정 보 검 색 론

2003. 9

이 준 호 교수님

숭실대학교 정보검색연구실

차 례

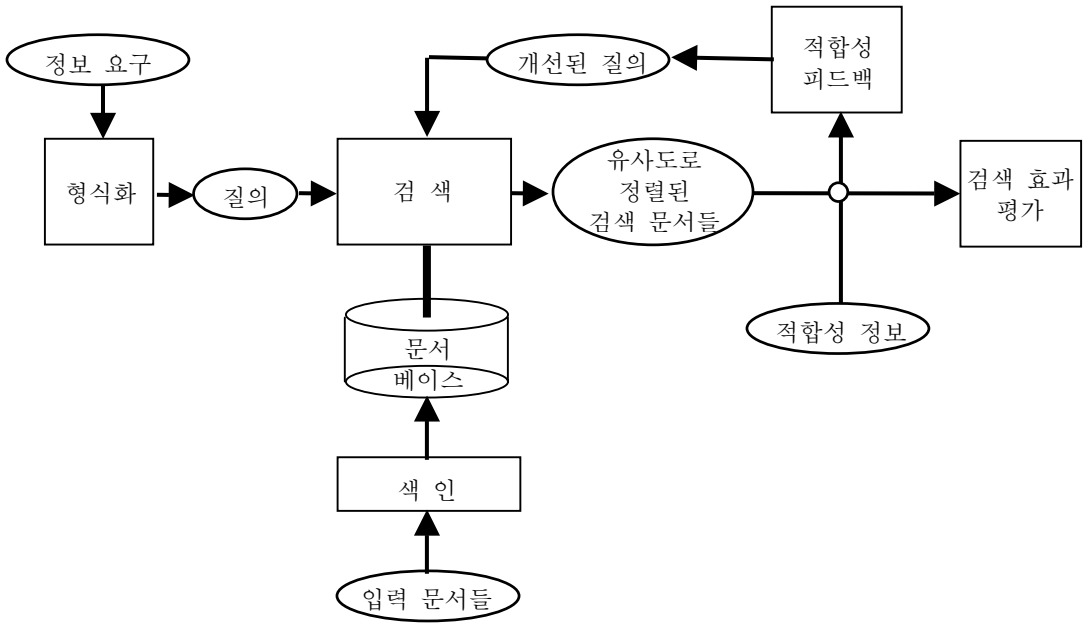
제 1 장 소 개

컴퓨터의 발달과 급속한 보급으로 인하여 전자 문서의 양이 기하급수적으로 증가하고 있다. 이러한 전자 문서들은 텍스트, 이미지, 오디오, 비디오 등과 같은 멀티미디어 객체들로 구성된다. 정보 검색 시스템은 방대한 양의 전자 문서들로부터 사용자가 지니고 있는 정보 요구(information need)를 만족시킬 수 있는 문서들을 검색하는 소프트웨어이다. 즉, 원하는 문서들이 포함하고 있는 멀티미디어 객체들에 대한 설명을 사용자가 정보 검색 시스템에 입력하면, 정보 검색 시스템은 사용자가 제공한 멀티미디어 객체들에 대한 설명을 기반으로 문서들을 검색한다.

1960 년대 이후로 정보 검색에 대한 많은 연구들이 수행되었으나, 텍스트 이외의 멀티미디어 객체들에 대한 검색은 검색 결과의 질이 미흡한 상태이기 때문에 아직까지 널리 활용되고 있지 않다. 그러나 현재 이미지 검색을 수행하는 정보 검색 시스템들이 상업적으로 판매되고 있으며 또한 연구가 지속적으로 수행되고 있기 때문에, 향후 텍스트 이외의 멀티미디어 객체들에 대한 검색도 활발히 보급될 것으로 예상된다.

한편, 텍스트에 대한 검색은 1960 년대 초에 일괄 처리 방식으로 MEDLARS 시스템에 의해 지원되기 시작하였으며, 이후 다수의 텍스트 검색 시스템들이 상용화되어 현재까지 사용되고 있다. 현재 텍스트 검색 시스템은 전자 도서관의 서지 정보 검색, 전자 쇼핑몰의 상품 정보 검색, 신문 잡지사의 기사 검색, 검색 포털들의 웹 문서 검색 등과 같은 다양한 분야에서 활용되고 있다. 즉, 텍스트에 대한 검색은 실용적으로 폭 넓게 사용되고 있으며, 따라서 본 책자에서는 텍스트에 대한 검색만을 기술하고자 한다.

<그림 1.1>은 정보 검색 시스템의 구조를 보여주고 있다. 색인 모듈에 입력된 문서들은 색인어 추출 과정을 거쳐 문서 베이스에 저장되고, 질의 형식화 모듈에 입력된 사용자의 정보 요구는 시스템이 지원하는 질의 언어에 적합한 형태로 형식화된다. 검색 모듈은 형식화된 질의와 정보 베이스에 저장된 문서들 사이의 유사도(similarity)를 계산하여, 유사도가 높은 문서들을 우선적으로 출력한다. 이때 사용자는 출력된 문서들 중에서 적합 문서와 비적합 문서들을 선정함으로써, 검색 결과에 대한 적합성 정보를 시스템에게 제공한다. 사용자의 적합성 정보는 검색 결과의 질을 측정하는데 이용되거나, 또는 검색 효과의 개선을 목적으로 질의를 재구성하는 적합성 피드백 모듈에 의해 활용될 수 있다. 다음에서는 정보 검색 시스템의 구성 요소들에 대해 자세히 설명한다.



<그림 1.1> 정보 검색 시스템의 구조

- **형식화**: 일반적으로 서로 다른 정보 검색 시스템은 서로 다른 질의 문법을 지원하기 때문에, 특정 시스템의 질의 문법에 익숙하지 않은 일반 사용자들은 검색에 효과적인 질의를 작성하는데 어려움을 겪는다. 따라서 사용자는 본인이 지니고 있는 정보 요구를 자연어를 사용하여 기술하고, 형식화 모듈이 자연어로 표현된 정보 요구를 분석하여 검색 모듈의 특성에 부합하는 질의로 변환하는 것이 바람직하다. 한편, 특정 정보 검색 시스템의 질의 문법에 익숙하고 정보 검색에 많은 경험을 지니고 있는 사용자는 형식화 모듈을 사용하지 않고, 질의를 직접 작성함으로써 정보 검색 결과의 질을 향상시킬 수 있다.

- **검색**: 순수한 불리안 검색 시스템은 질의로서 주어진 불리안 수식을 만족하는 문서들을 검색한다. 이러한 불리안 검색 시스템은 질의와 문서 사이의 유사도를 계산하는 정보 검색 시스템에 비하여 낮은 검색 효과를 제공하는 것으로 알려져 왔다. 1960 년대 초에 정보 검색이라는 연구 분야가 확립된 이후로, 질의와 문서 사이의 유사도를 보다 정확히 계산하기 위한 많은 연구들이 수행되어 왔으며, 본 책자에서는 이러한 연구들에 대하여 기술한다. 질의와 문서 사이의 유사도를 계산하는 검색 모듈은 계산된 유사도에 따라 문서에 순위를 부여한다. 높은 순위를 갖는 문서일수록 질의에 대한 만족도가 크며, 사용자는 높은 순위의 문서를 우선적으로 검토함으로써 필요한 정보를 얻는데 소모되는 시간을 최소화할 수 있다.

- **적합성 피드백**: 질의와 문서 사이의 유사도를 계산하는 정보 검색 시스템에서 질의 수정을 자동으로 수행하는 적합성 피드백은 유사도 계산의 질을 높이기 위한 효과적인

방법으로 알려져 왔다. 일반적으로 사용자들은 광범위한 정보 요구를 지니고 시스템에 접근하여 그 요구들을 질의로서 표현한다. 이러한 초기의 질의는 매우 임의적이며, 때때로 사용자들은 그들이 어떠한 정보 요구를 지니고 있는지조차도 파악하지 못 하고 있다. Belkin(1982)은 사용자들의 이러한 상태를 "비정상적인 지식의 상태"라고 불렀고, Bookstein(1983)은 불확실성은 정보 검색 과정의 본질적인 것이라고 말하였다. 따라서 불완전한 초기 질의는 개선되어야 하며, 적합성 피드백은 사용자의 적합성 정보를 기반으로 자동으로 질의를 수정한다.

- **색인**: 과거의 색인 작업은 훈련된 사서나 주제 전문가에 의해 수작업으로 수행되었다. 그러나 수작업에 의한 색인은 과다한 지적 노력을 필요로 하며, 사용된 비용에 비하여 효과면에서도 그다지 효율적이지 못하다는 사실이 여러 실험 결과 밝혀졌다. 또한 수작업에 의한 색인은 색인자의 주관에 따라 색인의 양이나 질이 달라질 수 있기 때문에, 색인의 일관성이 결여되는 문제점을 안고 있다. 따라서 컴퓨터를 사용하여 문서와 질의를 자동적으로 분석함으로써 색인어를 추출하는 자동 색인 기법들이 출현하게 되었다.

- **검색 효과 평가**: 정보 검색 시스템의 검색 효과는 일반적으로 재현율(recall)과 정확률(precision)로써 평가된다. 재현율은 문서 집합에서 사용자가 원하는 문서를 어느 정도 검색하였는가를 나타내고, 정확률은 검색된 문서들 중에서 사용자가 원하는 문서가 얼마나 포함되어 있는가를 나타낸다. 질의와 문서 사이의 유사도를 계산하는 정보 검색 시스템의 검색 결과에 대해서는 보간 기법을 사용하여 고정된 재현율에 대한 정확률을 계산할 수 있다. 이 외에도 검색 효과를 평가할 수 있는 다양한 방법들이 정보 검색 분야에서 연구되어 왔다.

제 2 장 웹 로봇

인터넷의 이용이 활발해짐에 따라 수 많은 정보들이 웹 문서의 형태로 공개되고 있으며, 이러한 웹 문서들을 효과적으로 검색하기 위하여 웹 검색 서비스들이 이용되고 있다. 웹 로봇은 지정된 URL 리스트에서 시작하여 웹 문서를 수집하고, 수집된 웹 문서에 포함된 URL 들의 추출 과정과 새롭게 발견된 URL 에 대한 웹 문서 수집 과정을 반복하는 소프트웨어로서 웹 검색 서비스의 구축을 위해서는 웹 로봇을 이용한 웹 문서 수집이 선행되어야 한다. 웹 로봇의 웹 문서 수집 결과는 웹 검색 결과의 품질에 많은 영향을 미치며, 이는 웹 검색 서비스들이 수집된 웹 문서들만을 대상으로 검색을 수행하기 때문이다.

