단어, 어원 Dictionary 에 의한 Text 압축

°이 재 영, 성 굉 모, 이 종 각
서울대 전자공학과

Text Compression by Word and Etymology Dictionary

°Jae Young Lee, Koeng Mo Sung, Chong Kak Lee
Dep. of Electronics Eng., Seoul National Univ.

Abstract

In this paper, a text compression method is proposed which is capable of reducing mean bits per character by word and etymology dictionary . This dictionary consists of 256 words and 512 etymologies with 10 bits length codes. Using this dictionary, a mean rate of 3.44 bits per character is achieved.

I. 서 론

Text압축 방법은 computer network 분야와 data base 분야에서 text를 code화 하는 과정에서 bit의 수를 줄이는 방법이다. 이러한 방법을 이용하여 Text를 압축할 때 얻을 수 있는 장점은 첫째 값 비싼 통신 회선의 점유 시간을 줄일 수 있는 점이며, 둘째 기억 장치를 효율적으로 사용할 수 있는 점이다.

압축은 text 뿐만 아니라 speech, telemetry, television, picture 등 여러 분야에 적용되고 있으며 연구 및 개발되고 있다. 그중 text를 압축하는 대표적인 방법들은 두가지로 분류할 수 있다. 첫째는 문자의 발생 빈도수에 따라 길이가 다른 code를 할당하는 방법이고(Huffman, 1952)이고 [3], 둘째는 8 bit code중 사용되지 않는 부분에 단어나 문자 string을 할당하는 방법들(Lea, 1978; Willams, 1987)이다 [9,11,12].

본 논문에서는 영어가 라틴어와 밀접한 관계가 있는 특수성을 이용하여 text를 압축하였다, 즉 영어 단어의 약 반수 이상이 라틴어로 구성되어 있기 때문에 사용 빈도수가 높은 라틴어계 어원을 512개를선택하여 그 어원에 확장 code를 할당하고 , 어원으로 압축할 수 없는 단어들 중 빈도가 높은 순으로 256개의 단어를 선택하여 그 단어에 확장 code를 할당 하여 단어.어원 dictionary를 만들었다. 또한 어원으로 분리되지 않는 문자를 압축하기 위해서 각 문자는 Huffman code법에 의한 문자table 을 사용함으로써 text를 압축 할 수 있는 방법을 제안한다.

II. 단어의 구성 및 Text 압축

1. 단어의 구성

일반적으로 단어는 합성와 파생어로 구성 및 생성되며 특히 영어 단어는 단어로 구성된 합성어와 접두어, 접미어, 어근 및 단어로 구성된 파생어가 대부분이다. 이러한 단어를 구성하는 합성법이나 접사 부가법은 단어와 단어와이 관계, 혹은 어근과 접사와의 관계를 나타내는 단어 syntax로부터 알 수 있다.

2. 평균 bit rate의 최소화

Text를 code화 할때 최소 bit 수를 나타내는 식을 유도하기 위해서 rankl 에서 N까지의 단어로 구성된 dictionary를 고려한다. 이 dictionary의 단어가 text에 나타날 비율은 다음과 같다.

$$P_N = \sum_{1}^{N} P_R \qquad (1)$$

여기서 P_R은 rank R인 단어의 확률로써 expirical rule(Zipf)에 의해서 다음과 같이 주어진다.

$$P_R = \frac{0.1}{R} \qquad (2)$$

N>1에 대해서 accurate가 1 %보다 더 좋은 summation식은 다음과 같다.(J. Pike)

$$\sum \frac{1}{R} = 0.577 + \ln\left(N+\frac{1}{2}\right) \quad (3)$$

$$P_N = 0.0577 + 0.1 \ln\left(N+\frac{1}{2}\right) \quad (4)$$

즉 dictionary의 단어의 수 N이 증가함에 따라 이 단어들이 text에 포함되는 비율이 증가하며, 그림1)의 점선곡선과 같이 나타낼수있다.

