대용량 한글 텍스트 검색 엔진 HMG의 구현

박미란† · 나연묵†

요 약

본 논문에서는 영문 텍스트 검색 엔진인 MG(Managing Gigabytes) 시스템과 한글 형태소 분석기 HAM (Hangul Analysis Module)을 이용하여 기가바이트 크기의 텍스트 데이타 처리가 가능한 한글 텍스트 검색 엔진 HMG(Hangul MG)를 구현하였다. 한글 처리를 위해 KSC 5601 완성형 코드를 사용하여 데이타베이스 구축 단계와 질의 처리 단계에서 사용하였다. HMG의 개발을 위해 MG 시스템의 렉시칼 분석기와 파서, 인덱스 구성 모듈을 수정하였다. HMG 시스템의 유용성을 보이기 위해 웹에서 한글 소설을 검색할 수 있도록 하는 NOD(Novel On Demand) 시스템을 구현하였다. HMG 시스템은 한글이 포함된 대규모 전문 검색 시스템의 구축에 활용될 수 있다.

Implementation of Very Large Hangul Text Retrieval Engine HMG

Miran Park† and Yunmook Nah†

ABSTRACT

In this paper, we implement a gigabyte Hangul text retrieval engine HMG(Hangul MG) which is based on the English text retrieval engine MG(Managing Gigabytes) and the Hangul lexical analyzer HAM(Hangul Analysis Module). To support Hangul information, we use the KSC 5601 code in the database construction and query processing stages. The lexical analyzer, parser, and index construction module of the MG system are modified to support Hangul information. To show the usefulness of HMG system, we implemented a NOD(Novel On Demand) system supporting the retrieval of Hangul novels on the WWW. The proposed system HMG can be utilized in the construction of massive full-text information retrieval systems supporting Hangul.

1. 서 론

텍스트 형태의 정보의 급속한 증가에 따라 컴퓨터를 이용하여 대용량의 문서를 검색할 수 있는 정보 검색 시스템에 관한 연구가 계속되고 있다. 최근에 와서는 정보의 양이 기하급수적으로 증가하여 이러한 시스템의 개발이 더욱 절실히 요구되고 있다. 전문 정보 검색(full-text information retrieval)이란 문서에 포함된 모든 단어에 대해 색인을 하고 색인에 의해 저장된 문서의 하부구조에서 사용자의 질의에 적합한 문서를 찾아내는 검색으로 구성되어 있다. 전문을 색인하여 검색하는 시스템은 색인어가 많고 따라서 발생하는 저장 용량의 증가와 질의 처리시 생기는 검색 공간의 방대함으로 인해 성능이 감소되는 문제점을 가진다. 이런 문제는 텍스트 데이타베이스를 압축함으로써 해결할 수 있다[1].

텍스트에 대한 인덱싱 기법으로는 전문 텍스트 스캐닝(full-text scanning), 역 화일(inverted file), 시그니처 화일(signature file), 클러스터링(clustering) 기법 등이 있는데 정보 검색 시스템의 하부구조는 주로 역 화일 방법과 시그니처 화일 방법이 사용되고 있다[2]. 시그니처 화일의 경우 시그니처가 차지하는 공간은 텍스트 크기의 10~15%로 공간 오버헤드가

† 단국대학교 컴퓨터공학과

적고 시그니쳐의 재구성이 필요없이 삽입 작업이 용이하다는 장점이 있는 반면 블록 시그니쳐가 우연히 검색 단어 시그니쳐와 일치하는 탈락 착오(false drop) 현상이 발생한다[2].

최근의 대부분의 상용화된 정보 검색 시스템은 역화일 방법을 기반으로 하고있다. 이러한 정보 검색 시스템의 예로 1960년 미국 국립 의학 도서관의 MEDLINE 시스템, IBM사의 STAIRS, 로히드사의 DIALOG 시스템, InQuery, Verity 등이 있다. 외국의 정보 검색 시스템을 한글화한 예로는 freeWAIS가 있다[6]. 역 화일을 이용한 방법은 키워드를 포함한 문서를 정확히 검색한다는 장점이 있는 반면 인덱스 자체의 크기가 텍스트 자체 크기의 10%~100%의 부가공간이 필요하며 텍스트 데이타의 내용 변경시 인덱스도 변경해야 한다는 단점이 있다[2].

실제 데이타베이스 구축시 랭킹 질의는 용어의 수가 많고 따라서 긴 역 리스트들이 스캔되어야 한다. 공간의 효율성을 위해 역 리스트를 압축하여 저장하면 반대로 압축된 역 리스트의 처리비용이 증가한다. 압축되지 않으면 역 화일은 쉽게 커지며 텍스트보다도 커진다. 역 화일의 압축은 크기를 80%이상 감소시키지만 디코딩은 처리시간에 있어 중요한 오버헤드를 발생시킨다[3].

MG 시스템은 Australian Research Council과 the Collaborative Information Technology Research Institute의 지원으로 개발되었는데 세부적인 부분은 Melbourne, Canterbury, Calgary, Waikato대학과 그 외의 많은 대학에서 연구에 참여하였다. MG 시스템은 대용량 데이터 처리가 가능한 전문 정보 검색 공개 소프트웨어이다. MG 시스템은 현재 Newzealand Digital Library에 사용되고 있으며, Victoria University의 Web 검색 시스템으로도 사용되고 있다[5]. MG 시스템은 블록된 사전의 사용으로 메모리 크기를 감소시키고, 압축기법의 사용으로 디스크 용량을 감소시키며, 스킵의 이용으로 처리 시간을 감소시킨다[3,4]. 이러한 장점은 대량의 데이타 처리가 필요한 디지탈 라이브러리나 온라인 텍스트 저장소 구축에 적합하다.

한글 문서에서 키워드를 추출하려면 각 단어들을 형태소 분석에 의하여 키워드로서 가치가 있는 명사들을 추출해야 한다. 이 때 형태소 분석 기법의 활용 정도에 따라 색인어휘집을 이용하는 방법, 기능어휘집을 이용하는 방법, 그리고 형태소 분석기를 이용하는 방법이 있다. 그런데 색인어휘집과 기능어휘집을 이용하는 방법은 자동색인 기능으로서 한계가 있기 때문에 최근에는 대부분 형태소 분석을 이용하는 방법을 취하고 있다[9].

