lớn (đúng hơn duy trì tất cả bucket cùng kích thước) và cho phép bucket thứ nhất chứa giá trị b (đúng hơn chỉ 1). Một vector có thể là V_T = <b, 2b, 4b, ..., 2^ib, ...>, trong đó một giá trị b cho các kết quả tốt là kích thước gap median trong mỗi một IL. Nghĩa là một nửa trong số gap rơi vào trong bucket thứ nhất của mã với 1 bit tiền tố đơn nguyên đơn. Đối với bất kỳ giá trị *ft* cho trước, phân bố lệch có một median nhỏ hơn so với các phân bố ngẫu nhiên, như vậy, thành phần hậu tố nhị phân đối với một nửa hoặc nhiều hơn trong số con trỏ danh sách cũng có thể ngắn hơn so với cùng mã V_G. Ở trường hợp xấu nhất, trên một IL thực sự ngẫu nhiên, median được gần tới trung bình và mã VT chỉ thực hiện xấu hơn mã Golomb chút ít.

Để sử dụng mã V_T, từ đó một giá trị b có thể được tính phải được lưu trữ trong mỗi một IL, vì f_t không còn hiệu quả. Vì b lớn đối với các thuật ngữ hiếm gặp, một biểu diễn mã γ của tỉ số N/b nên được thêm vào, từ đó b có thể được tính tại chế độ thực.

3.3 Mô hình nén nội suy

Mặc dù được thúc đẩy như một cơ chế đương đầu với gom nhóm xuất hiện từ, mã V_T vẫn là một mã tĩnh và tương đương với một mô hình bậc 0 đối với d-gap. Sử dụng một mô hình bậc cao hơn cũng cho phép nén nhạy với gom nhóm vì một dãy d-gap nhỏ là bằng chứng rõ ràng d-gap tiếp theo cũng nhỏ. Một cơ chế được giả thiết tham số b đã dùng đối với mỗi một d-gap bằng trung bình của số nào đó của d-gap đã giải mã trước đây. Trong khi hấp dẫn về lý thuyết, lợi ích nén phụ thường nhỏ và vì có nhiều trường hợp hơn được điều khiển, sự cài đặt phức tạp

hơn. Phải chú ý để đảm bảo hồi phục nhanh từ các đánh giá không đúng. Bất kỳ sự tiết kiệm nào bị mất ngay nếu chẳng hạn, một tham số b = 1 được tính tại điểm nào đó và một gap dài tiếp theo mã hoá bằng đơn nguyên. Vì vậy, các mã dựa trên vector kiểu V_T được sử dụng nhiều hơn so với vector kiểu V_G.

Một cách tinh tế hơn trong đó có thể nén mỗi một IL nhạy với gom nhóm. Mã nội suy được minh hoạ tốt nhất với một mẫu. Xét IL <7; 3, 8, 9, 11, 12, 13, 17> trong một CSDL có N = 20 tài liệu. Các cơ chế nén chỉ mục khác nhau mô tả ở trên chuyển đổi danh sách này thành một danh sách d-gap <7; 3, 5, 1, 2, 1, 1, 4> và mã hoá nó theo cách từ trái sang phải, có thể nhận dạng cluster giữa con trỏ thứ hai và thứ sáu.

Để thay thế giả thiết giá trị của con trỏ thứ hai đã biết đến một mức độ nào đó trước khi con trỏ thứ nhất phải được mã hoá. Ví dụ, nếu biết rõ con trỏ thứ hai đang trỏ tới tài liệu 8 và sau đó con trỏ thứ nhất bị hạn chế số tài liệu nào đó trong dải 1 đến hết 7. Một sự gán đơn giản của các từ mã sau đó thêm hậu tố để biểu diễn con trỏ tài liệu thứ nhất này thành 3 bit.

Bây giờ, giả sử cả số tài liệu thứ tư lẫn thứ hai đã biết. Con trỏ tài liệu thứ tư trỏ tới tài liệu 11, như vậy, con trỏ thứ ba bị ràng buộc từ dải 9 đến 10. Một mã đơn giản – ở trường hợp này chỉ đúng 1 bit dài – có thể được dùng lại để biểu diễn con trỏ thứ ba. Tính ngắn gọn của từ mã này là một hệ quả trực tiếp của sự kiện có một cluster và cả hai con trỏ cận trên và cận dưới ở trong cluster. Như một mẫu cực đoan hơn không thay đổi, nếu cả hai con trỏ thứ tư và thứ sáu biết rõ (tới tài liệu 11 và 13 tương ứng), sau đó, con trỏ tài liệu thứ năm có thể được biểu diễn dùng một từ mã dài 0 bit – nó phải trỏ tới tài liệu 12.