1990 년대 중반의 웹 문서 수는 현재에 비하여 매우 적었기 때문에, 최초로 개발된 웹 로봇 Wanderer 를 포함하여 이 당시 개발된 다수의 웹 로봇들은 대용량의 웹 문서들을 수집하도록 설계되지 않았다. 그러나, 현재는 전세계적으로 30 억 개 이상의 웹 문서들이 존재하며, 국내에도 5 천만 개 이상의 웹 문서들이 존재하고 있다. 따라서 이처럼 많은 수의 웹 문서들을 효율적으로 수집할 수 있는, 즉 초당 수백 또는 수천개의 웹 문서들을 수집할 수 있는 웹 로봇의 필요성이 증가하고 있다.

한편, 웹 환경은 다음과 같은 특성들을 지니고 있으며, 웹 로봇은 이러한 특성들을 고려하여 개발되어야 한다. 첫째, 웹에 공개되는 문서들의 수가 매우 빠르게 증가하고 있다. 둘째, 웹 로봇이 문서들을 수집하고 있는 동안에도 이들에 대한 수정 및 삭제가 수행된다. 셋째, 웹에는 양질의 문서들뿐만 아니라 다수의 유해 또는 스팸 웹 문서들이 존재한다. 넷째, 웹에는 다수의 동일한 내용을 지닌 문서들이 존재한다. 다섯째, 로봇 배제 표준이 준수되어야 한다.

본 연구에서는 위에서 언급된 웹 환경의 특성들과 기존 웹 로봇들의 기능들을 조사하여, 웹 로봇들의 성능을 체계적으로 평가하기 위한 기준들을 제시하고, 각 평가 기준의 향상에 도움이 되는 기능들을 기술한다. 또한, 본 연구에서는 네이버, 구글, 알타비스타에서 상업용으로 사용되고 있는 웹 로봇들과 폴리테크닉 대학교에서 연구용으로 개발된 웹 로봇에 구현된 기능들을 조사한다.

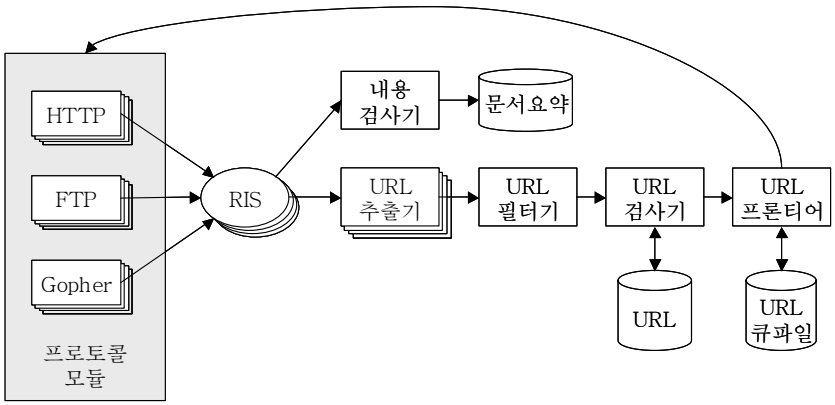
본 연구의 구성은 다음과 같다. 2.1 절에서는 지금까지 개발된 웹 로봇들의 구조와 구성요소들에 대하여 설명한다. 2.2 절에서는 웹 로봇을 체계적으로 평가할 수 있는 기준들을 제시하고, 또한 각 평가 기준들을 향상시킬 수 있는 기능들을 기술한다. 그리고 2.3 절에서는 기존의 웹 로봇들이 2.3 절에서 기술한 기능들을 포함하고 있는가를 분석한다.

2.1 웹 로봇의 구조

2.1.1 메르카토르

메르카토르(mercator)는 웹 검색 서비스인 알타비스타에서 이용하고 있는 웹 로봇으로서 DEC/Compaq 에서 개발되었다. 메르카토르는 필요한 기능들을 플러그인 방식으로 추가함으로써 시스템을 쉽게 확장할 수 있도록 설계되었으며, 또한 자바 프로그래밍 언어로 개발되었기 때문에 자바 가상 기계가 설치된 모든 플랫폼에서 실행될 수 있다.

<그림 2.1>은 메르카토르 시스템의 구조를 보여준다. 메르카토르는 URL 프론티어를 호출하여 수집할 웹 문서의 URL 을 획득하고, 이 URL 을 HTTP, FTP, Gopher 중에서 적합한 프로토콜 모듈에게 전달한다. 프로토콜 모듈은 URL 에 의해 지정된 웹 문서를 다운로드하며, 이 문서는 RIS 에 의해 관리된다. 메르카토르는 RIS 에 의해 관리되는 각각의 웹 문서에 대하여 내용 검사기를 호출함으로써 이미 수집된 웹 문서들과의 중복 여부를 확인한 후, 중복되지 않은 웹 문서로부터 URL 들을 추출한다. 이 URL 들 중의 일부는 URL 필터기에 의해 제거되며, 나머지 URL 들로부터 URL 검사기는 지금까지 수집되지 않은 웹 문서들의 URL 을 검출한 후, 이들을 URL 프론티어로 전달한다.

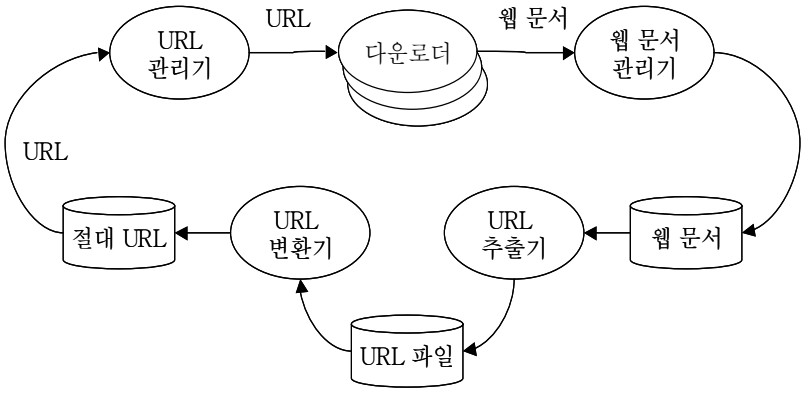


<그림 2.1> 메르카토르 시스템의 구조

2.1.2 구글봇

구글봇(googlebot)은 웹 검색 서비스를 제공하는 구글에서 사용하고 있는 웹 로봇으로, 스탠포드 대학의 학생이었던 Page & Brin 에 의해 개발되었다. 구글은 이러한 구글봇을 이용하여 전 세계를 대상으로 30 억개 이상의 웹 문서를 수집하고 있다. 또한, 구글은 상업화된 이후에도 스탠포드 대학과 웹 문서 수집에 관련된 연구를 지속적으로 수행하고 있으며, 그 결과로는 웹 문서들의 병렬 수집, 중복된 문서들의 검출, 동적인 웹 문서들의 수집, 웹 문서들의 수정 주기 분석 등이 있다.

<그림 2.2>는 구글봇 시스템의 구조를 보여 준다. 구글봇은 URL 관리기, 다운로더, 웹 문서 관리기, URL 추출기, URL 변환기로 구성되어 있으며, 각각의 구성 요소는 독립적인 프로세스로서 존재한다. URL 관리기는 수집할 웹 문서들의 URL 들을 다수의 다운로더들에게 분배한다. 각각의 다운로더는 서로 다른 컴퓨터에서 실행되고, 웹 문서 관리기는 다운로드된 웹 문서들을 압축하여 디스크에 저장한다. URL 추출기는 디스크에 저장된 웹 문서들로부터 URL 들을 추출하고, URL 변환기는 이 URL 들을 절대 URL 로 변환하여 디스크에 저장한다.

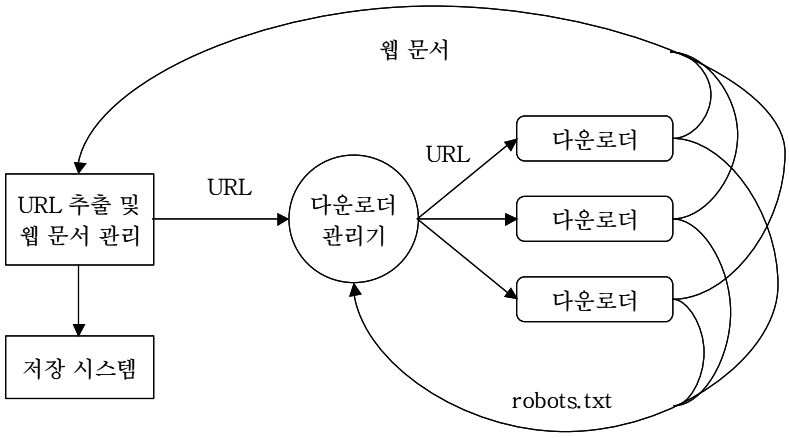


<그림 2.2> 구글봇 시스템의 구조

2.1.3 폴리봇

폴리봇(polybot)은 폴리테크닉 대학교에서 연구용으로 개발된 웹 로봇으로서, 폴리봇의 구성 요소들은 서로 다른 컴퓨터에서 독립적으로 실행될 수 있다. 또한, 각각의 구성 요소에 컴퓨터를 추가함으로써 웹 문서 수집 성능을 향상 시킬 수 있다. 이러한 폴리봇은 실험을 통하여 18 일 동안 약 5 백만 개의 호스트에서 1 억 2 천만 개 이상의 웹 문서들을 수집하였다.