본 논문에서는 형태소 분석을 기반으로한 한성대학교 정보전산학부에서 개발한 HAM(Hangul Analysis Module) 4.0과 MG 1.2 시스템을 이용하여 한글이 지원되며 대량의 데이타 처리가 가능한 정보 검색 시스템을 구현하였다.

HMG 시스템의 특징은 다음과 같다. 첫째, 한글 처리시 행망 표준인 2바이트 완성형 코드를 한영 혼용문서의 데이타베이스 구축 단계와 질의 처리의 어휘분석 단계에서 사용한다. 둘째, 한글 색인시 한글 형태소 분석기인 HAM을 이용하여 용어의 색인여부와 복합명사인 경우 정확한 유사도 측정을 위해 분해된 형태로 색인을 한다. 셋째, HMG 시스템의 WWW을 통한 서비스를 제공을 위해 CGI(Common Gateway Interface) 기술을 이용한다. HMG 시스템은 한영 정보가 혼합된 대규모의 전문 검색 시스템의 구축에 활용될 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 텍스트 인덱스의 효율적인 압축기법과 영문 MG 시스템의 전체적인 구조에 대하여 설명한다. 3장은 한영 혼용문서 처리를 위한 한글처리, 한글 스태밍, 한글 질의 처리에 대해 기술한다. 4장에서는 HMG 시스템을 이용한 응용 시스템 NOD(Novel On Demand) 시스템의 구현에 대해 설명하고, 마지막으로 5장에서는 결론과 앞으로의 연구 방향을 제시한다.

2. 텍스트 인덱스 압축 기술과 MG 시스템

2.1 역 화일 인덱스를 위한 압축 기법

대규모 컬렉션에서 질의 용어에 대한 역 리스트는 수 천 개의 문서 식별자(document identifier)를 포함한다. 문서 식별자는 정수값을 가지며 정수값으로 저장된 역 리스트는 다음의 압축 기법을 사용하여 저장공간을 감소시킬 수 있다. 정수에 대한 효율적인 압축 방법의 하나로 Elias 코드 계열이 있다[3,4,7]. Elias 코드는 다양한 비트로 정수를 표현하며 Elias 코드의 연속된 열은 유일하게 디코딩 가능하다. 주요 Elias 코드에는 unary 코드, gamma 코드, delta 코드, Go-

lomb 코드가 있다. unary 코드는 10진수 1,000,000을 위해 1,000,000 비트가 필요하며 gamma 코드는 39 비트, delta 코드는 29비트가 필요하다.

역 리스트의 특성을 이용하면 더 많은 압축을 제공해 줄 수 있다. 역 리스트에 저장된 수치의 대부분은 문서 번호와 위치이며 오름차순으로 배열되어 있다. 같은 종류 값들의 인접한 번호들 사이의 차분값만을 저장하면 더 적은 비트로 표현 할 수 있다. 예를 들어, 'uranium'이라는 단어가 3번 문서에서 한 번 나타나며 위치는 61이고, 10번 문서에서 2번 나타나며 위치는 14와 106라고 하자. 역 리스트가 문서 번호, 빈도수, 단어의 위치로 구성되어 있을 때 단어 'uranium'에 대한 역 리스트

3 : 1(61),
10 : 2(14, 106), ...

는

3 : 1(61),
7 : 2(14, 92), ...

로 표현할 수 있다. 문서 번호의 중요성을 고려할 때 차분에 의한 형태의 열의 표현에는 Golomb 코드가 최적이다. 3 G바이트 데이타베이스에서 Golomb 코드를 이용한 역 인덱스는 190 M바이트를 요구한다. delta 코드도 사용될 수 있는데 압축 효율이 좀 떨어진다. 빈도수를 위해 gamma 코드를, 단어의 위치를 위해 delta 코드를 사용하면 인덱스가 데이타 크기의 22%를 차지한다[7]. 압축된 인덱스는 저장공간 뿐만 아니라 한번의 I/O시 보다 많은 양의 데이타를 포함하므로 I/O 시간을 단축하여 검색 시간을 줄인다.

2.2 MG 시스템

MG 시스템은 기존의 역 화일 구조 시스템의 부가 공간의 크기로 인한 문제점을 효율적인 압축을 통해 해결하였고 압축된 데이타의 처리비용은 스킵을 이용하여 디코딩함으로써 약간의 처리비용을 감소시켰다. 또한 블록기법("3-in-4" 코딩)의 사용으로 대규모 사전 전체를 메모리에 로드하지 않고 일부 사전만 메모리로 로드하여 적은 메모리로도 처리가 가능할 수 있도록 하였다. 또한 압축된 인덱스는 저장공간 뿐만 아니라 한번의 I/O시 보다 많은 양의 데이타를 가지므로 I/O시간을 단축하여 검색 시간을 줄이고 있다. 그러므로, 메모리 크기, 디스크 용량, 처리시간을 동시에 줄여준다고 할 수 있다. MG 시스템의 이러한 특성으로 인하여 기존에 개발된 검색 엔진에 비해 월등하게 우수한 성능을 보이고 있다.

이 절에서는 MG 시스템의 특징과 텍스트, 인덱싱, 사전의 구성과 효율적인 압축 기법에 대해 설명한다. MG 시스템은 크게 2가지 모듈로 구성되는데, 텍스트와 인덱스의 압축 저장 모듈과 저장된 내용의 검색 모듈로 나눌 수 있다.

(1) 텍스트 압축 절차

원시 텍스트를 읽고 구동 프로그램인 mg_pass의 매개변수 T1과 T2에 따라 통계 데이타를 생성하고 텍스트를 압축한다. 텍스트 화일의 압축에는 산술 코딩이나 허프만 코딩기법이 사용된다. 텍스트의 압축 저장 절차는 다음과 같다.

① mg_pass -T1 : 원시 텍스트를 읽어서 각 토큰의 빈도수에 대한 통계화일(.text.stats)을 생성한다.
② mg_compression_dict : 통계 화일의 빈도수를 이용해서 단어의 사전 화일(.text.dict)을 생성한다.
③ mg_pass -T2 : 문서번호를 문서 오프셋(document offset)으로 할당한 매핑 화일(.text.idx)을 생성한다. 텍스트 화일을 인코딩하여 압축 화일(.text)을 생성한다. 텍스트의 첫 번째 패스를 통해 얻어진 통계 데이타를 기반으로 사전을 압축한다.
④ mg_fast_comp_dict : 보조 사전(.text.dict.aux)이 있을 경우 텍스트 사전과 결합해 준다.