Biểu diễn dựa trên giả thiết các con trỏ thứ hai, thứ tư và thứ sáu đã biết. Để biểu diễn chúng, một danh sách <3; 8, 11, 13> phải được mã hoá trước. Kỹ thuật tương tự có thể được dùng đối với danh sách này. Nếu con trỏ thứ hai (hướng tới tài liệu 11) biết rõ thì con trỏ thứ nhất (hướng tới tài liệu 8) lấy hầu hết 4 bit. Thật vậy, vì phải có 1 con trỏ hướng tới bên trái và 1 con trỏ hướng tới bên phải của tài liệu này, dải có thể bị hẹp hơn từ 2...9 và 1 mã 3-bit có thể được sử dụng. Bằng cách lập luận tương tự, con trỏ thứ ba phải nằm giữa 13 = 12 + 1 và 19 = 20 – 1 và 3 = [log7] bit hậu tố.

Vấn đề còn lại duy nhất là mã hoá con trỏ hướng tới tài liệu 11. Một mã 5-bit trong dải 1...20 chắc chắn là đủ và nếu biết rằng có 3 con trỏ tài liệu hướng tới bên trái và 3 hướng tới bên phải của con trỏ giữa này được khai thác, dải có thể bị hẹp hơn từ 4...17 và 4 bit là hiệu quả.

Quá trình hồi quy về tính các dải và mã giành được ở giải thuật mã hoá nội suy. Hàm *manhiphan(x, lo, hi)* được giả thiết để mã hoá một số $lo \le x \le hi$ theo cách thích hợp nào đó. Cơ chế thực hiện đơn giản nhất đòi hỏi [log(hi – lo + 1)] bit.

Đối với danh sách mẫu, dãy đầy đủ của bộ ba (x, lo, hi) xử lý bởi hàm *manhiphan* là (11, 4,17), (8, 2, 9), (3,1, 7), (9, 9, 10), (13, 13, 19), (12, 12, 12), (17,

14, 20). Với sự cài đặt đơn giản của *manhiphan*, độ dài của mã tương ứng là 4, 3, 3, 1, 3, 0 và 3 bit tương ứng đối với tổng 17 bit. Bằng cách so sánh, một mã Golomb đối với danh sách như nhau với b = 2 đòi hỏi 18 bit.

Có thể cải thiện nén nhiều hơn nữa bằng cách dùng một mã nhị phân cực tiểu. Chẳng hạn, khi bộ ba (13, 13, 19) đang được xử lý, chỉ có 7 số trong dãy, như vậy, một trong số từ mã có thể ngắn hơn 1 bit; nếu có 6 giá trị có thể, 2 trong số từ mã có thể ngắn hơn 1 bit. Nói chung, các từ mã ở giữa trong mỗi một dãy nên là từ mã ngắn hơn vì rất có thể giá trị ở giữa trong một danh sách con trỏ sẽ gần giữa dãy hơn so với gần điểm mút. Nhưng tại bước cuối cùng của quá trình, khi chỉ có một con trỏ bên trái ở mỗi một khoảng, nên đảo phân phối. Sau đó, các từ mã ngắn hơn 1 bit nên được gán tại điểm mút của dãy vì giả thiết cơ bản của toàn bộ phương pháp là các con trỏ tài liệu được gom nhóm.

Về phần *manoisuy*(L, f, lo, hi), trong đó L[0 ... (f – 1)] là một danh sách sắp xếp của số tài liệu f , tất cả nằm trong dải lo ... hi,

1. Nếu $f = 0$ thì trả lại xâu rỗng.
2. Nếu $f = 1$ thì trả lại *manhiphan*(L[0], lo, hi).
3. Khác
   a. Đặt h ← f div 2 và m ← L[h].
   b. Đặt f_1 ← h và f_2 ← f - h - 1.
   c. Đặt L_1 ← L[0 ... h-1] và L_2 ← L[(h + 1) ... (f – 1)].
   d. Trả lại *manhiphan*(m, lo + f_1, hi – f_2) ++
      manoisuy(L_1, f_1, lo, m – 1) ++
      manoisuy(L_2, f_2, m + 1, hi) ++

Giải thuật mã hoá nội suy

Ở trường hợp xấu nhất, mã nội suy chỉ không hiệu quả chút ít so với một mã Golomb và bởi vì tập các giá trị kề nhau có thể được mã hoá nhỏ hơn 1 bit mỗi một, ở trường hợp tốt nhất nó có thể tốt hơn nhiều. Hơn nữa, trong thực tế mã nội suy thường nén rất tốt. Thật vậy, hạn chế thực sự duy nhất của phương pháp là độ phức tạp cài đặt của nó – mã hoá và mỗi một giải mã sử dụng một stack của giá trị sắp xảy ra và vòng lặp mã hoá và giải mã trở nên chi tiết hơn nhiều so với mã Golomb đơn giản hơn và mã γ.