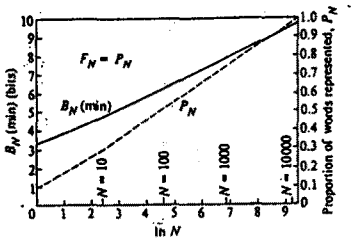

Fig 1. Minimum mean bits per word for a coding dictionary of N members

Rank R인 dictionary 단어를 $B_R$bit 로 coding 하고 이 bit 조합이 가능한 모든 bit 조합의 fraction F 로 표현한다면 , 즉 $B_R = -\text{lb}\ F_R$(lb : $\log_2$), 이때 text 에서 dictionary 단어의 평균 bit 길이 B_N 은 다음과 같다.

$$B_N P_N = \sum P_R B_R = -\sum P_R \text{lb}\ F_R \quad ---(5)$$

P_N 이 N 만의 함수일때 text 에 있는 dictionary 단어를 표현하는 bit 수를 최소화 하는 B_R (or F_R) 의 값은 B_N 을 최소화함으로서 얻어진 것과 같으며 $B_{N(min)}$ 은 다음과 같다[J.Pike].

$$B_{N(min)} = \frac{\ln 10}{\ln 2} - \frac{\ln(F_N/P_N)}{\ln 2} + \frac{[\ln(N+\frac{1}{2})]_{10} - 0.146}{[\ln(N+\frac{1}{2})+0.577]\ln 2} \quad (6)$$

$F_N = P_N$ 일때 단어당 최소 평균 bit 길이는 그림 1)의 실선 곡선으로 나타난다. 예를 들면 N = 100 일때 text 에 있는 단어 중 52 % 는 rank 1 에서 rank 100 까지의 dictionary 단어로 구성되며 이 단어는 평균 6.3 bit 로 나타낼수 있음을 의미한다. Text 에서 dictionary 단어를 제외한 free 문자를 code 화 할때 그 문자의 빈도 수에 따라 다른 bit 길이의 code 를 할당하면 문자당 평균 bit 길이는 다음과 같이 주어 진다[Huffman].

$$B_{K(min)} = -\text{lb}\ F_K - \sum_1^k \frac{P_J}{P_K} \text{lb}\ \frac{P_J}{P_K} \quad ---(7)$$

여기서 F_K 는 free 문자를 나타내는데 사용하는 bit 조합이 total fraction 이고, P_K 는 text 로부터 random 하게 선택한 문자가 free 문자일 확률이고, P_J 는 이 free 문자가 J 번째 문자일 확률을 나타내며 free 문자의 비율은 다음과 같다.

$$P_K = \sum_1^K P_J \quad ---(8)$$

모든 text 문자가 free 문자로 취급될때 최소 평균 값이 문자당 4.32 bit 로 주어진다. 200 단어 dictionary 를 이용한 경우는 다음과 같다.

$$B_{K(min)} = -\text{lb}\ F_K + 4.52 \quad ---(9)$$

full text 일때의 4.32 와 위 식의 4.52 간의 차이는 N 의 값에 따른 free 문자 분포의 변화를 반영한 것이다.

다음은 dictionary 단어에 대한 bit 의 최소치와 문자에 대한 bit 의 최소치를 구하기 위하여 text 전체가 code 화 되고 모든 available 한 bit 조합이 사용 된다면

$$P_N \frac{L_N}{L_\omega} + P_K = 1 \quad ---(10)$$

$$F_N + F_K = 1 \quad ---(11)$$

여기서 L_N은 text 에 있는 모든 dictionary 단어들의 평균 단어 길이이고 L_ω는 text 에 있는 모든 단어들의 평균 단어 길이이다. 실험적으로 L_ω 는 단어 앞의 space 를 포함해서 5.4 자이다. 또한 rank R 인 단어 길이에 대한 emperical rule 은 다음과 같다.[white]

$$L_R = 2.45 R^{0.159} \qquad R>10 \quad ---(12)$$

L_N 값은 이 L_R 값과 다음식으로부터 구할 수 있다.