(2) 인덱스 구조

MG 시스템과 다른 소규모 시스템과의 차이는 사전의 사용 여부에 있다. MG 시스템에서는 서로 상이한 목적을 위해 3가지 사전이 사용된다. 사전은 인덱스된 단어의 리스트와 단어의 특성을 유지한다. 사전은 역 화일과 분리되어 유지되는데 이는 대다수의 연산이 사전의 로딩은 요구하지만 역 화일의 로딩을 함께 요구하지는 않기 때문이다. 단어의 특성은 <f_t, F_t, I_t>로 구성된다.

- f_t : 용어가 나타나는 문서들의 수
- F_t : 컬렉션을 통한 용어의 발생 빈도수
- I_t : 역 화일 포인터

사전은 주어진 단어에 대한 빠른 검색과 각 리스트에 대한 매칭 문서의 빠른 처리를 위해 구성되어야 한다. 그러므로 가장 기본적인 경우 사전은 단어들의

배열처럼 저장되고 각 리스트도 순서화된 문서번호들의 배열처럼 저장되어야 한다. 매핑 테이블도 배열처럼 저장되며 이것은 문서번호와 매칭 문서들의 매핑에 사용된다.

인덱스 생성과 압축시에는 원시 텍스트를 읽고 mg_pass 프로그램의 매개변수 I1과 I2에 따라 스태밍된 사전을 생성하고 역 화일을 생성한다. 인덱스 생성과 압축 절차의 세부 작업 내용은 다음과 같다.

① mg_pass -I1 : 완전 전단 코드 사전(Completely Front coded Dictionary)을 생성한다. 완전 전단 코드 사전(**.invf.dict**)은 순차적인 처리가 요구될 때 사용되며 다른 사전을 생성할 때와 문서 가중치를 만들 때 사용되는 기본적인 사전이다. 이 사전은 순차적인 처리에는 적합하지만 이진 탐색이 불가능하다는 단점을 지닌다. 완전 전단 코딩은 단어가 정렬되어 있을 때 접두어를 공유할수 있다는 점에서 시작되었다. 두 개의 정수를 각 단어와 함께 저장하는데 하나는 이전의 단어와 같은 접두어의 갯수를 나타내며 다른 하나는 남아있는 접미어의 갯수를 나타낸다. 예를 들어, 길이가 7인 단어 'jezebel'을 완전 전단 코딩하면 '3,4,ebel'이 되는데 이는 이전의 단어와 3개가 같고 4개가 다르며 다른 접미어는 'ebel'이라는 것을 의미한다.

② mg_perf_hash_build : 완전 해쉬 사전(Perfect Hash Dictionary)을 생성하는데 ①에서 만들어진 사전을 이용한다. 완전 해쉬 사전(**.invf.dict.hash**)은 사전 전체가 메모리로 액세스되어야 하며 빠르고 많은 액세스가 요구되는 작업에 적당하다. 역 화일 생성시 블록 전단 코드 사전은 액세스가 너무 느리고 또 컬렉션의 모든 용어의 액세스가 필요하기 때문에 완전 해쉬 사전을 사용한다. 따라서 최소 완전 해쉬 테이블을 램에 저장하여 사용한다.

③ mg_pass -I2 : 이 패스에서는 ②에서 만든 완전 해쉬 사전을 로드하고 원문 텍스트를 읽으며 색인어에 대한 역 화일(.invf)과 역 파일의 주소를 갖는 역 인덱스(.invf.idx)를 생성한다.

④ mg_invf_dict : 블록 전단 코드 사전(Blocked Front code Dictionary)을 생성한다. 블록 전단 코드 사전(**invf.dict.blocked**)은 질의 처리시 사용되는 사전이다. 이 사전은 임의접근이 필요하지만 완전 해쉬 사전의 속도가 요구되지 않은 경우 램에 사전의 일부 블록만 상주시킬 때 사용된다. 이 사전은 부분 "3-in-4" 전단 코딩을 사용하여 부분적으로 전단 코딩되어 있고 그 결과 임의 접근이 가능하다. 부분 "3-in-4" 전단 코딩은 완전 전단 코딩과 방법은 같으나 4개의 단어씩 분리해서 처리한다. 이 방법은 완전 전단 코딩이 이진 탐색이 불가능하기 때문에 질의 용어를 찾는데는 적합한 구조가 아니므로 4개의 단어 중 첫 번째 단어는 그대로 저장하고 나머지 3개 단어만 전단 코딩하여 이진 탐색을 수행 할 수 있도록 한다.

이러한 사전들은 MG의 인터프리터 mgquery에 의해 사용되며 임의 질의 용어를 찾는데 사용된다.

압축된 역 리스트에서는 임의 접근 포인트가 불가능하므로 압축된 리스트의 첫 비트에서만 디코딩을 개시할 수 있다. 이것은 모든 역 리스트가 처리되어야 함을 의미하는데 실제로 모든 <d, $f_{d,t}$> 포인터를 요구하는 것은 아니다. AND 연산자에 의한 결합 불리언 질의의 경우 현재의 역 리스트에서 맴버 검사를 위한 후보만 필요하다. 따라서 동기화 포인트, 즉 디코딩을 개시할 수 있는 추가적인 위치를 압축된 역 리스트에 제공하면 빠르게 평가가 수행될 수 있다. 예를 들어,

<5,1><8,1><12,2><13,3><15,1><18,1><23,2>, <28,1><29,1>...

의 경우 3 쌍씩 그룹지어 3개씩 스킵 가능하다.

≪5,a2≫<5,1,><3,1><4,2>**≪13,a3≫**<1,3><2,1><3,1>

≪23,a4≫<5,2><5,1><1,1>...

a2는 두 번째 스킵 포인터의 첫 비트의 어드레스이다. a3는 세 번째 스킵 포인터의 첫 비트 어드레스이다. 다시 <d, $f_{d,t}$>의 중복을 제거하면 다음과 같이 나타낼 수 있다 .

≪5,a2≫<1,><3,1><4,2>**≪8,a3-a2≫**<3><2,1><3,1>

≪10,a4-a3≫<2><5,1><1,1>...