4. ĐÁNH GIÁ VÀ KẾT LUẬN

Bảng 2 trình bày nén nhận được trên CSDL TREC thử nghiệm bằng các phương pháp khác nhau mô tả ở trên. Các kích thước xuất được biểu diễn bằng số bit cho mỗi con trỏ. Kích thước tổng của chỉ mục có thể được tính bằng cách nhân với giá trị f thích hợp từ bảng 1. Chẳng hạn, sự sử dụng mã nội suy sinh ra một chỉ mục TREC có 83.4 MB, hoặc chỉ mục đúng 4% văn bản. Đây đúng là một thành tựu đáng kể khi nhớ rằng một số tài liệu đối với mỗi một từ và số trong CSDL 2 GB này được lưu trữ trong chỉ mục. Để tham khảo, hàng giá trị thứ hai chỉ ra

không gian được yêu cầu cho mỗi con trỏ bằng mã hoá nhị phân bình thường kích thước gap.

Các kết quả của bảng bao gồm bất kỳ chi phí cần thiết, như giá trị f_t mã hoá γ đối với mô hình Bernoulli cục bộ và tập đầy đủ các mô hình.

Tần suất của một thuật ngữ là một tiêu chí dự báo tốt hơn nhiều của phân bố kích thước gap của nó so với toàn bộ phân bố kích thước gap đối với tất cả thuật ngữ. Hơn nữa, căn cứ vào các mô hình Bernoulli lệch và Bernoulli cục bộ chỉ cần một tham số được lưu trữ trong bộ nhớ trong khi giải mã, so với hàng trăm và có thể hàng nghìn tham số đòi hỏi bởi các mô hình khác, chắc chắn các mô hình cục bộ tốt hơn đáng kể. Ngay cả các mã γ và δ toàn cục trở nên gần đáng kể tới nén đạt được bằng mô hình Bernoulli cục bộ và Bernoulli lệch. Hai mã cuối cùng này có ưu điểm chính không đòi hỏi tham số, có ích khi lưu trữ CSDL động.

Tất cả mô hình dựa vào mã tiền tố, như vậy, lợi ích nén không đáng kể có thể đưa đến nếu các phân bố xác suất cơ bản được dùng bắt buộc thay thế bộ mã số học. Tuy nhiên, không thể có bất kỳ lợi ích lớn hơn đáng kể để biện hộ thời gian giải mã phụ.

Bảng 2 - Nén IF bằng số bit cho mỗi con trỏ đối với TREC

Phương pháp	Số bit cho mỗi con trỏ
Các phương pháp toàn cục	
Đơn nguyên	1918
Nhị phân	20.00
Bernoulli	12.30
γ	6.63
δ	6.38
Các phương pháp cục bộ	
Bernoulli	5.84
Bernoulli lệch	5.44
Nội suy	5.18

Mã nội suy cho các kết quả tốt nhất trên CSDL TREC, tiếp theo mô hình Bernoulli lệch, với tham số b đã chọn bằng kích thước gap median trong mỗi một IL. Hơn nữa, tất cả mã đòi hỏi các tài nguyên tính toán tương đối vừa phải trong khi nén và giải nén chỉ mục nhanh như khi truy cập chỉ mục như thực hiện các mã nhị phân đơn giản. Mô hình Bernoulli cục bộ mã hoá dùng mã Golomb là một lựa chọn tốt, nhận được nén ít hơn không đáng kể so với mã nội suy nhưng cài đặt đơn giản hơn.

Tóm lại, các mô hình cục bộ có xu hướng thực hiện nén tốt hơn mô hình toàn cục và không hiệu quả hơn về thời gian xử lý đòi hỏi trong khi giải mã, vì chúng có xu hướng cài đặt phức tạp hơn. Đối với đa số mục đích thực hành, mô

hình nén chỉ mục phù hợp nhất là phương pháp Bernoull cục bộ, cài đặt dùng kỹ thuật mã hoá Golomb.

TÀI LIỆU THAM KHẢO

[1] Arms W.Y., *Digital Libraries*, MIT Press, Cambridge, 2003.

[2] Elias P., 'Universal Codeword Sets and Representations of the Integers', IEEE Transactions on Information Theory 21(2), pp. 194-203, 1975.

[3] Fox E.A., *Advanced Digital Libraries*, Virginia Polytechnic Institue and State University, 2000.

[4] Golomb S.W., 'Run-Length Encodings', IEEE Transactions on Information Theory 12(3), pp. 399-401, 1966.

[5] Journal of Network and Computer Applications, Special Issue of JNCA on Digital Libraries 20 (1-2), 1997.

[6] National Institute of Standards and Technology, NIST Special Publication Text Retrieval Conference (TREC).

[7] Mendelhall W., Sincich T., *Statistics for the Engineering and Computer Science*, 2nd Edition, Collier Macmillan, London, 1989.

[8] Salomon D., *Data Compression*, 2nd Edition, Springer, Berlin, 2000.

[9] Ziv J., Lempel A., 'A Universal Algorithm for Sequential Data Compression', IEEE Transactions on Information Theory 23(3), pp. 337-343, 1977.

[10] Ziv J., Lempel A., 'Compression of Individual Sequences via Variable-Rate Coding', IEEE Transactions on Information Theory 24(5), pp. 530-536, 1978.