$$L_N P_N = \sum L_R P_R = 1.58 (N+\tfrac{1}{2})^{0.159} - 1.42 \qquad N>10 \quad ---(13)$$

문자와 단어 coding 의 조합을 사용하여 text 를 code 화 할때 문자 당 평균 bit rate 와 최소 평균 bit rate 는 다음과같다.

$$B_T = \frac{P_N}{L_\omega} B_N + P_K B_K \quad ---(14)$$

$$B_{T(min)} = \frac{P_N}{L_\omega} B_{N(min)} + P_K B_{K(min)} \quad ---(15)$$

여기서

$$B_{N(min)} = \frac{\ln(10 P_N + 10 P_K L_N)}{\ln 2} + \frac{[\ln(N+\frac{1}{2})]^2 - 0.146}{20 P_N \ln 2} \quad ---(16)$$

$$B_{K(min)} = \ln[1 + P_N/(L_\omega P_K)] + 4.52 \quad ---(17)$$

$$P_N = 0.0577 + 0.1 \ln\left(N+\frac{1}{2}\right) \quad ---(18)$$

$$P_K = 1 - P_N L_N / L_\omega \quad ---(19)$$

$$= 1.26 - 0.289 (N+\tfrac{1}{2})^{0.159} \quad ---(20)$$

L_ω = 5.4 인 $B_{T(min)}$ 값에 대한 graph 가 그림2)에 나타나있다.

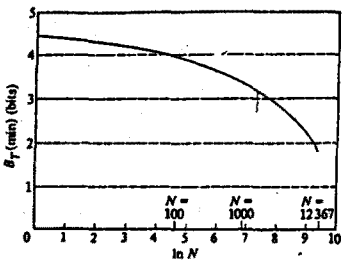

Fig 2. Minimum mean bits per character for the combined dictionary and character coding

이 그림은 dictionary 단어수 N 에 따른 문자 당 최소 평균 bit 수를 나타낸 것이다. 본 논문에서는 단어 이외에 어원 즉 접두어, 접미어, 어근을 포함한 dictionary 를 이용하여 문자 당 평균 bit 수를 줄이는 방법을 다음 절에 설명 하였다.

III. 단어.어원 dictionary 사용한 Text 압축

Dictionary를 사용한 text 압축 방법은 기본 문자, 256 개의 빈도수 높은 단어, 그리고 512개의 어원에 새로운 code를 할당한 다음 text의 각 단어를 읽어 분석한 후 그에 대응되는 code로 출력하는 것이다. 여기에 사용되는 code group은 문자 code, 단어 code 및 어원 code로 분류하며 그 format을 다음과 같이 표시하였다.

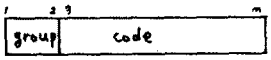

Fig 3. Code format

그림3)에서 각 format의 좌측 2bit는 code의 group을 구별하는 bit 이다. 이 bit가 "00"이면 문자 code format, "01"이면 단어 code format, "10"나 "11"이면 어원 code format을 의미한다.

문자 code 에서 문자는 A-Z의 영문자, space, ., , ' 만을 고려하였으며 Dewey(1924)가 100,000 단어를 분석한 문자의 빈도수 에 관한 graph는 그림4)과 같이 나타나 있다.

