한글의 형태적 특성을 이용한 한글 문서 압축 기법에 관한 연구

이 기 석† · 김 유 성††

요 약

본 논문에서는 한글 문서에 대해 높은 압축률을 얻기 위해 한글의 형태적 특징인 조사와 어말어미의 출현 빈도를 이용한 효율적인 한글 문서 압축 기법들을 제안하였으며 제안된 기법들의 성능 분석을 위하여 기존의 압축 기법들과 압축률을 비교 분석하였다.

한글 문서에서 조사와 어말어미가 반복적으로 출현한다는 형태적인 특성으로부터 높은 압축률을 얻기 위해 출현 빈도가 상대적으로 높은 64개의 조사 및 어말어미를 선정하여 고정 사전을 구성하고, 이를 이용하여 한글 문서를 압축하도록 기존의 LZ77기법과 LZW기법을 수정하여 각각 HLZ77기법과 HLZW기법을 제안하였다. 또한, 본 연구에서는 수정 제안된 HLZ77기법과 HLZW기법의 성능을 분석하기 위하여 4가지 기법을 실제 구현하여 여러 형태의 한글 문서를 대상으로 압축률을 비교하였다. 성능 결과로부터 일반적인 한글 문서에 대해 한글의 형태적인 특성을 이용하는 HLZ77기법과 HLZW기법이 각각 LZ77기법과 LZW기법 보다 우수한 압축률을 나타냄을 알 수 있었다.

A Study on Hangul Text Compression Using the Structural Feature of Hangul

Gi-Seog Lee† · Yoo-Sung Kim††

ABSTRACT

To achieve high compression ratio for Hangul texts, in this paper two text compression algorithms which use the structural feature of Hangul, the frequency of postpositional words, are proposed. The performances of these proposed algorithms are also compared with previous text compression algorithms.

The proposed compression algorithms named HLZ77 and HLZW come out from the modification of previous algorithms LZ77 and LZW, respectively. The major distinction of the proposed ones is that the proposed algorithms use the fixed dictionary of selected postpositional words that appear most frequently in Hangul texts. The performances of HLZ77 and HLZW also are compared with those of LZ77 and LZW, respectively, with respect to the compression ratio. According to the result of performance study, the proposed algorithms are better than the previous algorithms for descriptive Hangul text since the structural feature of Hangul is helpful to achievement of high compression ratio.

† 준 회 원:인하대학교 산업기술대학원 정보공학과
†† 정 회 원:인하대학교 산업기술대학원 정보공학과
논문접수:1996년 1월 24일, 심사완료:1996년 4월 9일

1. 서 론

컴퓨터로 저장 처리해야 할 정보의 양이 증가함에

따라 작은 저장 공간에 보다 많은 양의 정보를 저장하고 정보의 전송 비용을 감소하며 정보의 일괄적인 처리 비용을 절감할 수 있는 정보 압축의 필요성은 시간이 지날수록 더욱 증대되고 있다. 이러한 압축 기법의 역사는 1949년 발표된 셰논과 페노의 알고리즘으로부터 시작된다고 볼 수 있다[1]. VLC(Variable Length Coding) 방식이라 불리는 이 알고리즘은 당시로서는 파격적인 압축 방식을 통해 각종 프로그램의 저장에 상당한 효과를 보았는데 이는 RLC(Run Length Coding)방식 일변도였던 당시의 상황하에서 전혀 새로운 압축의 시대를 개막했던 것이다. 이후로 압축 프로그램에 대한 중요성과 열기는 점차로 뜨거워지기 시작했다. 그로부터 셰논과 페노의 압축 기법을 바탕으로 한 허프만 코딩과 산술 코딩이 약 30여년 동안 프로그램 압축의 시대를 이끌었다. 1977년 지브와 렘펠의 DBC(Dictionary Based Compression) 압축 방식은 압축의 역사에 새로운 전기를 마련했다. 지브와 렘펠이 제안한 압축 사전을 이용한 DBC 방식의 LZ77은 이후로 LZ78, LZW, 등 다양한 압축 프로그램의 탄생을 이끌며 현재까지 압축의 역사를 주도하고 있다.

그러나 한글은 영문과는 다른 독특한 형태적 구조를 가지고 있으며 따라서 보다 발전된 한글 압축을 위해서는 한글의 형태적 특성을 이용한 새로운 압축 기법의 연구가 필요하다. 이러한 측면에서 한글의 대표적 특징인 문법 형태소를 살피는 것은 매우 의미있는 일로 여겨진다.

한글 문서에서는 일부 조사와 일부 어미가 자주 출현하며 그 이외의 조사와 어미들은 출현 빈도가 상당히 낮은 것으로 확인되었다[2]. 따라서 출현 빈도의 상위에 있는 일부 조사와 어말어미는 텍스트 압축의 관점에서 고려할 때 충실한 압축 대상이 된다고 볼 수 있다. 그러므로 본 논문에서는 이러한 한글의 문법 형태소적 특징을 텍스트 화일의 압축에 이용한다면 효과적인 결과를 얻을 수 있으리라는 기대를 바탕으로 조사와 어말어미로 구성된 압축 사전을 이용하는 압축 알고리즘들을 제안하고 이들의 성능을 기존의 압축 알고리즘과 비교, 분석하였다.

본 논문은 다음과 같은 형식으로 구성되어 있다. 제2절에서는 관련 연구로서 VLC방식과 DBC방식으로 대별되는 압축 알고리즘을 개괄적으로 소개하고 있으며, 3절에서는 한글의 특성을 이용한 압축 알고리즘의 구현을 위해 대표적인 사전적 압축 알고리즘인 LZ77과 LZW에 조사와 어미 사전을 결합한 압축 알고리즘 HLZ77과 HLZW를 제안하였다. 4절에서는 한글의 형태적 특성을 이용한 한글 압축 기법인 HLZ77과 HLZW의 성능을 기존의 LZ77기법과 LZW기법의 성능과 각각 비교 분석하였으며 마지막으로 제 5절의 결론으로 본 논문을 마감하였다.

2. 관련연구

2.1 영문에 대한 압축 기법

2.1.1 VLC(Variable Length Coding) 방식[3]

초기의 압축 방식은 연속적으로 반복되는 문자의 저장 공간을 최소화하는 것을 기본으로 하는 RLC(Run Length Coding) 압축 방법이었다. 여기서 말하는 연속적인 반복이란 두 번 이상 되풀이되는 동일한 문자를 의미하는 것으로 이러한 연속하는 반복 문자만을 압축의 대상으로 삼는 RLC 압축 방식은 일반적인 문서 화일에 대해서는 그리 효율적이지 못한 알고리즘이었다. 이러한 문제를 개선한 것이 VLC(Variable Length Coding) 방식이다.

대표적인 VLC 방식으로 알려진 허프만 코딩(Huffman Coding)은 셰논과 페노의 알고리즘을 모태로 탄생한 효과적인 통계적 압축 알고리즘이다. 셰논과 페노의 알고리즘은 출현 빈도가 높은 문자에는 적은 비트를, 낮은 문자에는 많은 비트를 할당해 압축 코드를 구성하는 알고리즘으로 허프만 코딩은 이러한 셰논과 페노의 코딩 방식 중 압축 코드를 위한 이진 트리를 구성하는 방법을 개선해 보다 효과적인 압축을 수행하도록 한 압축 방법이다.

