허프만 부호화를 이용한 영문 텍스트 압축

구자룡, 최현호, *정제창
한양대학교
wkfyd5353@hanyang.ac.kr, hynuho0619@gmail.net, *jjeong@hanyang.ac.kr

English Text Compression using Huffman Coding

Jaryong Gu, Hyunho Choi, Jechang Jeong
Hanyang University

요 약

본 논문에서는 JPEG, MPEG 등 표준압축 기술에 사용되고 있는 무손실 압축 기법 중 Huffman coding 을 통해 영문 텍스트를 압축하고 압축률을 구해보았다. 각 글자를 Huffman coding 의 원리에 기초하여 빈도수에 따라 코드를 결정한다. 결정된 코드에 따라 영문 텍스트를 변환하여 압축을 진행한다. 본 연구에서는 MATLAB 을 이용하여 영문 텍스트의 각 글자 빈도수를 구하였고 Huffman coding 과정을 수행하였다. 또한 영문 텍스트를 코드로 변환과정을 수행하여 아스키코드와 압축률을 비교하였다. Huffman coding 은 아스키코드만으로 이용하는 것보다 1.89:1 의 압축률을 나타내었다.

1. 서론

최근 데이터 전송기술이 발전하면서 멀티미디어 전송 관련 기술들은 다양화 되고 있으며 데이터의 정보량을 줄임으로써 효율적인 데이터 전송을 가능하게 할 경우 전송 소요시간 단축, 통신비용 절감 등과 같이 데이터 통신에 많은 이득을 가져 올 수 있다. 데이터 압축 방법은 손실압축방법과 무손실압축방법이 있다. 무손실압축은 원래 내용을 그대로 디코딩 할 수 있기 때문에 다양한 압축방법의 마지막 과정에서 사용되고 있다. 무손실 압축 방법에는 Run-length coding, Arithmetic coding 및 Huffman coding 등이 있다[1]. 이 중 Huffman coding 은 출현빈도에 따른 코드할당을 통하여 데이터 압축을 진행하는 방법으로 높은 압축효율을 얻을 수 있기 때문에 널리 사용되고 있으며 압축률을 높이기 위한 다양한 방법들이 시도되고 있다[2].

아스키코드는 영문 및 기호 모두 8 비트로 표현하기 때문에 각 문자의 출현 빈도가 다른 데이터를 압축하는 데에는 낮은 효율을 나타낸다. 본 논문에서는 영문 텍스트에 사용되는 문자 및 기호를 Huffman coding 을 적용하여 압축률을 높이는 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2 장 본론에서는 Huffman coding 과 이를 이용한 영문 텍스트 압축과정에 대하여 논하며 3 장은 압축실험 결과에 대해서 다룬다. 마지막으로 4 장에서는 결론을 맺는다.

2. 본론

본 연구에서는 영문 텍스트 “1632”을 선정하였다[3]. 이 도서에서 사용되는 문자 및 기호들에 대해 Huffman coding 을 적용하여 코드로 변환하였다.

Huffman coding 은 문자의 출현 빈도에 따라 다른 길이의 부호를 할당하는 알고리듬이다. Huffman coding 은 문자들의 출현 빈도로부터 prefix code 를 만들어 내는 알고리듬으로, 출현 빈도가 낮은 문자일수록 긴 부호를 할당하고 높을수록 짧은 부호를 할당한다[4]. 예를 들어, 영문 텍스트에서 영문 한글자의 크기는 8 비트이고 컴퓨터는 8 자리 이진수인 아스키코드를 통하여 저장한다. 컴퓨터에서는 ‘ space’ 의 아스키코드 값을 ‘ 01000000’ 으로 나타낸다. 하지만 텍스트 내에서 ‘ space’ 는 높은 출현 빈도를 나타내므로 Huffman coding 은 ‘ space’ 에 짧은 코드를 할당하여 8 비트 이하로 부호화한다.

영문 텍스트 압축시스템은 아래 그림 1 과 같다. 각 과정은 다음과 같이 진행된다.

1) 빈도 조사: 영문 텍스트 "1632"의 Chapter1 에 있는 영문 A~Z 까지와 텍스트에 쓰이는 기호(‘ !’ , ‘ ?’ , ‘ ’ ’ , ‘ “ ’ , ‘ space’) 32 가지 문자의 출현 빈도수를 MATLAB 을 통하여 측정하였다.

2) Huffman coding: 코드에 대한 이진 트리를 만드는 방법은 먼저 출현 빈도가 가장 낮은 두 개의 문자를 할당하고 이 둘을 묶어 부모 노드를 만든다. 두 노드의 출현 빈도의 합이 부모 노드의 출현 빈도가 되는데, 이 과정을 반복적으로 진행한 결과 이진 트리가 형성된다.