첫 번째 스킵에서 두 번째 스킵의 a2 어드래스를 얻고 디코딩한다. 만약 문서 d1, d2에 이런 스킵이 제공되고 그 결과 평가를 원하는 문서 번호 d가 d1 ≤d<d2이고 d가 첫 번째 그룹에서 나타나면 첫 번째 그룹의 디코딩만이 필요하다. 만일 d2≤d 이면 두

번째 스킵은 세 번째에 위치를 찾고 두 번째 그룹에서 d가 놓여있는지를 결정한다. 이 경우 첫 번째 그룹은 디코딩되지 않으므로 약간의 처리 시간을 감소시킬 수 있다. 이 역 리스트의 압축에는 Golomb 코드가 가장 적합하다[3]. 그 이유는 이러한 코딩에서 역 리스트의 전체 그룹이 거의 같은 길이를 갖기 때문이다. MG에선 스킵 역 리스트를 "mg_invf_ rebuild"에 의해 생성할 수 있다.

(3) 질의 처리

MG 시스템은 불리언 질의, 랭킹 질의, 문서번호 검색을 제공한다. 문서 랭킹에 질의 용어(query term)들과 문서 용어(document term)사이의 유사도를 위한 데이타를 요구한다. 이 유사도 측정에 코사인 측정을 사용한다. 질의 처리시 가장 높은 점수의 문서가 사용자에게 보여진다. 불리언 질의는 AND(&), OR(|), NOT(!) 연산자를 지원한다. 랭킹 질의는 질의 용어 리스트로 나타낸다. 문서번호 검색은 질의 용어를 포함한 문서 번호만을 원할 때 사용한다.

3. HMG 시스템 구현

한글 질의를 처리하기 위해선 한글 사전을 구성해야 하고 한영 혼용문서에서 문서 저장시 이를 분리하여 각각에 해당하는 사전을 생성해야 한다. 질의 처리에서 사용할 역 인덱스 사전은 영문인 경우 Lovin의 알고리즘을 이용하여 생성하고 한글인 경우 한글 형태소 분석기를 이용하여 색인어를 추출하고 사전을 생성한다.

그림 1은 HMG 의 색인 과정을 나타낸 것이다. 영문인 경우 folding과 스태밍 단계를 거쳐 단어에 가중치를 부여하여 단어를 저장하고 한글인 경우 형태소 분석과 복합명사 분해모듈을 거쳐 단어에 가중치를 부여하고 데이타베이스에 저장한다.

3.1 한영 혼용문서의 저장

한글 처리를 위해 현행 표준 부호계인 KSC 5601 2 바이트 완성형 부호계를 사용한다. 각 바이트의 MSB(Most Significant Bit)는 한글임을 나타내기 위하여 1로 고정된다. 한글 음절은 제 1바이트의 값이 B0H에서 C8H까지 (24개 행), 제 2바이트의 값이 A1H에서 FEH까지 (94개 열)의 공간에 25×94 = 2,350

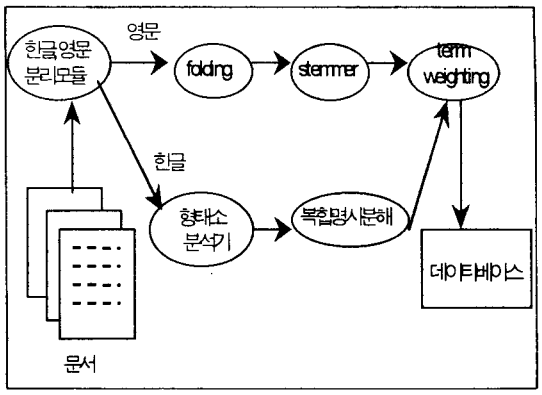


그림 1. HMG 색인 과정

자가 수용된다. 2350자는 한글 음절의 빈도에 의해 선정되었다.

색인어의 단위는 영어인 경우 어절과 단어의 단위가 같으므로 스테밍 기법에 의하여 원형을 복원하여 사용하지만 한국어와 같은 굴절어인 경우 원형 복원이 형태소 해석을 수행해야만 가능하다. HMG 시스템에선 한글 처리를 위해 아래와 같이 매크로를 정의하여 텍스트에서 영문단어, 한글단어, 비 단어를 추출한다.

영문인 경우는 ASCII 코드이면서 문자나 숫자인 문자코드를 뜻하며, 한글인 경우 첫번째 바이트가 0xb0~0xc8 사이의 값을 갖고 이어진 두번째 바이트가 0xa0~0xfe 사이의 값을 갖는다. 나머지는 비단어로 취급하였다. 다음은 원문 텍스트에서 단어를 추출하기 위한 매크로의 정의 이다 .

```
#define INAWORD(c) (isascii(c)) && isalnum(c))
#define INHANFIRST(c) ((c >= 0xa1) && (c<=0xff))
#define INFIRST(c) ((c >= 0xb0) && (c<= 0xc8))

#define INSECOND(c) ((c>=0xa0) && (c<= 0xfe))
#define MAXWORDLEN    15
#define MAXNUMERIC     4
```

영문인 경우 'INAWORD(c)'매크로를 이용하여 Word 버퍼에 영문워드에 해당하는 것만 추출 한 후 저장하고, Word[0]에 추출된 길이를 넣는다. s_in은 원문 포인터를 뜻하며 end는 원문의 끝 포인트이다.

```
#definePARSE_영문(Word버퍼,원문포인터,원문끝포인터)
do {
      while(원문의 문자포인터가 영문이고 끝이아님)
```

```
    {
        Word버퍼에 문자를 삽입;
        단어길이 증가;
        원문포인터 증가 ;
    } Word[0]에 단어길이 삽입;
}while(0)

#definePARSE_한글(Word버퍼, 원문포인터, 원문끝포인터)
do{
    while( 원문의 문자포인터가 한글의 첫바이트)
    {  원문 포인터 증가;
       if( 원문의 문자포인터가 한글의 두 번째바이트가
       아니면)  break;
       else  Word버퍼에 한글문자 삽입;
       단어길이 증가;
       원문 포인터 증가;
    } Word[0]에 단어길이 삽입 ;
}while(0)
```

'INFIRST(c)' 매크로를 이용하여 한글의 첫 바이트인지 체크하고 이어서 'INSECOND(c)'를 이용하여 두번째 바이트를 체크한다. 2바이트 단위로 버퍼 Word에 넣는다. Word[0]에 단어길이를 넣는다. 위의 매크로 정의문은 mg_pass의 매개변수를 이용한 역 화일 사전을 만드는데 사용된다. 역 화일 사전 생성 단계는 다음과 같다.