2.1.2 DBC(Dictionary Based Coding) 방식[4], [3]

최근에 많이 이용되는 사전적 방식은 통계적 방식과 달리 문자의 빈도에 상관없이 압축을 수행하게 되는데 입력된 텍스트의 각 문자나 구를 분절하고 이를 일정한 형식으로 변환하여 압축을 수행한다. 따라서 사전적 방식에서 압축 코드는 각 문자 단위로 구성되는 것이 아니라 연속적인 문자 스트링을 단위로 정해진 이러한 사전적 방법은 압축 시간과 압축 효율의 우수성으로 인해 현재 통계적 방식보다 널리 사용되고 있다.

사전적 압축 방법은 사전이 만들어지는 시점에 따라 정적 사전(Static Dictionary)과 동적 사전(Adaptive Dictionary) 방식, 그리고 이 두 가지 사전의 특성을 결합한 준 동적 사전(Semiadaptive Dictionary) 방식으로 구분할 수 있다[3].

(1) LZ77

지브와 렘벨이 만든 LZ77은 동적 사전을 이용한 대표적인 압축 알고리즘이다. LZ77은 입력된 구나 문자를 사전에 수록하고 동시에 압축 코드로 변환하여 저장한다.

변환된 압축 코드는 토른이라고도 불리는데 크게 두 부분으로 구성된다. 첫째, 문자인지 압축 코드인지를 구분하는데 필요한 1비트와 둘째, 사전으로 인덱스 하기 위한 코드부가 그것이다. 사전인지 압축 코드인지를 구분하는 인식부는 사전적 방식이 압축을 수행할 때 변환된 압축 코드 보다 작은 크기를 지닌 문자나 구가 압축 코드로의 변환 없이 저장될 수 있도록 하기 위해 존재한다. 코드부는 사전으로의 인덱스를 나타내는 부분으로서 사전의 크기와 깊은 관련이 있는데 사전의 크기에 따라 그 크기가 결정된다. 결국 압축화일은 사전에 수록된 구나 문자를 인덱스하는 코드로 축약되는 것이다.

LZ77은 이 사전을 관리하는데 있어서 두 가지 문제점을 지닌다. 첫째 사전이 꽉 찬 상태에서 새로운 단어가 사전에 수록될 경우, 가장 먼저 사전에 수록된 단어가 삭제된다는 점이다. 이를 가리켜 슬라이딩 윈도우(Sliding Window)기법이라고 부르는데 이러한 방식은 데이터 처리 방식 중 큐와 같이 FIFO(First In First Out) 방식으로 사전이 관리되기 때문이다. 따라서 특정한 문자나 구가 사전의 크기보다 크거나 같은 간격을 두고 반복되어 입력된다면 이미 사전에 수록되어 있던 구나 문자를 밀어내어 압축을 할 수 없게 된다. 이것은 LZ77이 가장 최근에 읽어 들인 데이터를 기초로 사전이 작성되기 때문에 일어나는 현상으로 압축하려는 데이터의 양이 커질수록 이러한 문제가 심각하게 대두된다.

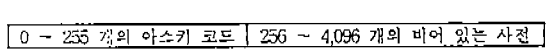
또 하나의 문제는 LZ77의 압축 코드가 표현할 수 있는 크기가 한정되어 있기 때문에 발생하는 문제로 길이가 긴 문자열을 간단하게 압축할 수 없다는 점이다. 즉, 위에서 설명한 압축 코드는 일정한 형식으로 표현하도록 되어 만일 그 길이가 지정된 길이로 표현할 수 없을 정도로 길다면 여러 차례에 걸쳐 나누어 한 단어를 표현해야 하므로 이는 가장 원시적인 압축 알고리즘이라 부르는 RLC(Run Length Coding)방식보다 효율적인 면에서 뒤지게 된다. 이러한 문제를 해결하기 위해서 사전의 구성을 새롭게 시도한 것이 LZ78이다.

(2) LZ78과 LZW

LZ78은 LZ77의 사전 수록 방식을 개선한 것으로 LZ77과 다른 두 가지 차이점을 지닌다. 첫째 일단 사전에 들어 있는 구나 문자는 추가하지 않는다. LZ77이 입력되는 데이터에 따라 무차별적으로 그 내용을 사전에 수록하였다면 LZ78은 이미 수록된 사전을 참조하여 사전에 없는 단어나 구만을 수록하도록 조절함으로써 사전의 저장 용량을 극대화 할 수 있다. 둘째로 압축 코드가 '사전으로의 인덱스+다음 문자'의 형식으로 구성된다. 이러한 두 가지 차이점은 사전의 구성이 다중 트리(multi tree)의 형태로 유지, 관리되기 때문이다. 즉, 입력된 문자나 구를 이미 수록된 사전과 비교하여 가장 유사한 구나 문자의 자식 노드에 새로운 구나 문자의 상이한 부분을 수록하는 방식이다.

LZ78은 문자열의 길이를 저장하지 않는다는 점 때문에 LZ77보다 일반적으로 나은 압축률을 구가 할 수 있다. 그러나 LZ78도 개선되어야 할 몇 가지 점이 발견되기 시작했다. 첫째 인덱스가 0인 경우(즉, 사전에 등록되지 않은 단어의 경우) 발생하는 오버 헤드를 최소화하여야 한다는 점과 둘째 압축 코드에서 다음 문자(즉, 부모노드의 아래에 있는 자식노드를 표현할 때 발생하는 문자)를 제외시킬 수 있다면 압축률의 보다 나은 성과를 가져올 수 있다는 점이다. 이러한 두 가지 문제점을 개선한 방식이 LZW 알고리즘이다.

LZW는 이미 정의된 사전을 가지고 출발한다. (그림 1)와 같이 LZW 사전의 앞부분 256바이트는 아스키 코드(ASCII Code)로 정의되어 LZ78에서와 같이 인덱스가 0인 코드를 생성하지 않는다.



(그림 1) 사전의 크기가 2^{12}인 LZW의 사전 구조
(Fig. 1) Dictionary Structure of of 2^{12} for LZW

하나의 문자가 입력되면 압축기는 우선 사전을 검색하여 같은 문자를 찾는다. 같은 문자가 발견되면 사전을 따라 일치하는 단어를 검색하게 된다. 만일 일치하는 단어가 없으면 일치하는 문자까지의 노드 아래에 새로운 문자를 삽입하여 인덱스를 부여하고 일치하는 단어가 있다면 상이한 문자가 나타날 때까지 계속 추적하여 위의 과정을 반복하게 된다. 이와 같은 방법으로 LZW는 LZ78과 같은 다중트리를 구성하게 된다.

2.2 한글에 대한 압축 기법

한글 압축 알고리즘에 대한 자료 조사 결과 한국어로 구성된 텍스트 화일을 사전적 방식을 이용해 압축하는 알고리즘의 개발은 드러난 연구가 없는 상태이며, 허프만 코딩을 이용해 한글을 압축하는 방식만이 연구 보고된 바 있다. 따라서 본 연구에서는 허프만 방식을 이용한 한글 압축의 과정을 살펴보고 이에 대한 문제점을 제기하고자 한다.

2.2.1 허프만 코딩 방식을 이용한 한글 압축기[6]

허프만 코딩 방식으로 구현한 한글 압축기는 조합형 한글에 대해 허프만 방식을 이용해 압축을 구현한 것으로 14개의 자음과 10개 모음의 조합으로 각각 초성, 중성, 종성이라는 3성의 조직적 규칙을 갖는 한글의 특성을 이용하여 복자음과 복모음을 포함해 초성에 올 수 있는 19개의 자음 집합, 중성에 올 수 있는 21개의 모음 집합, 그리고 종성에 올 수 있는 27개의 자음 집합을 구성하고 이를 빈도순으로 정렬하여 허프만 트리를 구성해 코드를 부여하였다.