각 문자는 트리의 잎 노드에만 연결이 되고, 트리의

표 1. 영문 텍스트에 대한 코드 변환 결과

문자	빈도수	확률	코드	코드길이
space	2789	0.0176	111	3
E	1499	0.0948	000	3
T	1014	0.0641	1010	4
A	1009	0.0638	1001	4
I	910	0.0575	1000	4
O	898	0.0568	0110	4
N	823	0.0520	0101	4
S	810	0.0512	0100	4
H	771	0.0488	0011	4
R	696	0.0440	11011	5
L	517	0.0327	10110	5
D	500	0.0316	01111	5
M	381	0.0228	00100	5
C	339	0.0214	110101	6
U	319	0.0202	110100	6
W	296	0.0187	110010	6
G	294	0.0188	110001	6
.	284	0.0180	110000	6
F	269	0.0170	101111	6
‘	268	0.0169	101110	6
Y	260	0.0164	011101	6
P	203	0.0128	011100	6
B	193	0.0122	001010	6
K	175	0.0111	1100111	7
V	110	0.0070	0010111	7
“	100	0.0180	0010110	7
?	48	0.0030	11001100	8
X	24	0.0015	1100110111	10
J	18	0.0011	1100110101	10
Q	16	0.0010	1100110100	10
Z	10	0.0000	11001101101	11
!	10	0.0000	11001101100	11
		Total = 1		

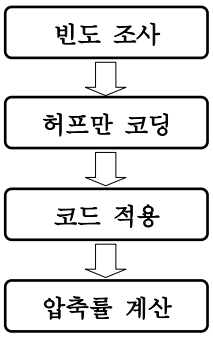

그림 1. 압축시스템의 구성

경로로부터 잎 노드에 이르는 경로에 따라 그 문자나 기호의 코드가 정해진다[5].

3) 코드 적용 및 압축률 계산: 아스키코드 대비 코드의 압축률을 비교하기 위하여 아스키코드와 코드의 비율을 구한다.

3. 결과

본 실험을 위해 사용한 영문 텍스트 "1632"의 Chapter1-62 까지는 8139856 바이트이고 이 중 Huffman coding 을 위해 사용한 Chapter1 은 126504 바이트이다. 출현 빈도수 검출 과정과 코드 치환 후 텍스트 길이 확인 과정은 MATLAB 을 이용하여 진행하였으며 Huffman coding 과정은 직접 진행하였다. 코드 변환 테이블은 표 1 과 같다. 표 1 에서 알 수 있듯이 Huffman coding 과정을 적용하여 얻은 코드의 크기는 짧은 경우 ‘ space’ 와 ‘ E’ 가 ‘ 111’ , ‘ 000’ 으로 3 비트를 나타내고 긴 경우는 ‘ Z’ 와 ‘ !’ 로 ‘ 11001101101’ , ‘ 11001101100’ 으로 11 비트로 나타냄을 확인하였다. 반면에 아스키코드는 영문 ‘ space’ , ‘ E’ , ‘ Z’ 및 ‘ !’ 를 포함한 영문 및 기호 모두 8 비트로 나타낸다. 본 실험에서는 영문 텍스트 “ 1632” 의 Chapter1, Chapter1-62 두 데이터를 가지고 압축을 진행하였으며 그 결과는 아래 표 2 와 같다. 압축률은 아스키코드와 코드 크기의 비를 구하였으며 두 경우 모두 1.89:1 임을 확인하였다. 이와 같은 연구 결과로부터 특정 문자의 빈도가 다른 문자에 비해 자주 사용되는 영문 텍스트는 8 비트의 고정길이를 가지는 아스키코드보다 가변길이를 가지는 코드를 사용하는 경우가 더 우수한 압축률을 나타냄을 확인하였다.

표 2. Huffman code 와 아스키코드의 성능 비교

	Chapter1	Chapter1-62
아스키코드	126504	8139856
Huffman code	66955	4297428
압축률	1.89:1	1.89:1

4. 결론

본 논문에서는 영문 텍스트의 각 글자당 출현 빈도수를 검출하고 이를 기반으로 Huffman coding 을 적용하여 압축하고 아스키코드와 비교하여 압축률을 비교하였다. Chapter1 과 Chapter1-62 두 가지 데이터의 압축률을 구해본 결과 1.89:1 로 코드가 아스키코드에 비해 우수한 압축 성능을 나타냄을 확인하였다. 추가적으로 본 연구에서는 영문과 기호에 대해서 Huffman coding 을 적용하였는데 추후 2 바이트 유니코드를 사용하는 한글 압축처리시에도 Huffman coding 을 적용하여 효율적인 압축률에 대한 연구가 더 진행되어야 할 것이다.

감사의 글

이 논문은 2015 년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2015R1A2A2A01006004).

참고문헌

[1] V. Singla, R. Singla, and S. Gupta, "Data compression modeling: Huffman and Arithmetic," International Journal of the computer, the Internet and Management, vol. 16, no. 3, pp. 64-68, Sep. 2008.

[2] S. Gupta, N. S. Mathur and P. Chauhan, "A New Approach to Encryption using Huffman Coding," International Journal of Progressive Sciences and Technologies vol. 2, no. 2, pp. 76-82, Apr. 2016.

[3] E. Flint, *1632,* 1st ed. Baen Books, 2000.

[4] M. Sharma, "Compression Using Huffman Coding," International Journal of Computer Science and Network Security, vol. 10, no. 5, pp. 133-141, May. 2010.

[5] H.R. Kim, Y. J. Jeong, C. H. Yim and H. S. Lim, "A Balanced Binary Search Tree for Huffman Decoding," The Journal of the Korean Institute of Communication Sciences vol. 30, no. 5, pp. 382-390, May. 2005.