① mg_pass -T1 : 원시 텍스트를 읽어 각 토큰을 추출하는 단계에서 한글단어를 분리하고 분리된 단어의 빈도수에 대한 통계화일을 생성한다.
② mg_pass -I1 : 완전 전단 코드 사전을 생성하는 단계로 원시 텍스트를 읽어 한글단어를 분리하고 스태밍 단계를 거친후 색인 가능한 단어에 대한 사전을 생성한다.
③ mg_pass -T2 : ① 에서 얻어진 통계 데이타를 기반으로 사전을 압축하는데 사전 생성시 한글 단어는 분리된다.
④ mg_pass -I2 : 원문 텍스트를 읽고 한글 색인어를 추출한 후 색인어에 대한 역 화일과 역 화일의 주소를 갖는 역 인덱스를 생성한다.

3.2 한글 스태밍

색인어의 단위는 영어권인 경우 어절과 단어의 단위가 같으므로 스태밍 기법에 의하여 원형을 복원하여 사용하지만 한국어와 같은 굴절어인 경우 원형 복원이 형태소 해석에 의해 가능하다.

형태소 단위의 색인은 색인어의 종류가 어절단위의 색인어의 종류보다 수십에서 수만배 정도 작다는 장점을 지닌다. 단점으로는 형태소의 정확성 보장과 원래 문서가 가지고 있는 정보를 분해함으로써 생기는 손실을 들 수 있다.

HMG에서는 한글 스태머로 한성대에서 만든 공개 소프트웨어인 HAM(Hangul Analysis Module)을 이용하였다. HAM은 HAM의 글로벌 변수를 초기화하고 사전을 로드하는 모듈init_ham_index("./hdic/"), 형태소 분석 모듈 get_keyword_1(aSentence, & keywords), 사전 및 글로벌 변수를 종료하는 모듈 end_ham()으로 구성된다. HAM 4.0의 hangul.top 화일은 색인어로 추출되지 않도록 하고 싶은 명사, 즉 불용어(stopword) 및 특수 색인어를 화일에 등록하면 색인어 추출시 참조된다. 또 hangul.usr 사용자 정의 사전에 <단어, 품사열> 쌍으로 구성하여 사용할 수도 있다. 영문인 경우 HAM을 이용하지 않고 MG 1.2에서 사용한 Lovin의 알고리즘을 그대로 이용하였다. 이 알고리즘은 모든 문자를 folding하여 소문자로 한 다음 Lovin의 스태밍 알고리즘에 따라 다양한 접미사를 지워준다.

한글인 경우 파싱한 토큰을 형태소 분석 모듈을 이용하여 불용어인지 색인어인지를 판별하고 복합명사인 경우 분해된 명사를 색인어로 사용한다. 한글 정보 검색기에서 가장 심각한 문제는 복합명사와 띄어쓰기의 차이에 의한 것이었고 이를 분해된 형태로 사용하면 정확한 유사도 측정을 할수 있다. 형태소 분석 모듈은 역 화일 사전을 구축하는 부분과 질의 처리 부분에서 사용된다. HMG에서는 형태소 분석 결과에서 복합명사는 제거하고 1음절 명사는 포함시켜 색인어를 추출한다.

한국어 색인을 수행하는 프로시져 han_index의 매개 변수는 한글단어이다. 한글단어에 대해 get_key_word_1()을 이용하여 형태소 분석의 결과를 얻는다. 다음 분석된 결과 리스트에서 복합 명사는 제거하고 분해된 형태의 단어만 추출하여 매개변수로 사용한 버퍼에 길이와 분리된 단어를 저장하고 버퍼에 저장된 단어의 갯수를 리턴한다. 한국어_색인 모듈을 호출한 프로그램은 리턴 값을 체크하여 리턴된 값이 0이면 색인어가 존재하지 않으므로 다음 단어를 추출하고 이와 같은 루틴을 수행하게 된다. 리턴

값이 0이 아니면 색인어가 존재하므로 버퍼에서 단어를 추출하고 역 인덱스 사전에 쓴다.

그림 2는 한글 인덱싱 프로시져이다.

```
procedure han_index(var word:uchar , var wcount:int)
 var p:HAM_PUCHAR;
     nkeyword:int;
 begin
  형태소 분석을하는 get_keyword_1(word, &keywords)을
  호출하여 불용어인지 색인어인지 분석한 후, 복합명사이면
  분해된 결과가 keywords에 저장되고 갯수는 nkeyword로 리턴 ;
  p를 keywords의 시작 포인트에 위치;
  for i:=0 to nkeyword do
  begin
     if( 명사 또는 1음절명사) then /* 복합명사 제외 */
     begin
        word 버퍼에 분해된 길이와 결과 복사;
        wcount증가;  /*복합명사가 제거된 색인 갯수*/
     end;
     p를 keywords의 다음 위치로 이동;
   end;
 end;
```

그림 2. 한글 인덱싱 프로시져

HMG 시스템의 구조를 단순화한 형태로 나타내면 그림 3과 같이 사전, 역 리스트, 문서와 매핑 테이블로 구성된다. 이와 같은 구조는 질의에 대한 평가를 할 수 있고 사전은 주어진 단어를 빠르게 검색하며 각각의 리스트를 처리하여 결과에 매칭되는 문서를 제시할 수 있다.

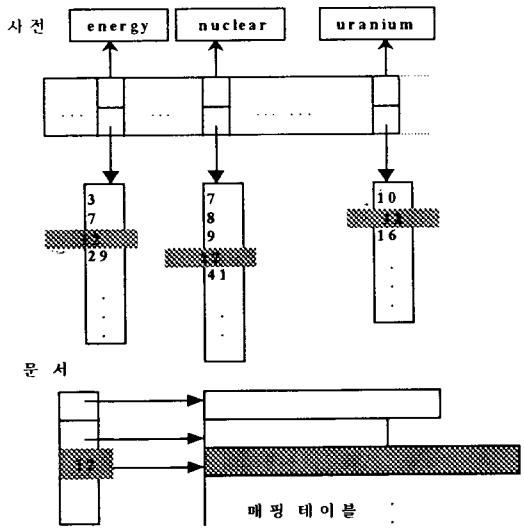


그림 3. 단순화된 인덱스와 데이타 구조

3.3 한글 질의 처리

HMG 시스템은 불리언 질의, 랭킹 질의를 제공한다. 질의에서 한글 지원을 위해선 한글 코드를 인식할 수 있도록 확장 되어야 한다.