허프만 코딩 방식을 이용한 압축 방식은 결정적으로 크게 두 가지 문제점을 가지고 있다. 첫째, 각 문자의 빈도를 저장하고 있어야 데이터의 복구가 가능하다는 점이다. 빈도의 저장이란 각각의 소스 화일에 따라 이루어져야 하는데 이는 압축의 효과를 저해하는 요소로 작용되기 쉽다. 크기가 비교적 작은 소스 화일의 경우 자칫 압축이 아닌 팽창의 효과를 가져올 수도 있기 때문이다. 둘째, 텍스트 화일의 경우 중복되는 일정한 문자 스트링 패턴을 효과적으로 압축할 수 없다. 즉, 구문적인 특성으로 일어나는 일정한 문자 스트링 패턴을 이용한 압축이 불가능하다. 텍스트 화일은 특히, 한국어에 있어서는 어휘적인 특성과 문법적인 특성으로 하나의 화일에 동일한 스트링이 중복 등장하는 경우가 자주 발생하는데 허프만 코딩 방식으로는 이러한 점에 대응할 수 있는 효과적인 방법이 없다. 따라서 본 연구에서는 허프만 코딩 방식에서 보여지는 문제점을 과감히 탈피하고 보다 효과적인 텍스트 화일의 압축을 수행하고자 DBC 방식에 한글의 특성을 고려한 새로운 압축 기법을 제안하고자 한다.

3. 한국어 압축

한국어를 실용적 입장이라는 측면에서 고찰할 때 중국어의 속성은 고립어, 영어의 특징은 굴절어로 요약되듯 한국어의 특징은 첨가어라는 속성으로 요약할 수 있다. 첨가어는 일정한 어근에 접사 또는 일정한 어간에 어미가 붙어서 단어를 이룬다. 때문에 한국어는 파생어가 많고 용언의 활용이 활발하며 합성에 의한 조어가 발달하였다[15].

3.1 한글의 형태적 특성

한국어 단어의 구조를 언어 체계상 뜻을 가지는 최소한의 단위인 '어휘 형태소'와 '문법 형태소'로 구분하여 비교한다면 복합어와 신조어, 외래어로 구성된 어휘 형태소의 집합은 문법 형태소에 비교할 수 없을 정도로 방대한 크기의 집합이다. 이에 비해 문법 형태소는 그 수가 매우 적으며 특히, 문법 형태소 중 조사와 어말어미의 수는 유한한 것으로 알려져 있다.[7]

자립성이 있는 말의 밑에 붙어 그 말과 다른 말과의 관계를 표시하는 관계언을 조사, 용언 활용형의 맨뒤에 와서 그 활용을 완성시키는 어미를 어말 어미라 정의하고 이러한 조사와 어말 어미의 상대적 출현 빈도를 조사해 본다면 우선 2개 이상의 조사가 결합한 형태를 포함한 통합형 조사의 경우 상대적 출현 빈도가 평균 상위 9개의 조사가 전체 조사의 70%를 차지하고 있고 상위 20개, 32개, 69개의 통합형 조사가 각각 90%, 95%, 99%를 차지하고 있으며, 통합형 어말 어미의 경우 평균 상위 10개의 어말어미가 전체 어말 어미의 70%를 차지하고 상위 33개, 54개, 117개가 각각 90%, 95%, 99%를 차지하고 있음이 연구 발표되었다[2].

따라서 한국어로 구성된 일반적인 텍스트 화일에는

극히 일부 조사와 어말어미의 출현 빈도가 높은 것으로 단정할 수 있으며 한국어 텍스트 화일의 압축에 있어서 이러한 조사와 어말어미를 이용한 압축 사전을 구축한다면 효과적인 결과를 가져오리라는 예측을 전제로 본 연구에서는 일반적인 한글 텍스트 화일에서 상대적 출현 빈도가 높은 61개의 조사 및 어미와 빈칸(blank), 마침표, 쉼표로 구성된 조사 및 어미 사전을 구축하게 되었다.

3.2 조사 및 어미 사전을 결합한 HLZ77

LZ77 알고리즘은 문자 단위로 데이터를 처리하던 허프만 방식을 탈피하여 여러 개의 문자열을 하나의 압축 코드로 대치하는 알고리즘이다. 이처럼 효과적인 알고리즘에 한글의 특성을 반영한 조사 및 어미 사전을 LZ77에 결합시켜 한글 압축의 효과를 높인 방식이 HLZ77이다.

HLZ77은 입력된 소스 화일을 대상으로 우선 분절을 실시한다. 이러한 분절은 사전으로의 등록을 이루기 위한 것으로 띄어쓰기(space)와 '\r', '\n' 단위로 실시된다. 일단 분절이 이루어지면 소스 화일은 '토큰(token)'이라고도 불리는 압축 코드로 변화되며 동시에 압축 사전의 구성이 함께 이루어진다. LZ77을 비롯한 모든 사전적 압축 방식은 입력된 소스 화일을 압축 코드로 변환하는 작업을 거치게 되는데 이때 소스 화일의 구성과는 다른 형식으로 보다 정밀하게 변환되는 것이다. 따라서 이러한 과정 중에 불필요한 공백의 낭비나 중복의 비효율성을 최소화시키는 것이 '압축'이라고 말할 수 있다.

HLZ77의 압축 과정을 알고리즘으로 표현하면 (그림 2)와 같다.

```
HLZ77 (infile, outfile)
{
  while((word = getword(infile))!=EOF)
  {
   split the word into '어근부분' & '조사 및 어미 부분'
         by using '조사 및 어미 사전'
   if ('어근 부분' is found in '동적 사전')
    output a token in 형식 2 또는 3 ;
   else
    output each character of '어근부분' in 형식 4
          by using '고정 사전'
   if ('조사 및 어미 부분' is found in '조사 및 어미 사전')
    output a token in 형식 1
```

(그림 2) HLZ77 알고리즘
(Fig. 2) HLZ77 Algorithm

HLZ77은 다음과 같은 과정을 거쳐 압축을 실시한다.

① 입력된 텍스트 화일을 '띄어쓰기(blank)'와 '\r', '\n' 단위로 읽어 들인다.

② 분절된 단어를 읽어 가며 조사 및 어미 사전과 비교하여 사전 내의 엔트리(entry)와 동일한 스트링이 있는가를 확인하고 동일한 조사나 어미가 발견되면 다시 분절하여 임시 저장한다.

③ 조사나 어미가 분절된 단어를 다시 읽어 들여 동적 사전과 비교하고 동적 사전에 등록된 단어인가를 확인한다. 만일 동적 사전에 등록된 단어라면 동적 사전의 주소를 참조하여 압축 코드를 작성한다.

④ 동적 사전에 등록되지 않은 단어임이 확인되면 고정 사전을 참조하여 문자 단위로 압축 코드를 작성하고 이를 동적 사전에 등록시킨다.

⑤ 분절되었던 조사나 어미를 조사 및 어미 사전을 참조하여 압축 코드로 작성한다.

⑥ ②의 과정부터 반복한다.

⑦ 화일의 끝을 알리는 EOF가 발견되면 작업을 종료한다.

HLZ77 알고리즘에서 사용하는 압축 사전 및 압축 코드의 형식에 대해서는 각각 3.2.1절과 3.2.2절에서 설명한다.

3.2.1 압축 사전

HLZ77에서 사용하는 압축 사전은 3가지이며 그 기능을 살펴보면 다음과 같다.