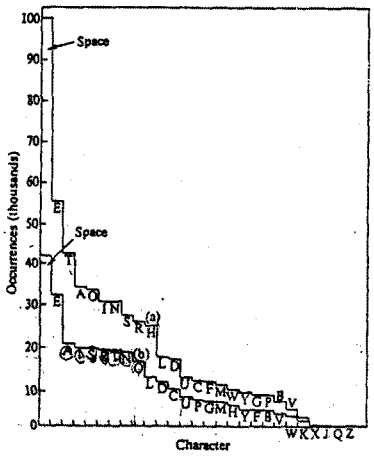

Fig 4. Character occurrence in a text of 100,000 word

a) full text (total 538023)
b) text minus 200 common words (total 308800 characters)

Fig 4. 에서 a)graph는 100000단어 즉 538023개의 문자로 구성된 text에서 free문자의 빈도수 순으로 나타낸 것이고 b)graph 는 이 text에서 빈도수가 큰 200 개의 단어를 제외한 308800 개의 free문자에 대한 빈도수를 나타낸 것이다. 문자 code의 할당 원칙은 Huffman code 방식과 같이 빈도수가 높은 문자는 짧은 bit 수의 code를 할당하고 빈도수가 낮은 문자는 긴 bit의 code를 할당하였으며 Fig 4. b)graph를 참조하였으며 table 1. 과 같다.

문자	code
space	00 00
E	00 01
A	00 10 00
I	00 10 01
⋮	
J	00 11 11 101
Q	00 11 11 110
Z	00 11 11 111

Table 1. Alphabet code

단어 code에서 단어는 빈도수가 높은 256개만 고려 하였다. 이 단어 code에 의한 압축은 어원이 포함되지 않는 짧은 단어에 효과가 있으며 또한 단어의 앞 뒤에 있는 space는 자동적으로 인식이 가능하기 때문에 space code를 생략할 수 있다.

어원 code에서 어원은 접두어, 접미어, 어근 및 빈도수가 높은 문자 string 512개 만을 고려 하였다. 이 어원은 여러개의 문자로 구성된 긴 단어를 압축할 때 주로 사용된다.

Dictionary를 사용한 text 압축 방법에 이용되는 table은 문자 code table, 이외에 단어 code와 어원 code로 구성된 단어-어원 dictionary로 구성되며 표 2)에 나타나 있다.

단어·어원	code
ab	10 0000000000
abil	10 0000000001
able	10 0000000010
about	01 0000000000
⋮	⋮
voc	11 1111111111
year	01 1111111100
years	01 1111111101
yet	01 1111111110
you	01 1111111111

Table 2. word-etymology dictionary

위에서 설명한 문자 code table과 단어-어원 dic

tionry을 사용하여 text를 압축하는 Algorithm은 간단하게 다음과 같이 기술할 수 있다.

step1: text file로부터 한 개의 단어 W(n)을 읽는다.

step2: 단어-어원 dictionary 에서 단어 W(n) 혹은 substring S를 찾는다.

step3: 단어를 찾았을 경우

그에 해당되는 단어의 확장 code를 단어-어원 dictionary 에서 찾아서 output 하고 그 단어의 전후에 있는 space code를 생략하고 step1 으로 가다.

substring S 를 찾았을 경우

그에 해당되는 어원의 code 를 단어 - 어원 dictionary 에서 찾아서 output 하고 이 substring S 를 제외한 나머지 문자중 최대의 substring S 를 가지고 step 2 로 간다.

step 4: substring S 를 찾지 못했을 경우

substring S 의 좌측 1 문자를 문자 code table 에서 찾아 그에 해당되는 code 를 output 하고 이 문자가 space 혹은 EOF 가 아니면 나머지 substring S 를 가지고 step 2 로 가고 space 이면 step 1 으로 가며 EOF 이면 stop 한다.

IV. 비교 및 검토

앞에서 제시한 text 압축 algorithm을 사용하면 약 1000개의 단어로 구성된 text를 압축한 결과로서 문자당 평균 3.44bit를 얻을 수 있다. 이 algorithm에 사용된 단어-어원 dictionary은 Dewey가 분석한 단어중에서 빈도수가 높은 256개의 기본 단어와 접두어, 접미어, 어근, 빈도수가 높은 string 등 약 512개의 어원으로 구성되었다. 이 algorithm과 비슷한 방법으로 J. Pike(1981)가 제안한 4bit Scheme을 이용한 text 압축 방법이 있으며 이 방법에 의한 text 압축 결과는 문자당 3.87bit로 나타나 있다. 이 방법과 본 논문에서 제시한 방법 간의 유사점은 text를 압축하기 위하여 빈도수가 높은 단어를 사용했다는 점이며 , J. Pike의 논문에 나타난 이론 및 실험 결과는 그림5)와 같이 나타나있다.