<그림 2.3>은 폴리봇 시스템의 구조를 보여준다. 웹 로봇이 특정 웹 서버로부터 많은 수의 웹 문서들을 단기간에 수집할 경우, 웹 서버에 과도한 부하를 유발시킬 수 있다. 이러한 문제를 방지하기 하기 위해 수집 관리기는 URL 들을 뒤섞음(shuffling)한 후, 이 URL 들을 다운로더들에게 전달한다. 또한, 수집 관리기는 웹 서버의 " robots.txt" 파일을 분석하여 로봇 배제 표준을 준수한다. 폴리봇은 다운로드된 웹 문서들로부터 URL 들을 추출하고, 지금까지 수집되지 않은 웹 문서들의 URL 들을 수집 관리기로 전달하며, 웹 문서들을 저장 시스템에 전달한다.

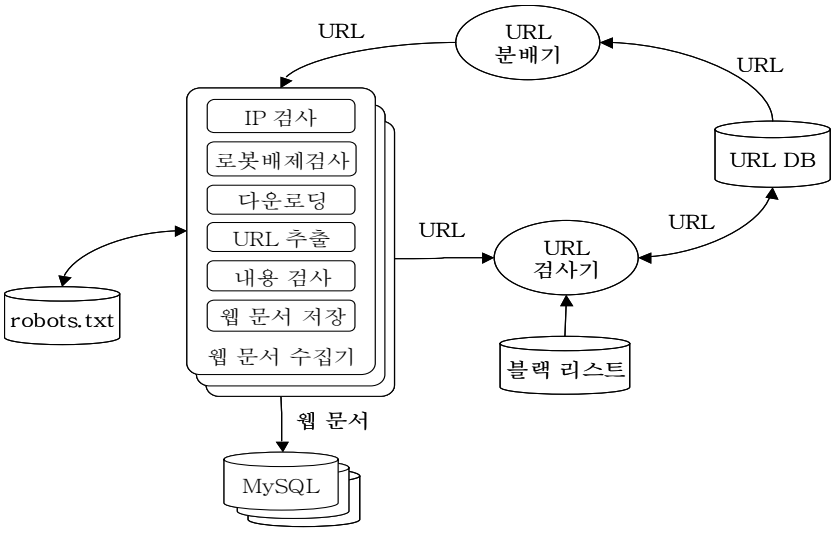


<그림 2.3> 폴리봇 시스템의 구조

2.1.4 네이봇

네이봇(nabot)은 웹 검색 포탈 네이버에서 사용하는 웹 로봇으로서 국내 및 일본의 웹 문서들을 수집한다. 네이봇은 데이터베이스 관리 시스템 MySQL 을 사용하여 수집된 웹 문서들을 관리하며, 또한 과거에 수집된 웹 문서들을 지속적으로 수집하기 위하여 지금까지 수집된 전체 웹 문서들의 URL 을 관리한다. 네이봇은 관리중인 URL 들이 지시하는 웹 문서만을 수집하고, 수집된 웹 문서들로부터 발견된 새로운 URL 들을 URL 데이터베이스에 추가한다. 따라서 새롭게 발견된 URL 들이 지시하는 웹 문서들은 다음 번 네이봇 수행시에 수집된다.

<그림 2.4>는 네이봇 시스템의 구조를 보여준다. URL 분배기는 관리중인 URL 들을 다수의 컴퓨터에 분산되어 있는 웹 문서 수집기들에게 분배하며, 웹 문서 수집기는 다음과 같은 작업들을 수행한다. 첫째, URL 의 IP 를 검사하여 국내 또는 일본의 웹 문서인지를 확인한다. 둘째, 로봇 배제 표준을 준수하기 위해서 웹 서버의 " robots.txt" 파일의 내용을 확인한다. 셋째, 웹 문서를 다운로드한다. 넷째, 다운로드된 문서로부터 URL 들을 추출하여 URL 검사기로 전달한다. 다섯째, 웹 문서의 내용을 분석하여 유해 또는 스팸 문서인가를 검사한다. 마지막으로, 웹 문서를 압축하여 데이터베이스에 저장한다. 한편, URL 검사기는 전달된 URL 들 중에서 블랙 리스트에 포함된 URL 들과 기존의 URL 들을 제거한 후, 나머지 URL 들을 URL 데이터베이스에 추가한다.



<그림 2.4> 네이봇 시스템의 구조

2.2 평가 기준

본 절에서는 웹 로봇의 성능을 체계적으로 평가하기 위한 기준으로서 효율성, 연속성, 신선성, 포괄성, 정숙성, 유일성, 안전성을 제시하고, 또한 이러한 평가 기준들의 향상에 도움이 되는 기능들을 기술한다. 웹 로봇 개발자는 각각의 평가 기준에 대하여 열거된 기능들을 웹 로봇에 추가함으로써, 해당 평가 기준을 향상시키고, 보다 높은 성능의 웹 로봇을 개발할 수 있다.

2.2.1 효율성

웹 문서들의 수가 급격히 증가하고 있는 현실을 고려할 때, 초당 수백 또는 수천 개의 웹 문서들을 수집할 수 있는 웹 로봇에 대한 필요성이 절실하다. 효율성(efficiency)은 주어진 시간 내에 수집할 수 있는 웹 문서들의 수를 의미하며, 보다 효율적인 웹 로봇의 개발을 위해서는 다음과 같은 사항들이 웹 로봇의 설계에 반영되어야 한다.

병렬 수집: 웹 로봇은 수집된 웹 문서들을 디스크에 저장하기 위해 많은 시간을 소비하는 입출력 집중 프로그램(I/O intensive program)이다. 따라서 하나의 컴퓨터에서 다수의 웹 로봇 프로세스를 동시에 실행함으로써 주어진 시간에 보다 많은 웹 문서들을 수집할 수 있다.

분산 수집: 현재 국내의 웹 문서들의 수는 5 천만 건을 넘어서고 있으며, 이러한 웹 문서들을 하나의 컴퓨터를 이용하여 수집할 경우, 몇 주 정도의 시간이 소비된다. 따라서 웹 문서 수집에 소비되는 시간을 보다 단축하기 위해서는 다수의 컴퓨터를 이용한 웹 문서들의 분산 수집이 절실하다.

분산 저장: 일반적으로 웹 로봇 시스템은 서로 다른 기능을 수행하는 다수의 컴퓨터로 구성되며, 네트워크를 통하여 웹 문서들을 내려 받는 기능과 웹 문서들을 저장하는 기능은 별도의 장비에서 수행된다. 따라서 하나의 저장 서버에 모든 웹 문서들을 저장할 경우, 저장 서버에 입출력 병목 현상이 발생할 수 있기 때문에, 웹 문서들을 다수의 저장 서버에 분산 저장하는 것이 바람직하다.

DNS 캐시: 하나의 컴퓨터에서 다수의 웹 로봇 프로세스들을 실행하고, 더불어 다수의 컴퓨터를 이용하여 웹 문서들을 수집할 경우, URL 을 IP 주소로 변환하는 DNS 서버에 병목 현상이 발생할 수 있다. 이러한 병목 현상은 호스트 이름과 IP 주소를 자체적으로 관리함으로써 줄일 수 있다.

2.2.2 지속성

일반적으로 웹 검색 서비스들은 주기적으로 웹 로봇을 수행하며, 지속성(continuity)은 이전에 수집된 후 현재까지 삭제되지 않은 웹 문서들에 대한 재수집 된 웹 문서들의 비율로서 정의된다. 즉, 지속성은 이전에 수집된 후 현재까지 삭제되지 않은 웹 문서들이 어느 정도 재수집 되었는가를 나타낸다. 이러한 지속성을 향상시키기 위해서는 다음과 같은 사항들이 웹 로봇의 설계에 반영되어야 한다.

전체 URL 관리: 많은 경우에 웹 로봇은 씨앗(seed) URL 들에서 시작하여, 이들로부터 접근 가능한 모든 웹 문서들을 수집한다. 그러나, 네트워크의 속도 저하, DNS 서버의 장애, 특정 웹 사이트에서의 일시 점검 등으로 인하여, 동일한 씨앗 URL 에 대하여 동일한 웹 로봇이 수행될지라도 웹 로봇의 수행 시점에 따라 서로 다른 웹 문서 집합들이 수집 될 수 있다. 따라서, 지속성을 높이기 위해서 과거에 수집된 모든 웹 문서들의 URL 을 관리하는 것이 바람직하다.

웹 문서 추가: 과거에 수집된 모든 웹 문서들에 대한 URL 을 관리하고, 이후 이들에 대한 재수집을 항상 시도할지라도, 다양한 장애 요인들로 인하여 이전에 수집된 웹 문서들의 재수집에 실패할 가능성이 여전히 존재한다. 이 경우 재수집에 실패한 웹 문서들을 이전에 수집된 웹 문서 데이터베이스로부터 추출함으로써 지속성을 높일 수 있다.

2.2.3 신선성

신선성(freshness)은 이전에 수집된 웹 문서들에 대한 현재 수정 및 삭제되지 않은 웹 문서들의 비율로서 정의된다. 즉, 신선성은 이전에 수집된 웹 문서들이 현재 어느 정도 수정 및 삭제 되었는가를 나타낸다. 수집된 웹 문서들의 신선성이 저하될수록, 삭제 또는 수정된 웹 문서들이 검색 결과에 포함될 가능성이 증가한다. 따라서 신성성의 저하는 웹 검색 결과에 대한 이용자의 만족도를 저하시키기 때문에, 이를 향상시키기 위하여 다음과 같은 사항들이 웹 로봇의 설계에 반영되어야 한다.

수집 주기 관리: 웹 문서는 그 내용에 따라 빈번하게 수정되거나, 또는 수개월이 지나도 수정되지 않을 수 있다. 이처럼 웹 문서들은 서로 다른 수정 주기를 지니기 때문에, 동일한 주기로 웹 문서들을 수집할 경우 신선성이 저하될 수 있다. 따라서, 웹 문서들의 수정 주기를 측정하고, 수정 주기가 짧은 웹 문서들을 보다 자주 수집함으로써 신선성을 향상시킬 수 있다.

URL 삭제: 웹 로봇이 지속성의 개선을 위하여 수집 가능한 전체 URL 을 관리하도록 설계된 경우, 웹 로봇은 보유중인 각각의 URL 에 해당하는 웹 문서의 수집을 시도하며, 종종 데드링크 즉, 접근되지 않는 웹 문서에 직면한다. 데드링크가 발생하는 이유들 중의 하나는 해

당 웹 문서가 삭제되어 더 이상 존재하지 않기 때문이다. 웹 로봇은 이러한 경우를 확인하여 삭제된 웹 문서의 URL 을 관리 중인 URL 데이터베이스로부터 삭제함으로써 신선성을 향상시킬 수 있다.