(1) 조사 및 어미 사전

이미 설명한 바와 같이 한글의 형태적 특징인 조사와 어말어미는 한국어로 구성된 문장에 있어서 상대적으로 높은 출현 빈도를 보이고 있다. 따라서 본 연구에서는 일부 통합형 조사와 통합형 어말어미를 선정하여 조사 및 어미 사전을 구축하고 이를 한글 텍스트 화일의 압축에 활용하기로 하였다.

조사 및 어미 사전의 크기는 압축 코드의 장단에 따른 압축률의 상관관계 속에서 결정되어야 한다. 보다 많은 조사와 어말어미를 사전으로 등록시킨다면 보다 좋은 압축 효율을 얻으리라 쉽게 생각할 수 있으나 토큰의 길이가 길어짐에 따른 압축률의 저하 또한 무시할 수 없는 것이다.

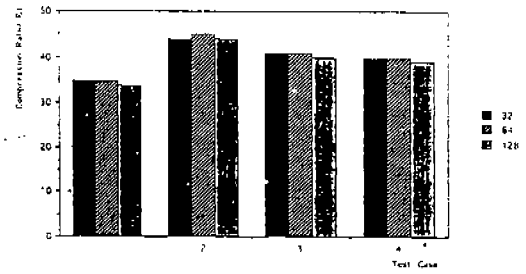
〈표 1〉 조사 및 어미 사전으로 구축된 통합형 조사 및 어말어미

〈Table 1〉 Josa/Eomi Dictionary for Compression

번호	엔트리	번호	엔트리	번호	엔트리	번호	엔트리
1	이라고	17	어서	33	을	49	만
2	어서는	18	으며	34	의	50	께
3	에서는	19	는데	35	를	51	며
4	이다.	20	어라	36	에	52	여
5	니다.	21	어야	37	이	53	어
6	지을	22	면서	38	는	54	기
7	거나	23	지만	39	은	55	게
8	으로	24	다는	40	가	56	지
9	하고	25	다고	41	로	57	면
10	에서	26	라고	42	과	58	던
11	보다	27	도록	43	와	59	아
12	에게	28	라는	44	고	60	려
13	이나	29	처럼	45	든	61	데
14	에는	30	대로	46	도	62	·
15	부터	31	요.	47	나	63	,
16	까지	32	다.	48	라	64	blank

본 연구에서는 (그림 3)와 같이 조사 및 어미 사전의 크기에 따른 압축률 변화를 고찰한 결과 총 64개(2^6)의 크기를 가지는 사전이 가장 적합한 압축률을 보이고 있음을 밝혀 내고 사전을 〈표 1〉과 같이 통합형 조사 33개와 통합형 어말어미 28개, 그리고 문장의 구조상 출현 빈도가 극히 높은 문장부호 마침표와 쉼표, 빈칸을 포함하여 총 64(2^6)개의 스트링으로 구성하였다.

조사 및 어미 사전에 있어서 또 한가지 주지할 사실은 한글로 구성된 문장에 있어서 모든 조사와 어말어미는 어절의 끝에 위치한다는 사실을 바탕으로 사전으로 구축된 개별 조사와 어말어미의 뒤에 빈칸



(그림 3) 조사 및 어미 사전의 크기에 따른 압축률 비교

(Fig. 3) Compression Ratio Graph for Different Josa/Eomi Dictionary Sizes

(blank)을 첨가하여 이를 하나의 단위로 취급하고 있다는 점이다. 또한 텍스트 화일에서 빈칸으로 처리되는 부분도 데이터의 구조상 개별 단위의 스트링으로 자리하고 있다는 사실에 기인하여 빈칸이라 하더라도 중복 등장하는 스트링의 출현을 최소화하기 위해 blank를 조사 및 어미 사전에 포함시켰다.

(2) 동적 사전(Dynamic Dictionary)

동적 사전이란 압축과 동시에 생성되어 압축의 완료와 동시에 소멸하는 압축 사전으로서 압축 효율과 긴밀한 연관을 가지고 있다. 일단 소스 화일이 입력되면 압축기는 blank와 \r, \n 단위로 분절을 실시하고, 입력된 음절을 조사 및 어미 사전과 비교하여 분절된 단어에 등록된 조사나 어미가 있는가를 확인한다. 이러한 확인 작업이 끝나면 나머지 단어에 대한 동적 사전의 확인 작업을 수행하게 되는데 만일 동적 사전에 입력된 단어가 등록되어 있지 않다면 글자 단위로 고정 사전의 주소를 참조하여 압축 코드를 작성하고 이 새로운 단어를 동적 사전에 등록하게 된다.

이렇게 등록된 단어는 계속되는 압축 과정 중에 다시 등장하는 같은 스트링에 대해 효과적인 역할을 수행하게 된다. 즉, 완성형 한글 한 문자를 특별한 조작없이 표현하는 경우는 16비트, 본 연구를 통해 구축된 고정 사전을 참조하여 문자를 압축 코드로 변환하는 경우는 인식부를 포함하여 14비트가 소요되는데

〈표 2〉 동적 사전의 크기에 따른 압축률

〈Table 2〉 Compression Ratios for Different Dynamic Dictionary Sizes

사전의 크기	5bit	6bit	7bit	8bit	9bit	10bit	11bit	12bit
평균 압축률	28%	29.5%	30.5%	31.5%	32%	31%	30%	29%

만일 같은 문자가 동적 사전에 등록되어 있다면 10비트만으로도 표현할 수 있는 것이다.

동적 사전의 크기 역시 압축 효율과 상관 관계 속에 있다. 만일 보다 많은 새로운 단어의 등록을 위해 사전의 크기를 크게 한다면 사전의 어드레스로 구성되는 압축 코드 주소부의 길이가 또한 커지게 되므로 전체적인 압축 코드의 확장으로 압축률은 떨어지게 될 것이다. 또한 압축 코드를 가능한 작게 유도하기 위해 사전의 크기를 제한한다면 사전은 쉽게 포화 상태를 이루게 된다.

사전의 포화상태는 새로운 문제를 발생시키기도 한다. LZ77에서와 마찬가지로 HLZ77에서도 동적 사전의 관리 방법은 밀어내기 방식(Sliding Window)으로 유지된다. 이것은 LZ77의 단점으로 지적되기도 하는 문제인데 사전이 포화상태를 이루면 가장 먼저 입력되었던 데이터를 삭제하는 방식으로 새로운 데이터를 저장하는 데이터 관리 기법이다. 따라서 사전의 크기는 〈표 2〉와 같이 압축 효율과의 상호관계 속에서 결정되어야 한다. 사전의 크기와 압축 효율의 상관관계를 실험한 결과에 의하면 사전의 크기는 512개($2^9=2^8\times2$)의 주소를 가지는 것이 가장 높은 압축 효율을 보이고 있는 것으로 드러났다. 따라서 본 연구에서는 2^8개의 주소를 저장할 수 있는 2개의 동적 사전을 구성하여 압축 효율을 배가시키고 있다.

(3) 고정 사전(Fixed Dictionary)

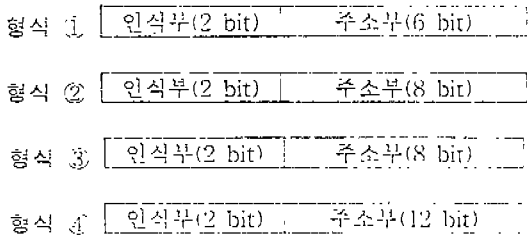
고정 사전이란 완성형 한글로 지정된 2,350자와 아스키 코드 256개, 그리고 화일의 끝을 알리는 EOF로 구성된 사전으로 2^{12}(4.096)개의 문자 코드를 저장할 수 있는 크기로 구성되어 있는 사전이다.