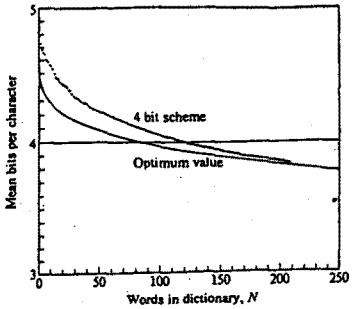

Fig 5. Mean bits per character for coding words

그림5)에서 수평축은 coding 한 단어의 수를 나타내고, 수직축은 한 문자당 평균 bit 수를 나타내었다. 이 graph에서 coding한 단어의 수가 증가할수록 한 단어당 평균 bit수가 감소한다는 것을 알 수 있다. 그러나 모든 단어를 coding한다는 것은 memory 공간과 search time 이 많이 요구되기 때문에 어려운 점이 많다. optimun value의 graph는 250개의 단어를 변화시키면서 coding 했을 때 문자당 가질 수 있는 한계의 bit 수를 타나낸 것이고, 4 bit Scheme의 graph는 4 bit Scheme 방법을 사용하여 text를 압축한 실험적 결과를 나타낸 것이다 [2]. "."로 표시한 점은 본 논문에서 제시한 dictionary를 사용하여 text를 압축한 실험 결과를 나타낸 것이다. 이 결과를 분석해보면 먼저 어원만을 사용했을 때 1000개로 구성된 text를 압축한 결과가 그림6)에 나타나있고, 어원과 단어를 사용했을 때 text를 압축한 결과가 그림7)에 나타나있다.

그림6)에서 어원에 의한 압축 효과는 6문자 이상으로 구성된 단어에 나타나있음을 알 수 있으며 text 전체의 압축률은 80%로서 문자당 평균 bit 수는 6.4bit로 나타났다.

그림7)는 어원과 단어를 고려하여 압축한 결과의 graph로서 단어에 의한 압축 효과는 6문자 이하로 구성된 단어에 나타나 있음을 알 수 있고 압축률은 46% 로서 문자당 평균 bit 수는 3.68bit 로 나타났다. 4개 이하로 구성된 단어를 압축하기 위하여 문자 code를 10bit로 고정하지 않고 Huffman code법을 사용하여 빈도수가 높은 문자는 적은 bit를 할당하고 빈도수가 낮은 문자는 많은 bit를 할당한 문자 code를 사용한 최종 결과는 43%로서 문자당 평균 bit수는

3.44 bit로 나타났다.