2.2.4 포괄성

포괄성(coverage)은 웹 로봇이 수행을 시작하여 종료할 때까지 수집한 웹 문서들의 수로 정의된다. 웹 검색 서비스들은 웹 로봇이 수집한 웹 문서들에 대한 검색을 지원하기 때문에, 포괄성은 웹 검색 결과의 질을 결정하는 중요한 요소들 중의 하나로 인식되고 있으며, 이를 향상시키기 위해서 웹 로봇은 다음과 같은 형태의 문서들을 수집할 수 있도록 설계되어야 한다.

정적 웹 문서: 정적 웹 문서는 HTML 로 작성된 가장 기본적인 형태의 웹 문서로서, URL 에 의해 서로 연결되어 있다. 이러한 정적 웹 문서를 수집하기 위해서 웹 로봇은 지정된 URL 리스트에서 시작하여 웹 문서를 수집하고, 이후 수집된 웹 문서에 포함된 URL 들의 추출 과정과 새롭게 발견된 URL 에 대한 웹 문서 수집 과정을 반복한다.

동적 웹 문서: 동적 웹 문서는 이용자가 원하는 정보를 얻기 위하여 웹 문서의 폼(form)에 값 또는 질의를 입력한 후 서버에 요청하여 생성되는 문서들이며, “ Hidden Web”, “ Deep Web” 등으로 불리기도 한다. 현재 웹 상에 존재하는 동적 웹 문서들의 수는 정적 웹 문서 수의 수백 배를 넘는 것으로 조사되고 있다. 웹 로봇이 동적 웹 문서를 수집하기 위해서는 웹 문서 내의 폼을 분석하고, 적절한 값들을 이용한 URL 작성이 선행되어야 한다.

일반 문서: 인터넷을 통한 정보 교류가 활발해 짐에 따라 학술지, 간행물, 논문집 등과 같은 많은 일반 문서들이 워드프로세서, PDF, PS 파일 형태로 인터넷에 공개되고 있다. 이러한 일반 문서들은 웹 문서들보다 양질의 정보들을 포함하고 있을 가능성이 높기 때문에, 현재 다수의 웹 검색 서비스들은 이러한 일반 문서들에 대한 검색을 제공하고 있다. 이러한 일반 문서를 수집하기 위해서는 웹 로봇은 정적 및 동적 웹 문서에 포함된 일반 문서 URL 들의 추출이 선행되어야 한다.

2.2.5 정숙성

웹 로봇들은 인터넷에 연결되어 있는 웹 서버들에 접근하여 그 서버에 존재하는 웹 문서들을 수집한다. 따라서 웹 로봇이 특정 웹 서버로부터 단기간에 다수의 웹 문서들을 수집할 경우, 과도한 부하를 유발하여 웹 서버가 제공하는 서비스들에 피해를 줄 수 있다. 이러한 피해의 회피 및 보유하고 있는 정보들의 보호를 위하여 일부 웹 서버들은 로봇 배제 표

준을 이용하여 웹 로봇들의 접근을 배제하고 있다. 정숙성은(silence)은 웹 문서 수집 기간 동안 피해를 주지 않은 웹 서버들의 수로 정의되며, 이러한 정숙성을 향상시키기 위해서는 다음과 같은 사항들이 웹 로봇의 설계에 반영되어야 한다.

로봇 배제 표준: 로봇 배제 표준은 " robots.txt" 파일 또는 로봇 메타 태그를 이용하여 웹 로봇들의 웹 문서 수집을 제한한다. 즉, 웹 서버 관리자들은 " robots.txt" 파일에 웹 로봇들의 이름을 명시함으로써 사이트 내의 전체 웹 문서들에 대한 수집을 방지하거나, 또는 디렉토리 이름들을 명시함으로써 일부 웹 문서들에 대한 수집만을 방지할 수 있다. 또한, 웹 문서 작성자들은 로봇 메타 태그를 이용하여 작성된 웹 문서에 대한 수집을 제한할 수 있다. 따라서, 웹 로봇은 웹 문서를 수집하기 전에 " robots.txt" 파일과 로봇 메타 태그들을 확인하여, 지정된 내용을 준수해야 한다.

수집 속도 조절: 웹 로봇이 효율성을 향상시키기 위하여 단기간 내에 많은 웹 문서들을 수집하도록 설계된 경우, 접근한 웹 서버의 부하를 급격히 증가시키는 경우가 종종 발생한다. 실제 웹 검색 서비스 업체의 웹 로봇 운영자들은 웹 로봇이 접근했던 웹 서버를 소유한 회사나 기관으로부터 항의성 메일이나 전화를 받은 경험들을 지니고 있다. 따라서, 웹 서버의 부하가 증가하지 않도록 웹 로봇의 웹 문서 수집 속도를 조절해야 한다.

2.2.6 유일성

인터넷에는 내용이 동일한 다수의 웹 문서들이 존재하며, 이러한 웹 문서들의 수집은 웹 로봇의 효율성을 저하시키고 저장 공간의 낭비를 초래한다. 유일성(uniqueness)은 수집된 전체 웹 문서들에 대한 중복되지 않은 유일 웹 문서들의 비율로 정의되고, 이러한 유일성을 향상시키기 위해서는 다음과 같은 사항들이 웹 로봇의 설계에 반영되어야 한다.

URL 정규화: 인터넷에는 하나의 웹 문서에 대한 주소가 다양한 형태의 URL 로 표현되고 있다. 예를 들어, " www.yahoo.co.kr" , " www.yahoo.co.kr/" , " www.yahoo.co.kr/index.html" , " www.yahoo.co.kr:80" 은 모두 동일한 웹 문서를 지시하는 URL 들이다. 웹 로봇이 이러한 URL 들 모두에 대하여 웹 문서를 수집할 경우, 동일한 내용의 웹 문서들이 중복 수집된다. 따라서, 이를 방지하기 위하여 하나의 웹 문서 주소에 대한 다양한 형태의 URL 들을 하나의 URL 로 정규화함이 바람직하다.

호스트 이름 그룹화: URL 정규화 이후에도 서로 다른 URL 들에 대해서 동일한 내용의 웹 문서들이 수집될 수 있으며, 그 이유는 다음과 같다. 첫째, 많은 경우에 하나의 호스트 이름은 다수의 별칭을 지니고 있으며, 이러한 별칭들은 URL 정규화 이후에도 서로 다른 URL 로 표현된다. 둘째, 웹 사이트 전체를 다른 호스트 이름의 컴퓨터에 동일하게 복사한 미러(mirror) 사이트들이 존재한다. 따라서, 웹 로봇은 하나의 호스트 이름에 대한 별칭들 또는 미러 사이트 이름들을 그룹화하여 관리함이 바람직하다.

중복 문서 검출: URL 정규화와 호스트 이름 그룹화를 통하여 수집된 웹 문서들 내에서도 여전히 동일한 내용의 문서들을 발견할 수 있다. 웹 로봇은 웹 문서들의 내용들을 비교하여 이러한 중복 문서들을 검출하는 것이 바람직하다. 일반적으로 웹 문서들의 내용을 비교하기 위해서 전체 문서 내용을 일정한 길이의 축약된 메시지로 변환하는 MD5, SHA 등과 같은 알고리즘들이 이용된다.

2.2.7 안전성

인터넷에 공개된 웹 문서들 중에는 음란, 폭력 등의 내용을 포함하는 유해 웹 문서와 상업적인 목적으로 이용자들의 방문을 유도하는 스팸 웹 문서들이 존재하며, 웹 검색 서비스들의 검색 결과에도 이러한 웹 문서들이 포함되는 경우를 종종 볼 수 있다. 안전성(safety)은 수집된 전체 웹 문서들에 대한 유해 및 스팸이 아닌 웹 문서들의 비율로 정의되며, 이러한 안전성을 향상하기 위해서는 다음과 같은 사항들이 웹 로봇의 설계에 반영되어야 한다.

유해 웹 문서 검출: 유해 웹 문서는 선정, 음란, 폭력적인 내용을 포함하고 있는 문서로서 성인들에게는 불쾌감을 유발하며, 특히 청소년들이나 어린이들에게는 악영향을 줄 수 있다. 따라서 웹 로봇은 이용자들이 안전하게 웹 검색 서비스를 이용할 수 있도록 수집된 웹 문서들의 내용을 분석하여 유해 웹 문서들을 검출한 후, 이들을 제거함으로써 안전성을 높일 수 있다.

스팸 웹 문서 검출: 일반적으로 스팸 웹 문서는 웹 문서의 내용과 관련 없는 인기 키워드들을 웹 문서에 추가함으로써 생성된다. 스팸 웹 문서는 상업적인 목적으로 이용자들의 방문을 유도하기 위해 작성되며, 검색 결과의 품질을 저하시키는 요인이 된다. 웹 로봇은 수집된 웹 문서들의 내용을 분석하여 스팸 웹 문서들을 검출한 후, 이들을 제거하는 것이 바람직하다.

블랙 리스트 관리: 블랙 리스트는 이전에 발견되었거나 웹 로봇의 수행 중에 발견된 유해 또는 스팸 사이트들의 목록이다. 웹 로봇은 이러한 블랙 리스트들을 이용하여 유해 또는 스팸 웹 문서의 수집을 사전에 차단할 수 있다.

2.3 웹 로봇 분석

메르카토르, 구글봇, 네이봇은 각각 알타비스타, 구글, 네이버에서 사용되고 있으며, 폴리봇은 폴리테크닉 대학교에서 연구용으로 개발되었다. 본 장에서는 지금까지 발표된 논문들을 기반으로 메르카토르, 구글봇, 폴리봇, 네이봇이 3 장에서 기술한 기능들을 포함하고 있는가를 분석하였으며, <표 2.1>은 이러한 분석 결과를 보여준다. 단, 네이봇에 대해서는 아직까지 발표된 논문이 없기 때문에, 이의 분석은 네이봇을 개발하고 있는 NHN㈜의 협조하에 수행되었다.