불리언 질의는 구문 분석기 bool_parser.y의 어휘처리 부분인 query_lex함수에서 매크로를 이용하여 한글을 파싱한 후 한글 스태머를 이용하여 형태소를 분석한다. 한글 파싱을 위해 렉시칼 분석을 하는 루틴은 그림 4와 같다.

```
begin
  whitespace를 제거한다;
  if( *ptr가 영문)  then
  {    parse_영문( )이용 영문단어 추출 ;
       stemmer로 어근 추출;
  } else if( *ptr & 0xffffff값이 한글) then
  {   parse_한글( )이용 한글단어 추출;
       han_index()로 색인어 추출;
  } else  nonword 이면 ptr증가 ;
end;
```

그림 4. 한글 렉시칼 분석 루틴

불리언 용어 처리 루틴은 주요 작업 절차는 다음과 같다.

- 각 토큰은 whitespace에 의해 나누어진다.
- iscomplex함수를 호출하여 'LG정밀' 처럼 한글과 영문이 혼용된 단어인지 검사하고, 한글인 경우 한글 스태머를 호출하고, '사무자동화'와 같은 복합명사는 분해한다.
- bool_expand 함수를 호출하여 한영 혼용단어와 복합명사인 경우는 AND 연산자로 결합해 준다. 예를 들어, LG정밀은 'LG' & '정밀'로 사무자동화는 '사무' & '자동화'로 나타낸다.

키보드로부터 입력받은 한글은 첫 번째 바이트가 0xffffffb0~0xffffffc8이고 두 번째 바이트가 0xffffffa0~0xffffffffe가 된다. 따라서 입력된 한글 코드와 0x000000ff을 비트연산자 &로 각 비트를 마스킹하여 원래의 한글 코드값을 얻고 키보드 입력 관련 함수에서 이용한다. 예를 들어,

```
      0xffffffb0
  &   0x000000ff
  ------------
          0xb0
```

을 얻을 수 있다. AND연산자에 의한 불리언 질의 평가에 대한 절차는 다음과 같다.

① 질의 프로그램은 시작할때 블록 인덱스를 읽는다. 질의 용어 참조시 우선 디스크 블록 인덱스를 살펴보고 블록이 이미 램에 로드되었으면 거기에서 체크하고 만약 로드되지 않았으면 블록을 로드한다.
② 각 불리언 질의 용어 t에 대해
  a. 용어 t를 스태밍한다.
  b. 사전에서 용어를 검색한다 .
  c. 용어에 대한 역 화일 인덱스 주소를 읽는다.
③ 질의 용어 t에서 가장 작은 빈도수의 용어를 지정한다.
④ 상응하는 역 리스트를 읽고 C로 초기화한다. C는 후보자 리스트이다.
⑤ 남아있는 질의 용어를 위해
  a. 역 화일 엔트리 It를 읽는다.
  b. 각 문서 번호 d가 C에 속할 때
    만일 d가 역 리스트 It에 없으면 C에서 d를 삭제한다.
  c. 만약 C가 0이면 답이 없다는 것을 의미이다.
⑥ 문서 d가 C에 속할 때
  a. 문서 d의 어드레스를 조사한다.
  b. 문서 d를 검색하고 사용자에게 프리젠테이션 한다.

랭킹 질의는 query.ranked.c 프로그램에서 수행한다. 이 프로그램은 질의 용어 파싱과 SimilarityGet을 호출하는데 힙(heap)을 사용하며 힙의 가장 위의 값을 선택하여 사용자에게 보여준다. 한글 인식을 위해 질의 용어 파싱 단계에서 매크로를 이용하여 한글 단어를 추출한다.

3.4 한글 인덱싱과 질의 처리 예

본 절에서는 HMG 시스템의 인덱싱과 질의 처리 예를 살펴본다.

(1) 데이타베이스 구축 예

"사무자동화는 어떤 회사에서나 ....."
와 같은 데이타의 일부가 있을 때 데이타베이스 구축 절차는 다음과 같다.

① whitespace에 의해 토큰이 분리된다.
  사무자동화는
  어떤
  회사에서나 ...
② han_index 함수에 의해 색인여부가 판별되며 색인어 갯수가 리턴된다.
  색인어 갯수= han_index("사무자동화")로 복합명사는 제거되고 분해된 형태만 길이와 함께 word 버퍼로 리턴된다. 색인어 갯수는 2가 리턴된다.
  word버퍼는 | 4 | 사무 | 6 | 자동화 | 이다.
③ 색인어 갯수가 0이 아니면 word버퍼에서 각 단어를 추출하여 HASH 매크로를 이용해 해쉬 테이블 포인트를 얻고 해쉬 레코드에 저장한다.

(2) 한글 질의 처리 예

질의어: "사무자동화"
가 주어졌을 때 한글 질의 처리 예는 다음과 같다.
① 질의 라인에서 "사무자동화"를 추출한다
② han_index 함수를 호출하여 데이터베이스 구축 단계와 같은 결과 값을 얻는다.
③ 사전에서 "사무"와 "자동화"란 단어에 해당하는 역 리스트를 읽고 처리한다.
  a. 블록 사전에서 검색용어를 찾고 버퍼에 역 리스트를 읽는다.
  b. 매개변수에 의해 Golomb 디코딩을 계산한다.
  c. 버퍼 처리를 위해 문서번호의 차분에 의한 Golomb 디코딩 용어 빈도수에 의한 Gamma 디코딩을 한다
  d. '사무&자동화'에 대한 질의 평가를 한다.

3.5 HMG 구현 환경

본 논문에서 설계한 HMG는 SUN Sparc 20에서 구현하였다. 사용한 language는 C language이며 컴파일러는 GNU의 gcc를 사용하였다. Web Server는 SUN server를 사용하였고 브라우저는 Netscape 4.0과 Explorer 4.0을 이용하여 검색할수있다. OS는 SunOS 5.6이며 MG 1.2 시스템과 한글 형태소 분석기 HAM 4.0을 결합하여 구현하였다.

4. NOD 응용 시스템 개발

이 논문에서는 HMG의 유용성을 보이기 위해 인

터넷에서의 정보제공 서비스인 WWW(World Wide Web)를 통해 몇 개의 컬렉션을 검색할 수 있도록 소설 검색 시스템 NOD(Novel On Demand)를 구현하였다.