HLZ77에서는 동적 사전에 등록되지 않은 문자(처음으로 등장한 문자)가 이 고정 사전을 참조하여 압축 코드로 변환되고, 일단 고정 사전을 통해 참조된 문자는 동적 사전으로 등록되기 때문에 동일한 문자 스트링이 두 번 이상 고정 사전을 참조하는 경우는 없게 된다.

3.2.2 압축 코드

HLZ77의 압축 코드는 네 가지 형태로 구분할 수 있다.

(그림 6)에 있어서 ①번의 토큰은 조사 및 어미 사전을 참조하는 압축 코드로 인식부를 포함해 8비트로 구성되어 있으며 64개의 조사와 어말어미로 구성된 조사 및 어미 사전을 참조하게 된다. 인식부의 크기가 2비트로 구성된 이유는 전체적으로 압축 코드의 형식이 3가지로 구분되기 때문이다. 다시 말한다면 각기 다른 사전을 참조하는 3가지 형식의 토큰을 변별하기 위해서는 최소한 2비트 이상의 인식부가 필요하다는 것이다.



(그림 4) HLZ77의 압축 코드 형식
(Fig. 4) Compression Code Form for HLZ77

형식 ②번과 ③번 형태의 압축 코드는 동적인 사전을 참조하기 위해 구성된 토큰으로 2비트의 인식부와 8비트의 주소부로 구성되어 있다. 동적인 사전의 크기는 8비트(256개의 어드레스를 가지는 사전)로 2개의 동적 사전을 모두 사용한다면 총 512개의 새로운 단어를 사전에 등록할 수 있다.

형식 ④번의 압축 코드는 고정 사전을 참조하는 토큰으로 인식부를 포함해 총 14비트의 크기를 가지는 압축 코드이다. 고정 사전(Fixed Dictionary)이란 3.2.1 절에서 설명한 바와 같이 2,350자의 완성형 한글과 256개의 아스키 코드(ASCII Code), 그리고 화일의 끝을 알리는 EOF로 구성된 사전으로 한글과 영문을 비롯한 기타 문자의 표기를 인덱스로 대신하게 하는 역할을 하고 있다.

3.3 조사 및 어미 사전을 결합한 HLZW

HLZW는 LZ78의 문제점을 보다 개선한 LZW 알고리즘을 기본 골격으로 탄생한 한글 전용 압축 알고리즘으로 아스키 코드(ASCII Code)로 정의된 사전을 가지고 압축을 수행하는 LZW의 알고리즘에 한글의 형태적 특징인 조사 및 어미 사전을 결합한 형태로

구성되어 있다.

HLZW는 다음과 같은 과정으로 압축을 실시한다.

①입력된 텍스트 화일을 띄어쓰기, \n, \r 단위로 분절하되 연속된 blank는 하나의 단위로 취급 한다.

②분절된 단어를 읽어 가며 조사 및 어미 사전과 비교, 동일한 스트링이 있는가를 살핀 후 등 록된 조사나 어미가 발견되면 다시 분절하여 임시 저장한다.

③조사나 어미의 분절이 완료되면 조사나 어미를 떼어 낸 나머지 부분을 다시 읽어 들여 준 동적 사전과 비교하고, 사전에 등록된 문자라면 다음 문자를 계속하여 읽어 들여 입력된 단어 와 동일한 스트링을 찾는다. 단, 읽어 들이는 단위는 1바이트 단위로 실시하며, ②의 과정을 통해 분절된 부분을 만나면 사전과의 비교 작업을 중지한다.

④사전에 등록된 내용과 동일한 스트링은 사전의

```
HLZW (infile, outfile)
{
  while(word = getword(infile))!=EOF)
   {
   split the word into '어근 부분', '조사 및 어미 부분'
          by using '조사 및 어미 사전'
   compress '어근 부분' by using LZW algorithm
              /* make LZW tree */
   if('조사 및 어미 부분' is found in '조사 및 어미 사전')
    output token in 인식 1 by using '조사 및 어미 사전'
   }
}
```

(그림 5) HLZW 알고리즘
(Fig. 5) HLZW Algorithm

주소를 이용해 압축 코드를 완성하고 이어지 는 다음 스트링을 포함해 동적 사전에 등록시킨다.

⑤②에서 분절하여 임시 저장한 조사 및 어미 부분은 조사 및 어미 사전의 주소를 이용하여 압축 코드를 완성한다.

⑥②의 과정부터 반복한다.

⑦화일의 끝을 알리는 EOF가 발견되면 작업을 종료한다.

3.3.1 압축 사전

HLZW에서 사용하는 압축 사전은 2가지이다.

(1) 조사 및 어미 사전

HLZW에 사용된 조사 및 어미 사전은 〈표 1〉에 나와 있는 HLZ77에 사용된 조사 및 어미 사전과 동일하다. 한글의 형태적 특징인 조사와 어말어미 중에서 가장 높은 출현 빈도를 보이는 일부 통합형 조사와 어말어미 61개, 빈칸 그리고 마침표와 쉼표를 하나의 조사 및 어미 사전으로 구축하여 압축에 활용한다.

(2) 준 동적 사전(Semiadaptive Dynamic Dictionary)

준 동적 사전이란 동적 사전과 정적 사전의 특성을 함께 소유한 형태의 사전 방식으로 두 가지 형식의 압축 방식이 갖는 장점이 적절하게 조화된 효과적인 알고리즘이다. 여기서 말하는 두 가지 장점이란 첫째, 정적인 사전이 갖는 장점으로 압축의 대상이 되는 데이터의 구성을 예측하여 미리 사전을 구성하였다는 점이다. 이는 차별적인 특징을 지닌 어떠한 화일이라 할지라도 그것이 텍스트 화일이라면 개별적인 문자의 위치에서 관찰할 때 256개 아스키 코드 중의 일부로 판단할 수 있다는 점에서 출발한다. 둘째, 동적인 사전이 갖는 장점으로 살펴본다면 사전의 구조를 다중 트리(multi tree)로 구성하여 연속하는 문자를 이미 정의된 문자의 노드 아래에 기록하고 주소를 부여하여 사전의 활용도를 높인다는 점이다.

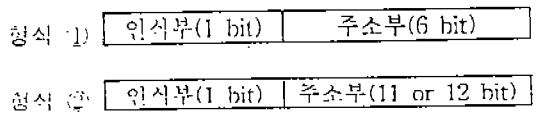
HLZW에서도 이처럼 동적인 사전 일부에 아스키 코드를 저장하여 입력되는 새로운 문자의 사전 구성에 효율적으로 이용하고 있다. 사전의 주소 0부터 255까지를 256개의 아스키 코드로 예약하고 새로 입력되는 문자는 이 아스키 코드를 참조하여 압축 코드로 변환한다. 연속되는 문자는 선행한 문자의 노드 아래에 자식 노드로 표기되는데 이 순서에 따라 주소가 부여된다. 그러므로 동적 사전의 주소는 256부터라고 할 수 있는 것이다.

LZW는 토큰의 길이가 인식 비트 없이 12비트로 구성되었다. 이것은 압축 사전이 이미 정의된 256개의 아스키 코드를 포함하여 4,096개의 주소를 저장할 수 있는 사전을 의미하는 것이다. 그러나 HLZW의 토큰은 준 동적 사전과 조사 및 어미 사전을 구분하기 위한 인식 비트를 보유하고 있어야 한다. 따라서

본 연구에서는 보다 높은 압축률을 가지는 HLZW 알고리즘을 구현하기 위해 준 동적 사전의 크기를 11비트와 12비트로 달리하는 두 가지 알고리즘을 모두 구현하여 압축률을 비교하였다.