<표 2.1> 웹 로봇의 분석 결과

평가 기준	기능	웹 로봇			
		메르카토르	구글봇	폴리봇	네이봇
효율성	병렬 수집	○	○	○	○
	분산 수집	○	○	○	○
	분산 저장	○	○	○	○
	DNS 캐시	○	○	○	×
지속성	전체 URL 관리	○	○	?	○
	웹 문서 추가	?	?	?	○
신선성	수집 주기 관리	×	○	×	×
	URL 삭제	?	?	?	○
포괄성	정적 웹 문서	○	○	○	○
	동적 웹 문서	○	○	?	○
	일반 문서	○	○	?	○
정숙성	로봇 배제 표준	○	○	○	○
	수집 속도 조절	○	○	○	○
유일성	URL 정규화	○	○	○	○
	호스트 이름 그룹화	○	○	?	○
	중복 문서 검출	○	○	○	○
안전성	유해 웹 문서 검출	?	?	?	○
	스팸 웹 문서 검출	?	?	?	○
	블랙 리스트 관리	?	?	?	○

<표 2.1>에서 "○"와 "×"는 각각 웹 로봇에 구현된 기능과 구현되지 않은 기능을 의미하고, "?" 는 발표된 논문들에서 확인할 수 없는 기능을 의미한다. <표 2.1>로부터 웹 로봇들은 효율성, 지속성, 포괄성, 정숙성, 유일성에 주안점을 두고 개발되었고, 상대적으로 신선성을 향상시키기 위한 기능들이 개발되지 않았으며, 또한 국외에서 개발된 웹 로봇들은 안전성을 위한 기능들의 개발에 많은 관심을 두지 않고 있음을 알 수 있다. 다음에서는 각각의 평가 기준과 연관된 웹 로봇들의 기능에 대하여 기술한다.

효율성: 웹 로봇들은 병렬 수집, 분산 수집, 분산 저장 기능을 공통적으로 사용하고 있다. 또한, 메르카토르, 구글봇, 폴리봇은 DNS 캐시 기능을 이용하여 DNS 접근시 발생할 수 있는 병목 현상을 완화하고 있다.

지속성: 폴리봇을 제외한 웹 로봇들은 최상위 URL 뿐만 아니라 전체 URL 들을 관리하고 있다. 또한, 네이봇은 재수집에 실패한 웹 문서들을 이전에 수집한 웹 문서 데이터베이스로부터 추출하여 추가함으로써 지속성을 향상시키고 있다.

신선성: 구글봇은 웹 문서들의 수정 주기를 측정하여 그 주기에 따라 웹 문서들을 수집하며, 다른 웹 로봇들은 모든 웹 문서들에 동일한 수집 주기를 적용한다. 또한, 네이봇은 주기적으로 웹 문서들의 접근 가능 여부를 확인하여 삭제되었다고 판단된 문서들의 URL 을 URL 데이터베이스로부터 제거한다.

포괄성: 모든 웹 로봇들은 HTML 로 작성된 정적인 웹 문서들과 더불어 일부의 동적인 웹 문서들을 수집하고 있으며, 폴리봇을 제외한 웹 로봇들은 워드프로세서, PDF, PS 파일 형태로 인터넷에 공개된 일반 문서들도 수집하고 있다.

정숙성: 웹 로봇들은 로봇 배제 표준을 준수함으로써 네티켓을 지키고 있으며, 접근한 웹 서버에 과도한 부하를 주지 않기 위하여 수집 속도를 조절한다. 메르카토르는 일정한 시간 내에 수집하는 문서들의 수를 제한함으로써 직접적으로 수집 속도를 조절하고, 폴리봇과 네이봇은 수집할 전체 URL 들을 뒤섞음하여 간접적으로 수집 속도를 조절한다.

유일성: 모든 웹 로봇들은 URL 정규화와 중복 문서 검출을 수행하며, 폴리봇을 제외한 웹 로봇들은 호스트 이름 그룹화를 수행함으로써 웹 문서의 유일성을 향상시키고 있다. 한편, 중복 문서를 검출하기 위하여 구글봇은 웹 문서들 사이의 유사도를 계산하고, 네이봇과 메르카토르는 각각 MD5 와 Finger Print 알고리즘을 사용하여 웹 문서에 대한 요약 정보를 생성한다.

안전성: 네이봇은 웹 문서를 수집하는 과정에서 웹 문서들의 내용을 분석하여 유해 웹 문서와 스팸 웹 문서를 검출하여 제거한다. 또한 블랙 리스트를 구축하여 유해 또는 스팸 사이트들에 포함된 웹 문서들의 수집을 사전에 차단한다.

제 3 장 웹 검색 행태

웹 검색 이용자들의 검색 형태 연구를 위해 기존에 많이 사용되었던 설문 조사 또는 인터뷰 자료를 분석하는 방법은 이용자의 실제 검색 행위와 다소 차이가 있는 결과를 생성할 수 있다. 이러한 문제를 해결하기 위하여, 국외에서는 웹 검색 서비스 시스템이 생성한 질의 로그를 분석하는 방법을 사용하고 있다. 이용자와 검색 서비스 시스템의 모든 검색 과정을 기록, 저장한 질의 로그는 이용자의 실제 검색 행위를 사실적으로 반영한다. 따라서 이러한 질의 로그의 분석은 웹 검색 이용자들의 검색 행태 연구를 위한 합리적이고 객관적인 방법으로 인정 받고 있다(Jansen and Pooch 2001). 질의 로그 분석을 통하여 웹 검색 이용자들의 검색 행태를 조사한 국외 연구들로는 익사이트 연구(Jansen, Spink, and Saracevic 2000; Spink et al. 2001; Spink et al. 2002), 파이어볼 연구(Hoelscher 1998), 알타비스타 연구(Silverstein et al. 1999) 등을 들 수 있다.

Jansen, Spink & Saracevic(2000)은 1997년 3월 9일 익사이트에서 생성된 51,473개의 질의를 분석하였다. 그리고 Spink et al.(2001)은 1997년 9원 16일 익사이트 엔진의 이용자들이 남긴 100만개 이상의 질의를 분석하였다. 또한, Spink et al.(2002)은 1997년, 1999년, 2001년에 수집된 자료들을 비교, 분석하였는데, 그 결과 이용자들이 주로 검색하는 주제는 연예와 성 관련으로부터 전자 상거래에 관련된 주제로 바뀌었으나, 이용자들의 전반적인 검색 행태는 변하지 않았음을 발견하였다. Hoelscher(1998)는 1998년 7월 한 달 동안 독일의 웹 검색 엔진인 파이어볼에서 생성된 약 1,600만개의 질의 로그를 분석하였다. Silverstein et al.(1999)은 1998년 8월 2일부터 9월 13일까지 6주간 알타비스타 이용자들이 남긴 2억 8천 5백 만개 이상의 이용자 세션, 9억 9천 만개 이상의 질의를 분석하였다. 이 연구는 지금까지 질의 로그를 분석한 연구 중 가장 장기간에 걸쳐 가장 방대한 자료를 연구 대상으로 했다는 점에서 의미가 있다.

질의 로그 분석을 이용한 이들 국외 연구들이 공통적으로 발견한 것은 웹 검색에 있어서 검색 방식의 단순성이다. 즉 웹 검색 이용자들은 복잡한 검색식이나 연산자를 사용하지 않고, 적은 수의 검색어로 구성된 단순한 질의를 통해 정보 검색을 수행하는 경향이 있었다(Spink et al. 2002; Jansen and Pooch 2001; Abdulla, Liu, and Fox 1998). 이러한 검색 행태는 전통적인 정보 검색 시스템 이용자들의 검색 행태와는 매우 상이하다고 할 수 있다.

한편, 국내 웹 검색에 관한 연구는 전산학, 경영학, 문헌정보학, 심리학, 신문방송학 등 다양한 분야에서 수행되고 있다. 그리고, 이용자 검색 행태를 분석한 연구들은 주로 문헌정보학 분야에서 수행되어 왔으며, 이들 중 대부분의 연구들은 전공, 연령, 검색 경험에 있어서 비교적 동질적인 소수의 이용자들을 대상으로 이루어져 왔다. 즉, 오경묵, 황상규, 이용현(1999)은 전자과 대학원생 19명을 대상으로 수행한 실험을 통해 이용자들의 행동

양식을 조사하고, 4개의 검색 시스템을 비교하였으며, 이명희(1998)는 교육학 분야의 주제 전문가 10명을 주제 전문가와 검색 전문가로 나누어 이들의 검색 행태를 비교하였다. 또한, 오삼균, 박희진(2000)은 대학생 30명을 대상으로 실험을 실시하여 한글 알타비스타와 네이버를 검색 효율성, 검색 결과의 정확률, 검색 결과의 갱신성, 이용자의 만족도로 비교, 평가하였다. 이들 연구들은 실험, 설문 조사 등을 통하여 웹 검색 이용자들의 검색 행태를 분석하였다는 데에 의의가 있다. 그러나 웹 검색 이용자들은 매우 다양한 계층 및 연령으로 구성되어 있기 때문에, 기존의 국내 연구를 통한 전체 웹 검색 이용자들의 검색 행태 파악은 어려운 실정이다.

국내에서도 질의 로그를 상용 로그 분석 도구를 이용하여 분석함으로써, 검색 서비스 시스템에 입력된 질의 수를 파악하는 작업은 많이 수행되고 있다. 그러나, 질의에 포함된 단어 수, 이용자가 검토하는 검색 결과 페이지 수 등과 같은 웹 검색 이용자들의 검색 행태를 분석하기 위하여 질의 로그 형식을 설계하고, 생성된 질의 로그를 분석하는 연구는 드문 실정이다. 한편 박소연, 이준호(2002)는 웹 검색 서비스 네이버 이용자들의 검색 행태를 조사하기 위하여 로그 형식을 직접 설계하고, 세션 정의 방법, 로그 정제 및 질의 유형 분류 방법, 검색어 정의 방법 등 한글 웹 검색 서비스 시스템의 질의 로그 분석에 필요한 방법론을 제시하였다. 또한 이들은 웹 검색 서비스 네이버에서 하루 동안 생성된 질의 로그 중 일부를 분석하여, 이용자들의 웹 문서 검색 행태를 조사하였다.