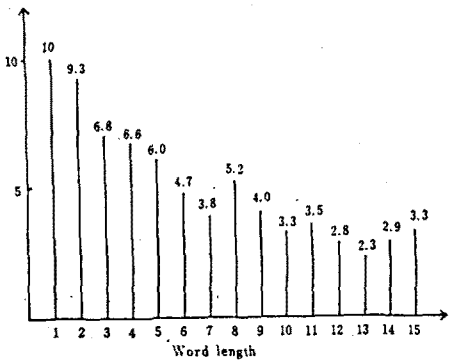

Fig 6. Mean bit per character for coding etymologies

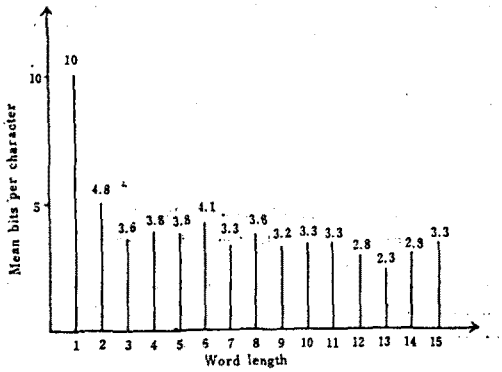

Fig 7. Mean bits per character for coding words and etymologies

이와 같은 문자, 단어와 어원에 의한 압축 효과를 분석해 볼 때, 그림5)에 나타난 optimun value나 4bit Scheme 방법은 문자와 단어 code만을 사용하였으나 본 논문에서 제시한 dictionary에 의한 방법은 이들 code 이외에도 어원 code를 부과함으로써 256 개의 단어를 code화한 조건 하에서 optimun value 나 4 bit Scheme 에서 얻은 결과 3.87bit 값보다 더 적은 3.44bit 값을 얻을 수 있었다.

V. 결 론

본 논문에서는 computer network 이나 data base 에서 text를 전송하거나 저장할 때, 통신 회선과 memory 를 효율적으로 사용하기 위하여 text 의 bit수를 줄일 수 있는 방법을 제시하였다. 이 방법에서 문자는 Huffman code를 할당하였고 단어, 어원은 기존의 8bit code 대신 10 bit로 확장한 code를 할당하였으며, 이 code 로 구성된 문자 code table과 단어-어원 dictionary을 사용하여 text를 압축하였다. 이 방법에 의한 text 압축률은 43%로서 한 문자당 평균 bit 수가 3.44bit로 나타났다. 256개의 단어를 code화 하는 똑 같은 조건하에서 비교해 볼 때 문자, 단어 이외에 어원을 고려함으로써 문자, 단어만을 고려한 optimun value 나 4 bit Scheme 방법에서 얻은 3.87bit 보다 더 개선된 결과를 얻을 수 있었다.

VI. 참고 문헌

1. Lynch,T.J. "Data Compression", Lifetime Learning Pub. 1985
2. Pike,J. "Text Compression using a 4-bit coding scheme", Computer Journal,ol.24,no.4,Nov.1981,pp324-330
3. Huffman,D.A. "A Method for the Construction of Minmum Redundancy Codes" Pro I.R.E.40 Sept. 1952,1098-1101
4. Cortesi,D. "An Effective Text-Compression Algorithm", Byte 7,1, Jan.1982 p 397-403
5. Longdom,G. and Rissanen,J. "A Double Adaptive File Co mpression Algorithm",IEEE Trans.on Comm.vol com-31,no 11,Nov. 1983,pp1253-1255
6. Young,T. and Liu,P. "Overhead Storage Consideration and a Multilinear Method of Data File Compression", EEE Trans. on Software Engineering vol. se-6, no 4 ,July 1980,pp340-347
7. Rubin,F. "Experiments in Text File Compression", comm. of the ACM,vol 19,no 11, Nov. 1976, pp617-623
8. Dewey,G."Relative Frequency of English Speech Sounds", Harvard Univ.Press,Cambradge,Mass. 1923
9. Lea,R.M. "Text Compression with an Associative Parallel Processor",he Computer Journal,vol.21,no1,Feb.1978, pp50-64
10. Shannon,C.E. "Prediction and Entropy of Printed English",Bell Sys.Tech.ournal,vol.30,1951,pp50-64
11. White,H.E. "Printed English Compression by Dictionary Encoding", pro. of theIEEE, vol.55,no.3,1967,pp390-396
12. Williams,P.W. "Criteria for Choosing Subjects to Obtain Maximum Relativ Entropy",the Computer Journal,vol. 21,no.1,1978,pp57-65
13. Wolff,J.G. "Recoding of Natural Language for Economy of Tranamission or Storage", the Computer Journal,vol. 21,no.1,1978,pp42-44
14. Selkirk,E.O. "The Syntax of Words",MIT Series 10,1982