3.3.2 압축 코드

HLZW는 두 가지 형식의 압축 코드로 구성되어 있다. (그림 7)에서와 같이 ①의 토큰은 HLZ77과 마찬가지로 조사 및 어미 사전을 참조하는 압축 코드로 1개의 인식부와 6개의 주소부로 구성되어 있다. HLZW는 HLZ77과 달리 사전의 구성이 두 가지로 제한된다. 따라서 어떠한 사전을 참조하는가를 나타내는 인식부는 하나의 비트로 변별이 가능하게 된다.



(그림 6) HLZW의 압축 코드 형식
(Fig. 6) Compression Code Form for HLZW

②의 형식은 HLZW 고유의 준 동적 사전을 참조하는 압축 코드로서 사전의 크기에 따라 총 12비트 혹은, 13비트로 구성되며 사전으로의 인덱스를 가리키는 주소부는 11비트 혹은, 12비트로 구성되어 있다. 본 연구에서는 사전의 항목 수가 2,048(2^{11})개인 HLZW 기법을 HLZW11이라 하고, 사전의 항목수가 4,096(2^{12})인 HLZW 기법을 HLZW12라고 명명한다.

4. 실험 및 비교

4.1 구현 환경 및 가정

본 연구에서는 한국어의 압축에 있어서 문법 형태소인 조사와 어말어미를 이용한 고정 사전이 텍스트 화일의 전체적인 압축에 어떠한 영향을 미치는가를 알아보기 위하여 조사 및 어말어미의 상대적인 출현 빈도가 작은 한글 텍스트 화일과 범용 텍스트 화일을 대상으로 다음과 같은 실험을 실시하였다.

실험에 사용된 프로그램은 Borland C^{++} 4.0으로 구현된 한글 전용 압축 프로그램으로 동적 사전 방식으로 대표되는 LZ77과 여기에 한글의 특성을 반영한 HLZ77, 준 동적 사전으로 설명된 LZW와 여기에 한글의 특성을 반영한 HLZW11, HLZW12이다.

실험에 이용된 텍스트 화일은 조사 및 어말어미의 효과적인 압축이 전체 화일의 압축률에 미치는 영향을 알아보기 위해 조사 및 어말어미의 출현 빈도가 작은 화일과 일반적인 문장으로 구성된 텍스트 화일을 대상으로 삼았는데 조사 및 어미의 출현 빈도가 작은 텍스트 화일로는 크기가 다른 시 5편이 사용되었으며 일반적인 텍스트 화일로는 각각 크기를 달리한 9개의 성서와 9개의 과학 서적 화일을 사용하였다. 실험 환경은 IBM PC 486-66 Mhz였다. 입력 화일에 대한 가정으로서는 한글에 대한 보다 정확한 압축률을 측정하기 위해 입력된 모든 소스 데이터를 완성형 한글로 제한하였으며 가능한 한글 텍스트 화일의 압축에 영향을 미치는 아라비아 숫자와 영문 그리고 기타 특수 문자들은 입력을 제한하였다.

4.2 LZ77, HLZ77, LZW, HLZW의 압축률 비교

(그림 8)와 (그림 9)은 〈표 3〉과 〈표 4〉의 결과를 그래프로 나타낸 것이다.

이 실험에 따르면 화일의 크기가 작은 텍스트 화일의 압축률은 LZ77과 HLZ77이 LZW와 HLZW에 비해 우수한 것으로 나타난다. 이것은 크기가 작은 텍스트 화일인 경우 크기가 큰 텍스트 화일에 비해 상대적으로 압축 사전의 활용도가 떨어지게 되는데 그 이유가 있다. 따라서 압축 코드의 길이가 상대적으로 큰 LZW와 HLZW에 비해 LZ77과 HLZ77이 우수한 압축률을 보이게 된다. 데이터 "book"의 경우 LZ77과 HLZ77이 약 1만 바이트 크기의 텍스트 화일에 이르기까지 우수한 압축 효율을 보이다가 LZW와 HLZW의 압축률에 뒤지게 되는데 1만 바이트 이상의 크기를 지닌 텍스트 화일이 입력되면 더 이상 작은 길이의 압축 코드로 얻게 되는 압축 효율 보다 사전의 활용으로 인해 얻게 되는 압축 효율이 높아지는 것을 의미하는 것이다. 이러한 현상 은 다른 텍스트 화일의 압축에서도 보여지는 동일한 현상으로 단지 어느 정도의 크기에서 이러한 '압축률의 변화가 오는가'라는 차이만이 있을 뿐이다. 이것은 텍스트 화일의 특성을 이용하면 보다 명확하게 구분할 수 있다.

(그림 10)은 시에 대한 압축률을 실험한 것으로 압축의 대상이 되는 소스 데이터의 특성에 따라 동적

〈표 3〉 데이터 "bible"에 대한 각 알고리즘의 압축률
〈Table 3〉 Compression Ratios for Bible Data

단위 %

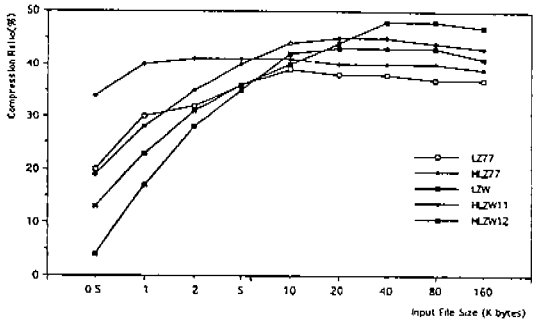
화일명	SIZE(k)	LZ77	HLZ77	LZW	HLZW11	HLZW12
bible1	0.5	20	34	4	19	13
bible2	1	30	40	17	28	23
bible3	2	32	41	28	35	31
bible4	5	36	41	35	40	36
bible5	10	39	41	42	44	40
bible6	20	38	40	43	45	44
bible7	40	38	40	43	45	48
bible8	80	37	40	43	44	48
bible9	160	37	39	41	43	47

〈표 4〉 데이터 "book"에 대한 각 알고리즘의 압축률
〈Table 4〉 Compression Ratios for Book Data

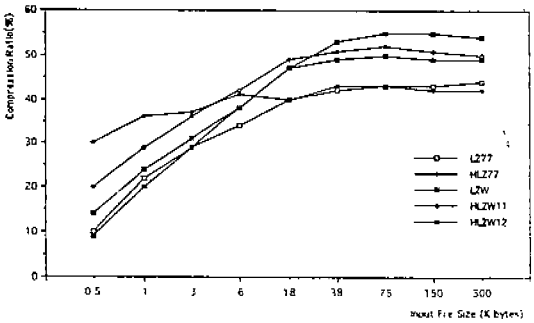
단위 %

화일명	SIZE(k)	LZ77	HLZ77	LZW	HLZW11	HLZW12
book1	0.5	10	30	9	20	14
book2	1	22	36	20	29	24
book3	3	29	37	29	36	31
book4	6	34	41	38	42	38
book5	18	40	40	47	49	47
book6	38	42	43	49	51	53
book7	75	43	43	50	52	55
book8	150	43	42	49	51	55
book9	300	44	42	49	50	54

사전 방식과 준 동적 사전 방식이 어떠한 압축률의 차이를 보이는가를 설명해 준다. 이 실험에서는 LZW와 HLZW에 비해 LZ77과 HLZ77이 보다 우수한 압축률을 보이고 있다. (그림 11)의 실험에서도 (그림 8)과 (그림 9)에서 보여준 결과와 같이 조사나 어미 사전의 참조를 통해 얻는 효과 보다 동일한 문자와 단어 스트링의 패턴에 의해 얻어지는 효율이 높다는 것을 의미한다. 따라서 단순히 문자나 단어의 반복되는 패턴을 동적인 사전으로 저장하는 LZ77과 HLZ77이 LZW와 HLZW 보다 높은 압축 효율을 보이는 것이