본 연구는 네이버 이용자의 웹 문서 검색 행태에 대한 박소연, 이준호의 2002년 연구(이하 2002년 연구)의 후속 연구로서 2002년 연구에서 이용한 질의 로그 분석 방법론을 보완하여 제시하고자 한다. 또한 본 연구는 질의 로그 수집의 기간을 하루에서 일주일로 연장하고, 수집 범위를 웹 문서 검색과 더불어 통합 검색, 디렉토리 검색과 같은 다양한 검색 유형들로 확대하여 네이버 웹 검색 이용자들의 전반적인 검색 행태를 파악하고자 하였으며, 특히 재검색 질의의 세부적인 분석을 통하여 이용자들의 질의 변경 행태를 조사한다.

3.1 연구 방법

본 연구에서는 웹 검색 이용자들의 검색 행태를 파악하기 위하여 질의 로그를 분석하였다. 분석 대상이 된 질의 로그는 2003년 1월 5일부터 11일까지 웹 검색 서비스 네이버에서 생성되었으며, 본 연구에서는 이들 중에서 일부에 대한 분석을 수행하였다. 네이버는 대중성이나 인지도에 있어서 국내 주요 웹 검색 서비스로 인정받고 있다. 즉, 웹사이트 평가 및 트래픽 분석업체인 인터넷 매트릭스(http: //www.internetmatrix.co.kr)에 의하면 네이버는 2003년 2월 1개월 동안 국내 네티즌들이 가장 많이 방문하는 사이트 중 3위를 차지하였으며, 네이버는 2002년 12월 한국인터넷기업협회와 한국기자협회가 선정한 올해의 인터넷기업 대상을 수상하였다. 따라서, 네이버 질의 로그를 분석함으로써 국내 네티즌들의 전반적인 웹 검색 행태를 추측할 수 있다.

네이버는 디렉토리 검색, 웹 문서 검색, 지식iN 검색, 일본 웹 검색, 백과사전 검색, 뉴스 검색, 이미지 검색 등을 개별적으로 지원하고 있으며, 또한 이들 검색 결과들을 통합하여 보여주는 통합 검색을 제공하고 있다. 본 연구에서는 이들 중에서 가장 큰 비중을 차지하는 통합 검색, 디렉토리 검색, 웹 문서 검색에 대한 질의 로그를 분석 대상으로 하였다. 네이버의 초기 페이지에서 기본 검색은 통합 검색으로 설정되어 있으며, 이러한 설정은 이용자가 풀다운 메뉴에서 웹 문서 검색이나 디렉토리 검색을 선택함으로써 웹 문서 검색과 디렉토리 검색으로 변경될 수 있다.

질의 로그는 온라인 정보 검색 시스템과 이를 사용하는 이용자의 상호 작용에 대한 전자적인 기록으로서, 질의 로그 분석은 정보 검색 분야에서 이용자의 검색 행태 연구를 위한 합리적이고 객관적인 방법으로 인정 받고 있다(Jansen and Pooch 2000). 질의 로그 분석은 웹이 등장하기 이전부터 도서관의 OPAC(Online Public Access Catalog) 시스템(Ballard 1994), 실험적 정보 검색 시스템 등 다양한 환경에서 활용되어 왔다. 즉, 연구자들은 정보 검색 시스템, 이러한 시스템에 대한 인간의 이용, 그리고 시스템이 어떻게 이용되는가에 대한 인간의 이해를 개선할 목적으로 질의 로그 자료를 이용하고 있다(Peter 1993, 37).

질의 로그를 분석한 국외 선행 연구들을 검토한 결과, 아직까지 일반화된 용어와 방법론이 정착되어 있지 않다는 문제점을 발견하였다. 예를 들어 동일한 용어가 연구마다 약간씩 다른 의미로 사용되며, 로그를 처리하는 방식에 있어서도 연구마다 차이가 있음을 확인하였다. 또한 이들 연구들은 영어 문서를 검색하는 시스템을 대상으로 하였기 때문에, 이들 연구의 방법론을 한글 검색 질의 로그 분석에 적용할 경우 문제점이 발생한다. 따라서 다음에서는 일반적인 질의 로그 분석에 필수적인 세션 정의 방법, 로그 정제, 질의 유형 분류, 그리고 한글 검색 질의 로그 분석에 필수적인 검색어 정의에 대하여 기술한다.

3.1.1 세션 정의 방법

일반적으로 세션은 한 명의 이용자가 단일한 정보 요구를 지니고 처음 검색을 시작하여 검색을 종료하기까지의 일련의 과정으로 정의된다. 선행 연구들은 세션에 관한 일반적인 정의에는 동의하나, 검색 질의들을 세션으로 구분함에 있어서 상이한 방법을 적용하고 있으며, 익사이트의 질의 로그를 분석한 Spink et al.(2001)의 연구, 알타비스타의 로그를 분석한 Silverstein et al.(1999)의 연구와 로이터, 알타비스타, 익사이트 로그를 분석한 He & Goker(2000)의 연구가 그 대표적인 예이다.

Spink et al.(2001)은 세션의 정의를 위하여 익사이트 서버가 할당한 이용자 식별자를 이용하였다. 즉, 임의의 컴퓨터가 브라우져를 통하여 익사이트에 검색을 요청할 때, 익사이트는 쿠기를 생성하여 검색을 요청한 컴퓨터에게 쿠기의 소멸 시간을 지정하지 않은 상태로 전달한다. 이후부터 이 쿠기는 이용자 식별자로 사용되며, 검색을 요청한 컴퓨터의

브라우져들이 모두 종료될 때 소멸된다. 따라서 이 연구에서 세션은 일 초가 될 수도 있고, 몇 시간이 될 수도 있다. Spink et al.이 사용한 세션 설정 방법은 공공 장소에 위치한 하나의 컴퓨터를 다수의 사용자가 이용할 경우 또는 다수의 컴퓨터들이 하나의 프록시로 설정되어 있는 경우, 하나의 세션에 포함되는 질의 수가 과다해지는 문제점을 지니고 있다 .

He & Goker(2000)는 로이터와 익사이트 로그를 대상으로 세션의 시간 간격을 1분부터 50분까지 달리한 분석을 수행한 후, 10분에서 15분이 웹 검색 로그의 최적의 세션 간격이라고 제시하였다. 그러나 He & Goker의 세션 정의 방법은 비교적 소규모의 로그에만 사용되었고 대규모의 웹 트랜잭션 로그를 이용하여 검증되지 않았으며, 이들이 제시한 결론은 엄격한 통계 분석에 기초한 것이 아니므로 본 연구에 적용시키지 않았다.

한편 Silverstein et al.(1999)은 세션이 일정 기간 동안의 일련의 검색 과정이기 때문에, 이용자가 특정한 검색 목적을 다른 검색 목적으로 전환하게 되는 경우 시간적 공백이 발생하는 점에 착안하였다. 즉 Silverstein et al.은 검색을 요청한 컴퓨터에게 쿠기를 전달할 때, 이 쿠기가 5분 후에 소멸되도록 지정하였다. 또한, 5분 이내에 다시 검색이 요청될 때, 동일한 쿠기를 5분 후에 소멸되도록 지정하여 전달하였다. 따라서 이용자가 5분 동안 질의를 입력하지 않으면 새로운 세션이 정의되며, 5분 이내에 질의를 입력하면 기존 세션이 연장된다. 다시 말해 이들은 5분 내에 새로운 질의가 입력될 경우 이전 세션을 연장하는 세션 정의 방법을 제시하였다. 본 연구는 Spink et al.이 사용한 세션 정의 방법에 분명한 오류가 있다고 판단하고, Silverstein et al.이 사용한 세션 정의 방법을 사용하였다.

3.1.2 로그 정제

질의 로그들로부터 이용자가 정상적으로 입력한 질의라고 판단하기 어려운 다수의 로그들을 발견하였다. 대부분의 국외 선행 연구들은 로그 정제 과정을 수행하지 않고, 이러한 비정상적인 질의들을 분석 대상에 포함시킴으로써 로그 분석 결과에 오류를 포함하고 있다. 예를 들어 Silverstein et al.(1999)은 질의와 검색어 분석 시 이러한 극단치(outliers)들을 포함시킨 결과 편차와 평균이 지나치게 과대 평가되는 문제점이 발생되었다.

비정상적인 질의 로그들이 생성되는 이유는 다음과 같이 크게 3가지로 구분될 수 있다. 첫째, 이용자가 질의를 입력한 후 네트워크의 속도 저하, 검색 시스템의 과부하 등의 이유로 검색 결과 화면의 생성이 다소 지연되면, 이용자는 새로고침 버튼을 클릭하거나 재검색을 수행하는 경향이 있다. 이때 이용자가 검색 결과를 받지 않은 질의에 대해서도 로그가 생성되며, 동일한 질의에 대한 로그가 연속해서 파일에 기록된다. 본 연구에서는 이용자가 검색 결과를 받지 않았다는 사실을 기록하도록 질의 로그를 설계함으로써, 이러한 질의에 대한 로그들을 제거하였다.

둘째, 3.1.1절에서 설명된 것처럼 본 연구에서 사용한 세션 정의 방법이 현재까지 알려진 최선의 방법일 지라도, 이 방법 역시 문제점이 있음을 발견하였다. 즉, 일부 이용자들은 5분의 휴식이나 공백 이후, 이전 질의의 검색 결과를 기반으로 문서 질의 또는 반복 질의를 수행하였다. 이러한 경우 새로운 세션이 생성되며, 이 세션은 입력 질의를 포함하지 않고 문서 질의와 반복 질의만으로 구성된다. 본 연구에서는 이러한 세션을 비정상적으로 간주하고, 로그 파일로부터 제거하였다.

셋째, 질의 로그들을 살펴보면 프로그램이 자동으로 입력한 질의라고 판단되는 다수의 로그들이 존재한다. 예를 들면, 하나의 세션에 수백 수천 개의 입력 질의, 문서 질의, 반복 질의가 포함된 경우를 발견할 수 있다. 본 연구에서는 질의 로그에 대한 수작업 검토와 더불어 세션 내에서 연속된 질의 쌍에 대하여 질의가 입력된 시간의 간격을 계산하고, 이러한 시간 간격들에 대한 통계 분석을 수행하였다. 그리고, 이러한 분석을 기반으로 프로그램에 의해 자동으로 생성되었을 가능성이 높다고 판단되는 세션들을 로그 파일로부터 제거하였다.