4.1 NOD 시스템 구조

WWW 구조를 이용하여 구성된 NOD 시스템 구조는 그림 5와 같다.

NOD 시스템의 구조는 다음과 같다.

- 데이타베이스 : MG의 저장구조에 따라 저장된 소설 데이타베이스를 나타낸다.
- IR 엔진: 정보 검색을 위한 엔진으로 HMG 시스템이다.
- 서버 : 브라우저에서 문서 검색을 위해 지정한 HTML 문서와 검색 후 브라우저에 결과를 내보내기 위한 CGI 프로그램이 놓이게 된다.
- 클라이언트: 검색 서비스를 요청하는 브라우저로 MS Explorer나 Netscape등의 프로그램을 뜻한다.

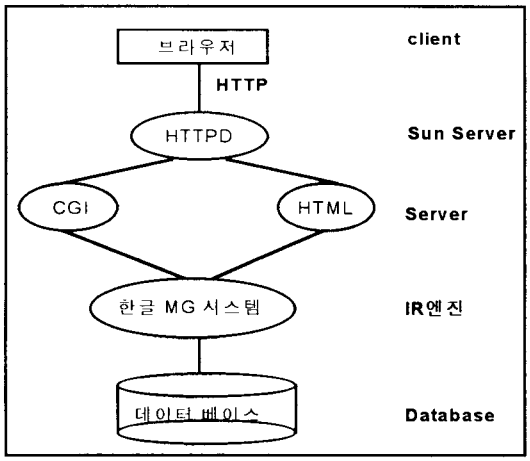


그림 5. NOD 시스템 구조

4.2 데이타베이스 구축

본 실험에서는 한국 단편 소설 50여편의 데이타를 4개의 컬렉션으로 구성하였다. 표 1은 NOD 시스템의 데이타 크기를 보이고 있다. HMG는 입력 데이타로 압축 텍스트 파일을 사용한다.

mg_invf_rebuild에 의해 스킵을 역 리스트에 삽입하면 추가적인 공간이 더해져 스킵하지 않은 역 파일

표 1. NOD 시스템의 데이타 크기 (단위: byte)

실 험 데이타	텍스트크기 (압축크기)	역파일크기 (스킵삽입)	완전전단코드사전 (블록전단코드사전)
data1	1430888 (734022)	53298 (59741)	42785 (60639)
data2	301199 (162225)	13390 (14025)	54904 (78092)
data3	347695 (185027)	13880 (14253)	51371 (73032)
data4	55665 (32310)	3130 (3303)	19223 (27216)

크기보다 더 많은 공간이 필요하다. 웹 환경에서 구현되었기 때문에 스킵된 역 리스트의 질의 처리 시간을 얻어내기 어려웠다. HMG는 블록 전단 코드 사전을 사용하는데 헤더와 블록 인덱스 데이터로 인해 완전 전단 코드 사전보다 많은 공간이 필요하다.

4.3 실행 결과

NOD를 실행시키려면 MG 환경변수를 다음과 같이 CGI 프로그램에 설정한다.

```
putenv("MGDATA=/user/mrpark/mgdata");
putenv("MGSAMPLE=/user/mrpark/SampleData");
```

MGDATA는 MG의 텍스트와 인덱스가 저장된 곳이고 MGSAMPLE은 원문이 저장된 곳인데 프로그램 실행시 환경 변수를 참조한다.

그림 6은 실험에 사용된 단편 소설 중의 일부를 나타내고 있는데 페이지별로 검색을 할 경우 각 페이지

53. 갈 매 기

- 이범선

파도 소리가 베개를 때린다.

좀처럼 잠이 오지 않는다. 여느 날 같으면 벌써 나갔을 전등이 그대로 들어와 있다. 아마 이 또 무슨 일이 생겼나 보다. 기쁜 일이나 그렇지 않으면 슬픈 일이.

섬 안은 그대로 한집안이다. 그러기 어느 집안에든지 잔치가 있거나 또는 애사가 생기면 이렇게 밤새도록 전등이 들어오는 것이다. 시장에서 생선 장사를 하는 상이군인이 새색시를 맞던 날도 그랬다. 읍장님의 어머니 진갑 날도 그랬다. 고아원에서 어린애가 죽던 날도 그랬고, 일전 파도가 세던 날 나갔던 어선 한 척이 돌아오지 않던 밤도 그랬다 ……

그림 6. 컬렉션에 사용된 데이타의 일부

의 끝을 나타내는 Ctrl+B를 페이지 별로 원문에 삽입하고 문단별로 검색을 원할 경우 Ctrl+C를 원문의 문단에 삽입한다.

그림 7은 브라우저에서 mg1.html을 실행했을때의 화면으로 원하는 컬렉션과 질의 타입을 선택하고 소설에서 찾고자하는 질의어를 입력하면 서버에서는 mgquery.cgi 프로그램이 질의 처리를 수행하고 결과를 브라우저에 리턴한다.

그림 7에서 랭킹 질의를 선택하고 질의어를 리스트처럼 스페이스를 이용하여 나열한 후 검색요청을 하면 질의어와 문서 사이의 유사도가 가장 높은 문서를 리턴한다. 그림 8은 랭킹 질의에 대한 결과 화면이다. 화면 하단에는 유사도값이 나타나고 유사도가 높은 문서 번호순으로 프리젠테이션 된다.