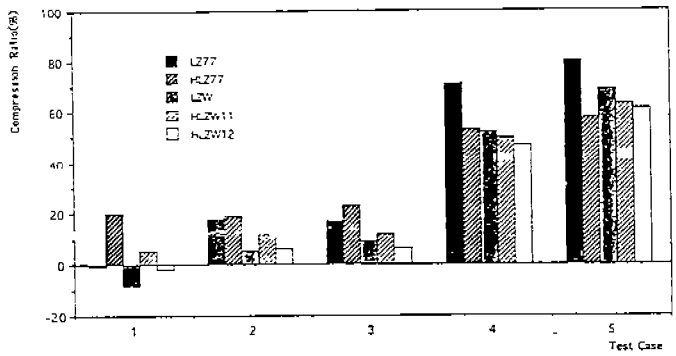
(그림 7) 데이터 "bible"에 대한 각 알고리즘의 압축률 비교
(Fig. 7) Compression Ratio Graph for Bible Data



(그림 8) 데이터 "book"에 대한 각 알고리즘의 압축률 비교
(Fig. 8) Compression Ratio Graph for Book Data

다. 특별히 (그림 10)의 그림 중 CASE 4와 5는 동일한 내용의 시를 여러 차례 반복하여 전체 화일의 크기를 인위적으로 늘린 것으로 동적 사전의 충분한 활용을 통해 얻어지는 압축 효율의 결과를 보여주고 있다. 또한 CASE 1에서 LZ77과 LZW가 압축이 아닌 팽창이 일어났다는 사실은 크기가 작은 화일에 있어서 동일한 문자나 단어의 스트링 패턴이 출현하지 않는 경우 압축이 아닌 팽창이라는 결과를 초래할 수 있다는 기존의 알고리즘의 문제점을 증명한 것이며 HLZ77과 HLZW11의 압축이 성공적으로 이루어 진 것은 작은 크기의 화일이라도 한국어로 구성된 화일에는 조사나 어말어미가 등장할 확률이 높다는 것을 의미하는 것이다.

한편, HLZW12가 팽창의 결과를 가져 온 것은 조사나 어말어미의 압축으로 얻어진 압축률에 비해 압축 코드의 길이가 증가한 것에 의해 손실된 압축률이 더욱 크다는 반증이다.



(그림 9) 크기가 다른 "poem"에 대한 압축률 비교
(Fig. 9) Compression Ratio Graph for Poem Data

〈표 5〉 데이터 "poem"에 대한 각 알고리즘의 압축률
〈Table 5〉 Compression Ratios for Poem Data

단위 %

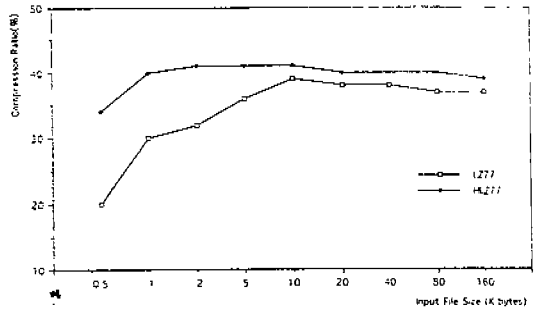
화일명	SIZE(byte)	LZ77	HLZ77	LZW	HLZW11	HLZW12
poem1	382	−1	20	−8	5	−2
poem2	494	18	19	5	12	6
poem3	907	17	23	9	12	6
poem4	10907	71	53	52	50	47
poem5	100031	80	58	69	63	61

4.3 LZ77과 HLZ77의 압축률 비교

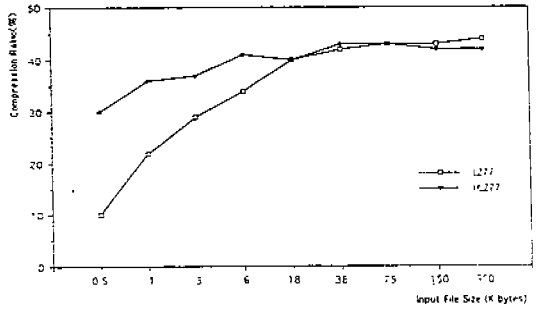
LZ77과 HLZ77의 압축률의 비교는(그림 11)와 (그림 12)에 나타나 있다.

(그림 11)와 (그림 12)에 의하면 화일의 크기가 비교적 작은 경우에는 HLZ77이 LZ77보다 훨씬 우수한 압축률을 보이고 있으나 화일의 크기가 커질수록 그 차이가 좁아지는 현상을 보이고 있다. 이는 화일의 크기가 작은 경우에 화일에 등장하는 조사와 어말어미의 효과적인 압축이 화일 전체의 압축률에 영향을 크게 미치는 것으로 볼 수 있으며 화일의 크기가 커질수록 동일한 단어와 동일한 구의 등장으로 LZ77의 압축률이 향상하는 것으로 해석할 수 있다.

데이터 "bible"의 경우((그림 11) 참조) LZ77과 HLZ77은 약 1만 바이트에 이르면 더 이상 압축률의 변화 없이 거의 같은 수준으로 유지되는데 이는 화일의 특성에 의한 것으로 보인다. 다른 자료인 (그림 12)는 과학



(그림 10) 데이터 "bible"에 대한 LZ77과 HLZ77의 압축률 비교
(Fig. 10) Comparison of LZ77 and HLZ77 Based on Bible Data



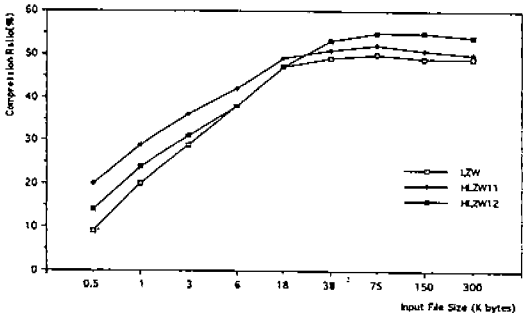
(그림 11) 데이터 "book"에 대한 LZ77과 HLZ77의 압축률 비교
(Fig. 11) Comparison of LZ77 and HLZ77 Based on Book Data

서적을 소스 데이터로 사용한 경우인데 이 경우 LZ77과 HLZ77은 화일의 크기가 약 1만 바이트에 이르면 어느 것이 우수한 압축률을 보이고 있다고 단정하기 어렵게 압축률의 혼선을 빚고 있다. 즉, 입력된 텍스트 화일의 구성에 있어 문학적 요소가 충만하여 조사와 어말어미의 사용이 많다면 HLZ77이 우수한 압축 효율을 보이고 논문과 과학 서적과 같이 보다 단정적이며 간결한 문체로 구성되어 조사와 어말어미의 사용이 적다면 LZ77이 우수한 압축률을 보이는 것으로 해석할 수 있다. 그러나 전체적인 압축 효율을 판단한다면 조사와 어말어미의 효과적인 압축을 수행하는 HLZ77이 보다 우수한 압축률을 보이고 있다고 말할 수 있다.