3.1.3 검색어 정의

검색어는 질의를 구성하는 기본 단위로서, 영어의 경우 빈칸, 마침표, 쉼표, 개행 문자 등과 같은 공백 문자(blank character) 들에 의해 구분되는 일련의 문자 또는 숫자로서 정의된다. 그러나, 한글은 복합 명사를 구성하는 단일 명사들 사이의 띄어쓰기를 자유롭게 규정하고 있기 때문에, 검색어를 영어의 경우에서처럼 정의할 경우 문제점이 발생한다. 즉, 정보검색시스템과 정보 검색 시스템은 동일한 질의임에도 불구하고, 서로 다른 색인어들을 포함하고 있는 것으로 인식된다. 따라서 한글 질의 로그 분석에서 검색어에 대한 통계를 추출할 경우, 검색어에 대한 정확한 정의가 요구된다. 본 연구에서는 첫째, 영어의 경우와 유사하게 어절 단위로 검색어를 인식한 경우, 둘째, 어절을 형태소 단위로 분리한 후, 각각의 형태소를 검색어로 인식한 경우 모두에 대하여 분석을 수행하였다.

3.1.4 질의 유형 분류

질의는 세션의 구성 요소로서, 하나 이상의 검색어들로 구성된다. 본 연구에서는 전체 질의들을 최초 질의와 재검색 질의로 구분하고, 또한 재검색 질의를 다음과 같이 4가지 유형으로 분류하였다. 따라서 전체 질의 수는 최초 질의 수, 재검색 질의 수의 합과 동일하며, 재검색 질의 수는 입력 질의 수, 전환 질의 수, 문서 질의 수, 반복 질의 수의 합과 동일하다.

입력 질의: 이용자가 검색창에 직접 입력한 스트링으로서 검색어들로 구성된다. 하나의 세션에 포함된 재검색 입력 질의들은 검색어 추가 질의, 검색어 삭제 질의, 검색어 추가 및 삭제 질의, 이전 질의와 중복된 검색어를 포함하지 않는 변경 질의, 그리고 동일 질의로

구분될 수 있다. 또한, 동일 질의는 이전 질의에 포함된 검색어들만으로 구성된 질의를 의미하며, 띄어 쓰기가 변경된 질의, 검색어 순서가 변경된 질의, 그리고 불용어 제거 후 이전 질의와 동일한 검색어들을 포함하는 질의로 세분화될 수 있다. 여기에서 검색어는 형태소 단위로 인식되었다.

전환 질의: 전체 질의는 검색 유형별 분류에 의하여 통합 검색, 디렉토리 검색, 웹 문서 검색으로 구분될 수 있다. 네이버 서비스가 제공하는 검색 결과 화면에는 다른 검색 유형으로 전환할 수 있는 탭 버튼이 존재하며, 전환 질의는 사용자가 이러한 탭 버튼을 클릭한 경우에 생성된다. 예를 들면, 사용자가 초기 검색으로 디렉토리 검색을 수행하고, 그 결과 화면에서 웹 문서 탭 버튼을 클릭한 경우, 디렉토리 검색에서 웹 문서 검색으로의 전환 질의가 생성된다.

문서 질의: 네이버 서비스에서 웹 문서 검색을 수행한 후, 그 결과 화면에는 관련문서검색으로 표기된 링크가 존재하며, 문서 질의는 사용자가 이 링크를 클릭한 경우에 생성된다. 문서 질의는 검색어를 포함하지 않는 질의로서, 이용자가 지정한 웹 문서 전체가 질의로 사용된다.

반복 질의: 일반적으로 웹 검색 시스템들은 디렉토리와 웹 문서에 대한 검색을 수행한 후, 질의에 적합할 가능성이 높은 상위 10개 정도의 문서들을 검색 결과로서 이용자에게 제공한다. 이때 이용자가 상위로부터 11번째 이후의 문서들을 보고자 할 경우, 즉 다음 결과 화면을 보고자 요청할 때 반복 질의가 발생한다.

3.2 로그 분석 결과

3.2.1 질의 수 분석

본 연구는 2003년 1월 5일부터 11일까지 웹 검색 서비스 네이버에서 생성된 질의 로그들 중 일부를 분석하였다. 분석 대상이 된 로그 파일에 포함된 전체 세션 수는 22,562,531개이고, 전체 질의 수는 40,746,173개이었다. 전체 질의들을 최초 질의와 재검색 질의로, 재검색 질의는 입력 질의, 전환 질의, 반복 질의, 문서 질의로 구분한 후, 각 유형별로 질의 수를 살펴보면 <표 3.1>과 같다. 즉, 전체 질의 중 최초 질의 수는 22,562,531개(55.37%)이었고, 재검색 질의 수는 18,183,642개(44.63%)이었다. 재검색 질의 중 입력 질의 수는 15,037,296개(82.7%), 반복 질의 수는 2,471,414개(13.59%), 전환 질의 수는 433,026개(2.38%), 문서 질의 수는 241,906개(1.33%)이었다. 전체 질의 중 최초 질의 수가 절반 이상이며, 재검색 질의 중 입력 질의가 가장 큰 비중을 차지함을 알 수 있다.

<표 3.1> 질의 유형별 질의 수 분석

구 분		질의 수	비율(%)
최초질의		22,562,531	55.37
재검색 질의	입력 질의	15,037,296	36.91
	전환 질의	433,026	1.06
	반복 질의	2,471,414	6.07
	문서 질의	241,906	0.59
총 계		40,746,173	100.00

<표 3.2> 최초 질의의 검색 유형별 질의 수 분석

구 분	질의 수	비율(%)
통합검색 (디렉토리 검색) (웹 문서 검색)	22,549,768 (9,770,489) (11,304,742)	99.94
디렉토리 검색	3,536	0.02
웹 문서 검색	9,227	0.04
총 계	22,562,531	100.00

<표 3.3> 전체 질의의 검색 유형별 질의 수 분석

구 분	질의 수	비율(%)
통합검색 (디렉토리 검색) (웹 문서 검색)	36,841,650 (15,408,275) (19,311,804)	90.42
디렉토리 검색	239,067	0.58
웹 문서 검색	3,665,456	9.00
총 계	40,746,173	100.00

최초 질의들을 검색 유형별로 구분해 보면 <표 3.2>와 같다. 전체 질의 중 통합 검색은 22,549,768개(99.94%), 디렉토리 검색은 3,536개(0.02%), 웹 검색은 9,227개(0.04%) 이었다. 네이버 서비스의 초기 페이지에서 기본 검색은 통합 검색으로 설정되어 있으며, 최초 질의 중 통합 검색이 가장 많은 비중을 차지한다는 사실은 이용자들이 수동적이어서 시스템에 의해 설정된 환경을 변경하지 않거나, 또는 통합 검색의 결과에 가장 만족하기 때문일 것으로 추정된다. <표 3.3>은 전체 질의들을 검색 유형별로 구분한 결과이며, 역시 통합

검색이 가장 큰 비중을 차지하고 있다. 그리고, <표 3.3>은 통합 검색이나 웹 문서 검색에 비해 비교적 양질의 정보들을 선택하여 분류해 놓은 디렉토리 검색의 이용률이 상대적으로 낮음을 보여준다.

한편, 이용자가 통합 검색을 수행할 경우, 디렉토리 검색 결과에 10건 이상의 문서가 포함되면 웹 문서 검색을 수행하지 않으며, 또한 디렉토리 검색 및 웹 문서 검색에 의해 0건의 문서가 검색될 수 있다. 따라서 통합 검색 결과의 일부로서 디렉토리 검색 또는 웹 문서 검색 결과가 포함되지 않는 경우가 존재한다. <표 3.2>, <표 3.3>에서 괄호 안의 숫자는 통합 검색들 중에서 디렉토리 검색과 웹 문서 검색의 결과가 출력된 질의 수이다. 즉, 통합 검색들 중에서 디렉토리 검색과 웹 문서 검색의 결과가 출력된 비율이 최초 질의의 경우 각각 43.33%, 50.13%이며, 전체 질의의 경우 각각 41.82%, 52.42%임을 알 수 있다.

<표 3.4> 재검색 질의의 검색 유형별 질의 수 분석

구 분	질의 수			
	입력 질의	전환질의	반복 질의	문서 질의
통합 → 통합	13,981,286	0	0	0
통합 → 디렉토리	7,089	136,168	0	0
통합 → 웹 문서	17,112	164,486	803,211	100,243
디렉토리 → 통합	21,590	20,584	0	0
디렉토리 → 디렉토리	55,484	0	11,152	0
디렉토리 → 웹 문서	214	32,719	10	0
웹 문서 → 통합	211,794	56,628	0	0
웹 문서 → 디렉토리	3,197	22,441	0	0
웹 문서 → 웹 문서	739,530	0	1,657,041	141,663

<표 3.4>는 재검색 질의를 검색 유형별로 분석한 결과를 보여 준다. 이용자가 질의를 다시 입력하는 입력 질의의 경우, 통합 검색이 절대적인 비중을 차지하고 있다. 질의를 변경하지 않고 탭 버튼을 클릭하여 검색하는 전환 질의들 중에서 동일한 검색 유형으로의 전환 질의는 이용자에게 이전 검색 결과와 동일한 검색 결과를 제공한다. 그러나, <표 3.4>는 통합 검색에서 통합 검색으로의 전환이 다수를 차지하고 있으며, 디렉토리 검색에서 디렉토리 검색, 웹 검색에서 웹 검색으로의 전환도 많이 수행되었음을 보여준다. 이처럼 유용하지 않은 동일 검색 유형으로의 전환이 빈번하게 수행되는 이유는 이용자가 전환 질의의 성격을 정확히 이해하지 못하고 있기 때문인 것으로 추정된다.