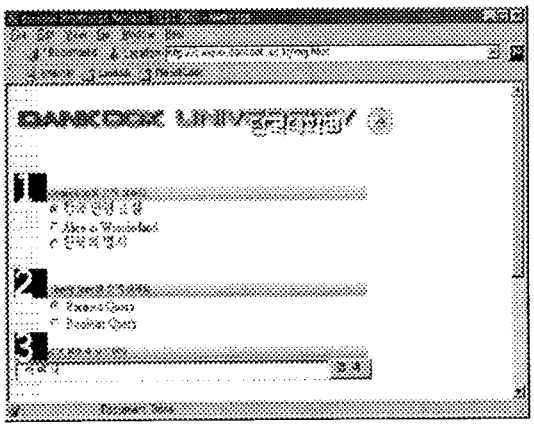


그림 7. HMG 질의 화면

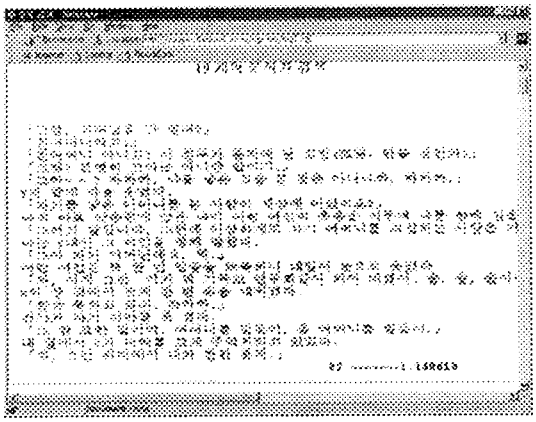


그림 8. 랭킹 질의 결과 화면

5. 결 론

본 연구에서는 MG 1.2 시스템과 한글 형태소 분석기 HAM 4.0을 이용하여 HMG 시스템을 구현하였다. MG 시스템은 기존의 역 화일 구조 시스템의 공간 오버헤드 문제점을 효율적인 압축을 통해 해결하였고, 블록코딩("3-in-4" 코딩)과 스킵 등의 기법을 이용해 메모리 크기, 디스크 용량, 처리시간을 동시에 줄이는 우수한 기법으로, 이러한 특성으로 인하여 기존에 개발된 검색 엔진에 비해 월등한 성능을 보이고 있다. HMG는 이 MG 엔진의 탁월한 성능을 한글 텍스트 데이타베이스에서도 활용할 수 있게 하려는 의도에서 개발되었다.

HMG에서는 한글 처리시 완성형 코드로 한영 혼용문서의 데이타베이스 구축 단계와 질의 처리 단계에서 사용하였다. 한국어 색인시 한글 형태소 분석기인 HAM을 이용하여 정확한 유사도를 반영하였다. 또한 HMG 시스템의 유용성을 보이기 위해 웹상에서 한글 소설 검색 시스템 NOD를 구현하였다. HMG 시스템은 전자 신문, 전자 잡지, 전자 법전, 전자 사전, 디지털 라이브러리 등 한글 텍스트의 지원이 필요한 대규모 전문 검색 시스템이나 텍스트 데이타베이스 구축에 활용할 수 있다.

텍스트 데이타베이스는 웹의 빠른 확장과 인트라넷 그리고 데이타베이스가 통합되어 빠르게 변화하고 있다. 이런 변화에 따라 문서의 구조적 특성이나 속성 값을 사용하여 정보 검색을 수행해야 할 필요성이 대두되고 있다. 구조 기반 검색 시스템의 경우 새로운 인덱싱과 질의 평가 기술에 대한 연구가 필요하다. 또 부분 검색에 있어 B-트리를 이용하였기 때문에 접두어(prefix) 위주의 부분검색에 대한 연구가 수행되어 왔다. 그러나 영문을 위한 부분 검색의 경우 lab* 뿐만 아니라 lab*r과 같은 *n*-gram을 사용한 인덱스에 대한 연구가 있었고[4] 한글에 있어서도 이러한 고급 부분 검색에 대한 연구가 요구된다.

참 고 문 헌

[1] Zobel, J., Moffat, A., and Sacks-Davis, R. "An efficient indexing technique for full-text database systems," in *Proc. of the-18th VLDB Conference,* Vancouver, British Columbia, pp.

352-362, 1992.

[2] Frakes, W. B. and Baeza-Yates, R., *Information Retrieval: Data Structure & Algorithms*, Prentice Hall Inc, 1992.

[3] Moffat, A. and Zobel, J. "Self-indexing Inverted Files for Fast Text Retrieval," *ACM Transactions on Information Systems*, 14(4), pp. 349-379, 1996.

[4] Witten, I. H., Moffat, A., and Bell, T. C., *Managing Gigabytes*, Nostrand Reinhold, New York, 1994.

[5] RMIT, "MG Programmers' Notes", <URL: http://www.mds.rmit.edu.au/mg/prog_notes/toc.html>

[6] 김지승 외, "한글 freeWAIS 구현", 한국정보과학회, '98 춘계학술발표논문집, pp. 6-8, 1998.

[7] Zobel, J., et al., *Indexing Technique for Advanced DataBase System*, pp. 151-183, 1997.

[8] Salton, G. and McGill, M. J., *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

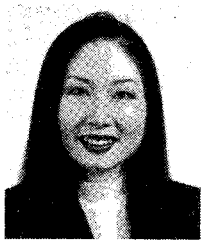
[9] 강승식, "한국어 형태소 분석기와 한국어 분석 모듈", <URL: http://ham.hansung.ac.kr/ham/ham.html>

[10] 송병호, 복합명사 검색을 위한 한글 요약 추출기법, 서울대학교 박사 학위 논문, 1994.2.

[11] Faloutsos, C., *Searching Multimedia Database by Content*, Kluwer Academic Publishers, 1998.

[12] Prabhakaran, B., *Multimedia Database Management Systems*, Kluwer Academic Publishers, 1997.

[13] Korfhage, R. R., *Information Storage and Retrieval*, John Wiley & Sons, Inc., 1997.



박 미 란

1997년 2월 한국방송대 전자계산학과 졸업(이학사)

1999년 2월 단국대학교 컴퓨터공학과 데이터베이스 및 멀티미디어 연구실(졸업예정)

관심분야 : 멀티미디어 데이터베이스, 정보검색, 분산 데이터베이스

나 연 묵

1986년 2월 서울대학교 컴퓨터공학과(공학사)

1988년 2월 서울대학교 대학원 컴퓨터공학과(공학석사)

1993년 2월 서울대학교 대학원 컴퓨터공학과(공학박사)

1991년 미국 IBM T.J.Watson 연구소 (객원연구원)

1993년 8월~현재 단국대 공과대학 컴퓨터공학과(조교수)

1996년 9월~현재 한국정보과학회 데이타베이스연구회(운영위원)

1996년 9월~현재 DAVIC-KR Information Representation WG(분과위원)

1997년 10월~현재 서울시 정보화사업 자문위원회(자문위원)

1997년 10월~현재 한국멀티미디어학회 논문지 편집위원회(편집위원)

관심분야 : 객체지향 데이타베이스, 멀티미디어 데이타베이스, 데이터 모델링, 데이타베이스 설계, 객체지향 시스템, 멀티미디어 정보 검색