4.4 LZW와 HLZW의 압축률 비교

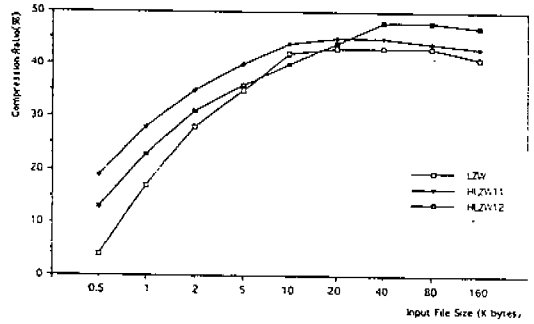
(그림 13)와 (그림 14)는 LZW와 HLZW의 압축률 비교로서 한글의 특성을 반영한 HLZW는 LZW에 비해 얼마만큼의 압축 효과를 얻는가를 증명한 실험이다. 특히 이 그림은 같은 동적 사전을 2^{11}과 2^{12}의 크기로 각각 구현한 HLZW의 압축률을 동시에 보여주고 있는데 데이터 "book"의 경우 HLZW11이 LZW 보다 항상 우수한 것으로 나타났으며 HLZW12는 LZW와 같거나 보다 좋은 압축률을 보이고 있다.

HLZW 알고리즘의 우수성은 다른 그림인 (그림 14)에서도 유사한 결과로 나타나고 있다. (그림 14)에 의하면 HLZW11은 LZW에 비해 항상 높은 압축률을 보이고 HLZW12는 일부 화일에 있어 LZW 보다 낮은 압축을 보이고 있으나 전체적으로 우수한 압축률을 나타내고 있다. 이러한 결과는 한글의 특성을 반영한 조사 및 어미 사전이 일반적인 텍스트 화일에 있어서 효과적인 역할을 수행하며 텍스트 화일의 압축에 상당한 영향을 미치는 것으로 해석할 수 있다.



(그림 12) 데이터 "book"에 대한 LZW와 HLZW의 압축률 비교

(Fig. 12) Comparison of LZW and HLZW Based on Book Data



(그림 13) 데이터 "bible"에 대한 LZW와 HLZW의 압축률 비교

(Fig. 13) Comparison of LZW and HLZW Based on Bible Data

5. 결 론

본 논문에서는 일반적인 한글 문서에 대해 효과적인 압축을 얻기 위해 한글의 형태적 특성인 조사 및 어말어미를 이용하는 효과적인 한글 문서 압축 알고리즘들을 제시하였다. 또한, 제시된 한글 문서 압축 알고리즘들의 성능을 기존의 알고리즘들의 성능과 비교 분석하여 본 연구에서 제안한 한글 문서 압축 알고리즘이 기존의 압축 알고리즘 보다 높은 압축률을 제공할 수 있음을 증명하였다.

본 논문에서 제시한 압축 방식은 사전적 압축 방식의 대표적인 알고리즘인 LZ77과 LZW를 기본 골격으로 여기에 조사 및 어미 사전이라는 한국어의 특징을 결합시켜 기존의 압축 알고리즘 보다 효율적인 결과를 얻어냄으로써 텍스트 화일에 있어서 한국어의 형태적 특징을 이용함이 갖는 효율성이 점검되었다. 보다 구체적인 효율성을 거론한다면 일반적인 텍스트 화일에 있어서 한글의 형태적 특성을 반영한 HLZ77은 압축 대상 화일의 크기가 증가함에 따라 LZ77과 거의 유사한 압축률의 기울기를 가지며 LZ77보다 평균 5%의 더 우수한 압축 효율을 보이고 있다. 또한 HLZW는 압축 대상 화일의 크기에 따라 LZW와 압

축률의 고저를 반복하고 있으나 대체적으로 HLZW11은 6%, HLZW12는 4%가 우수한 것으로 나타나고 있다. 그러나 압축 효율을 결정하는 근본적인 요인은 대상 화일이 '어떠한 형식의 글인가'라는 압축 대상의 구성 양식에 있음이 본 연구를 통해 함께 밝혀짐으로써 소스 데이터 화일에 따라 압축률은 진폭을 크게 할 것으로 예측된다. 그럼에도 불구하고 한국어로 작성된 텍스트 화일을 압축 대상으로 하는 한글 전용 압축 프로그램의 개발에 있어 한국어의 형태적 특징인 조사와 어말어미를 이용하는 방식은 상당한 압축 효율을 가져올 수 있으리라 예견된다.

그러나 이번 연구에 의해 구현된 압축기는 순수한 한글 텍스트만을 그 대상으로 하고 있어 영문을 비롯한 다른 특수 문자가 혼용된 화일에 대해서는 그 효과가 반감하고 있어 이에 대한 보다 효과적인 압축 방법과 압축 시간을 단축하기 위한 데이터 구조의 개선 등이 추가로 연구되어져야 할 과제로 남아 있다.

참 고 문 헌

[1] 박명신, "압축 프로그램을 만들자-압축이 뭐길래" 마이크로소프트웨어, 1993. 4, 202-210쪽.

[2] 강승식, "상대적 출현 빈도를 이용한 조사/어미 사전의 구성", 제 7회 한글 및 한국어 정보처리 학술대회 논문집, 1995. 10, 188-194쪽.

[3] Timothy C. Bell, John G. Cleary, and Ian H. Witten, "Text Compression", Prentice-Hall, 1990.

[4] 박명신, "압축 프로그램을 만들자-DBC 알고리즘 Ⅰ, LZ77과 LZSS" 마이크로소프트웨어, 1993. 9, 236-243쪽.

[5] 박명신, "압축 프로그램을 만들자-DBC 알고리즘 Ⅱ, LZ78과 LZW" 마이크로소프트웨어, 1993. 10, 266-285쪽.

[6] 조기현, "한글의 원리를 이용한 화일 압축 기법 제안", 석사 학위논문, 숭실대학교, 1991.

[7] 강승식, "음절의 정보와 복수어 단위 정보를 이용한 한국어 형태소 분석", 박사 학위논문, 서울대학교, 1993.

[8] Tom R. Halfhill, "데이터 압축 안전한가", COMPUTER MAGAZINE, 1994. 3, 138-151쪽.

[9] Gerard Salton, "Automatic Text Processing", Addison-Wesley, 1988.

[10] 나동렬, "한국어 파싱에 관한 고찰" 한국정보과학회지, Vol. 12, No. 8, 1994, 33-46쪽.

[11] 박명신, "압축 프로그램을 만들자-세논-페노의 변형, 허프만 코딩" 마이크로소프트웨어, 1993. 5, 184-199쪽.

[12] 박명신, "압축 프로그램을 만들자-산술코딩" 마이크로소프트웨어, 1993. 6, 220-225쪽.

[13] 박명신, "압축 프로그램을 만들자-유용한 압축 라이브러리", 마이크로소프트웨어, 1993. 7, 208-213쪽.

[14] 박명신, "압축 프로그램을 만들자-데이터 압축 라이브러리 제작", 마이크로소프트웨어, 1993. 11, 230-243쪽.

[15] 남태현, "실무자를 위한 새한글 맞춤법", 연암출판사, 1992.

이 기 석

1991년 목원대학교 무역학과 졸업(경제학사)

1994년~현재 천주교 인천교구 홍보부 재직

1996년 인하대학교 산업기술대학원 정보공학과 졸업(공학석사)

관심분야: 자연어 처리, 한국어 정보처리

김 유 성

1986년 인하대학교 전자계산학과 졸업(이학사)

1988년 한국과학기술원 전산학과 졸업(공학석사)

1992년 한국과학기술원 정보 및 통신공학과 졸업(공학박사)

1992년~현재 인하대학교 전자계산학과 조교수

관심분야: 멀티미디어 데이타베이스 시스템, 분산 데이터베이스 시스템, 트랜잭션 처리 시스템