<표 3.5>는 전제 질의의 세션당 질의 수에 대한 기술 통계를 보여 준다. 이용자들은 세션당 평균 1.8개의 질의를 수행하였다. <표 3.6>은 전체 질의를 질의 유형별로 구분한 후, 각 질의 유형별 세션당 질의 수에 대한 기술 통계를 보여 준다. 최초 질의는 세션 시작 질의이므로 세션당 평균이 1회이며, 최초 질의의 총계는 전체 세션 수와 동일하다. 재검색 질의들 중에서 관련문서검색으로 지칭되는 문서 질의와 전환 질의는 각각 세션당 평균 0.02회, 0.04회 수행되었으며, 이로부터 이용자들이 문서 질의와 전환 질의를 드물게 사용하고 있음을 알 수 있다. 반복 질의는 세션당 평균 0.11회 수행되었으며, 이는 이용자들이 평균적으로 출력하는 결과 화면이 약 1.11개 페이지임을 의미한다. 입력 질의와 반복 질의의 표준 편차가 비교적 큰 이유는 일부 이용자가 많은 수의 입력 질의 및 반복 질의를 수행하기 때문이다. <표 3.7>에는 재검색 질의 중 입력 질의의 유형별 세션당 질의 수에 대한 기술 통계가 요약되어 있다. 이 표로부터 이용자는 검색어를 추가 또는 삭제하기 보다는 이전 질의를 전적으로 변경하는 경우가 많음을 알 수 있다. 한편, <표 3.8>은 검색 유형별 세션당 질의 수에 대한 기술 통계를 보여준다.

<표 3.5> 전체 질의의 세션당 질의 수에 대한 기술 통계

구 분	평 균	중간값	표준편차	최소값	최대값	질의 수
전체 질의	1.8	1	2.03	1	147	40,746,173

<표 3.6> 질의 유형별 세션당 질의 수에 대한 기술 통계

구 분		평 균	중간값	표준편차	최소값	최대값	질 의 수
최초 질의		1.00	1	0.00	1	1	22,562,531
재검색 질의	입력 질의	0.67	0	1.56	0	140	15,037,296
	전환 질의	0.02	0	0.18	0	28	433,026
	반복 질의	0.11	0	0.95	0	133	2,471,414
	문서 질의	0.01	0	0.25	0	103	241,906

<표 3.7> 재검색 질의 중 입력 질의의 유형별 세션당 질의 수에 대한 기술 통계

구 분	평 균	중간값	표준편차	최소값	최대값	질의 수
검색어 추가	0.066	0	0.30	0	28	1,500,401
검색어 삭제	0.039	0	0.22	0	23	888,761
검색어 추가 및 삭제	0.067	0	0.37	0	80	1,517,102
변경	0.478	0	1.19	0	134	10,791,403
동일(띄어 쓰기 변경)	0.006	0	0.09	0	20	131,411
동일(검색어 순서 변경)	0.001	0	0.03	0	4	14,529
동일(불용어 제거 후)	0.009	0	0.12	0	32	193,689

<표 3.9>는 전체 질의의 세션당 질의 수의 분포를 보여 주며, <표 3.10>은 전체 질의를 질의 유형별로 구분한 후, 각 질의 유형별 세션당 질의 수의 분포를 보여 준다. 전체 세션의 83.52%가 2개 이하의 질의를 포함하고 있다. 이는 이용자들이 정보의 발견을 위하여 웹 검색에 소비하는 시간과 노력이 많지 않음을 시사하며, 이에 대한 이유로서 다음과 같은 사항들을 고려할 수 있다. 첫째, 웹 검색을 통하여 이용자가 접근하고자 하는 정보들이 매우 일반적이기 때문에, 웹 상에 많은 수의 적합 문서들이 존재한다. 따라서 이러한 적합 문서들의 일부가 웹 검색 시스템에 의해 쉽게 검색될 수 있다. 둘째, 웹 상에 적은 수의 적합 문서들이 존재할 지라도, 웹 검색 시스템의 성능이 우수하기 때문에 이용자가 만족하는 검색 결과를 제공한다. 셋째, 웹 검색 이용자는 정보 발견에 있어서 절실한 필요성을 지니고 있지 않기 때문에, 정보의 발견을 도중에 쉽게 포기하는 경향이 있다. 한편, <표 3.11>은 세션당 질의 수의 분포를 검색 유형별로 보여 준다.

<표 3.8> 검색 유형별 세션당 질의 수에 대한 기술 통계

구 분	평 균	중간값	표준편차	최소값	최대값	질의 수
통 합	1.65	1	1.51	0	143	36,841,650
디렉토리	0.01	0	0.16	0	89	239,067
웹 문서	0.16	0	1.23	0	143	3,665,456

<표 3.9> 전체 질의의 세션당 질의 수 분포

	1개	2개	3개	4개	5개	6개 이상
전체 질의		3,681,717	1,581,043	796,304	447,179	892,559

<표 3.10> 질의 유형별 세션당 질의 수 분포

		1개	2개	3개	4개	5개	6개 이상
최초 질의		22,562,531	0	0	0	0	0
재검색 질의	입력 질의	3,726,846	1,550,794	734,567	383,547	219,799	394,001
	전환 질의	262,662	53,099	12,285	3,502	1,261	952
	반복 질의	360,157	158,579	86,298	54,285	35,205	109,903
	문서 질의	77,182	19,371	7,900	4,356	2,744	7,075

<표 3.11> 검색 유형별 세션당 질의 수 분포

	0개	1개	2개	3개	4개	5개 이상
통 합	11,099	15,707,812	3,715,739	1,500,044	707,055	920,782
디렉토리	22,398,753	105,611	38,429	10,291	4,740	4,707
웹 문서	21,240,146	398,551	493,164	121,450	79,950	234,270

3.2.2 검색어 수 분석

본 절에서는 최초 질의와 재검색 질의 중 입력 질의에 포함된 검색어 수에 대한 분석 결과를 기술한다. <표 3.12>, <표 3.13>은 검색 유형별 질의당 검색어 수에 대한 기술 통계를 각각 어절 단위와 형태소 단위로 보여 준다. 검색어가 어절 단위로 정의되는 경우 질의당 평균 검색어 수는 1.13개이었고, 검색어가 형태소 단위로 정의되는 경우 질의당 평균 검색어 수는 2.03개이었다. 이러한 결과를 통하여 웹 검색 이용자들은 매우 적은 수의 검색어로 구성된 단순한 질의를 수행하는 경향이 있음을 알 수 있다. 또한, <표 3.12>, <표 3.13>은 이용자가 다른 검색 유형에 비하여 웹 문서 검색을 위해 보다 많은 수의 검색어를 입력하는 경향이 있음을 보여준다.

<표 3.12> 검색 유형별 질의당 어절 단위 검색어 수에 대한 기술 통계

구 분		평 균	중간값	표준편차	최소값	최대값
최초 질의	통 합	1.09	1	0.38	1	71
	디렉토리	1.12	1	0.47	1	10
	웹 문서	1.17	1	0.62	1	16
재검색 입력 질의	통 합	1.18	1	0.55	1	49
	디렉토리	1.19	1	0.52	1	10
	웹 문서	1.50	1	0.93	1	28
최초 질의 + 재검색 입력 질의		1.13	1	0.48	1	71

<표 3.13> 검색 유형별 질의당 형태소 단위 검색어 수에 대한 기술 통계

구 분		평균	중간값	표준편차	최소값	최대값
최초 질의	통 합	1.83	1	2.13	1	930
	디렉토리	1.92	1	2.59	1	100
	웹 문서	2.21	2	3.93	1	210
재검색 입력 질의	통 합	2.26	2	2.92	1	665
	디렉토리	2.29	2	2.45	1	100
	웹 문서	3.38	2	5.10	1	280
최초 질의 + 재검색 입력 질의		2.03	2	2.56	1	930

한편 <표 3.14>, <표 3.15>는 검색 유형별 질의당 검색어 수의 분포를 어절 단위와 형태소 단위로 보여준다. 검색어가 어절 단위로 정의되는 경우 하나의 검색어만을 포함하는 질의가 90.42%이었으며, 검색어가 형태소 단위로 정의되는 경우 하나의 검색어만을 포함하는 질의가 47.67%이었다. 따라서 질의에 대한 형태소 분석의 수행 여부와 관계없이 하나의 검색어로 구성된 질의를 수행하는 검색 서버의 별도 구축이 바람직함을 알 수 있다.

<표 3.14> 검색 유형별 질의당 어절 단위 검색어 수 분포

구 분		1개	2개	3개	4개	5개	6개 이상
최초 질의	통합	21,021,711	1,206,047	239,493	53,041	16,584	12,892
	디렉토리	3,238	225	47	18	5	3
	웹 문서	8,210	676	232	67	21	21
재검색 입력 질의	통합	12,392,989	1,340,589	345,206	87,038	27,900	20,948
	디렉토리	55,987	7,712	1,606	325	91	49
	웹 문서	517,033	153,293	57,669	17,750	6,363	4,748

<표 3.15> 검색 유형별 질의당 형태소 단위 검색어 수 분포

구 분		1개	2개	3개	4개	5개	6개 이상
최초 질의	통합	11,332,779	8,625,673	985,866	928,405	25,764	651,281
	디렉토리	1,871	1,196	159	169	5	136
	웹 문서	4,420	3,320	417	503	34	533
재검색 입력 질의	통합	6,319,872	5,074,650	880,147	978,034	38,556	923,411
	디렉토리	29,095	21,340	4,790	5,594	312	4,639
	웹 문서	236,224	201,809	64,133	104,096	5,314	145,280

제 4 장 정보 검색 시스템의 기능

정보 검색 시스템은 사용자가 원하는 정보를 빠르고 정확하게 검색할 수 있도록 다양한 기능들을 제공해야 한다. 이러한 기능들은 다음과 같이 크게 4가지로 분류될 수 있다. 첫째, 질의 작성 관련 기능들로서 사용자의 정보 요구를 보다 정확하게 표현하기 위해 불리안 연산, 근접 연산, 절단 연산, 유사어 연산, 색인어 조회, 필드 지정이 사용될 수 있다. 또한, 불리안, 근접, 절단 연산자에 익숙하지 않은 사용자들은 자연어 질의 기능을 사용함으로써 편리하게 질의를 작성할 수 있으며, 질의 보관 기능은 동일한 질의의 반복적인 수행에 편리를 제공한다.

둘째, 색인 DB 생성 관련 기능들로서 정보 시스템 연동, 문서 형식 변환, 색인 방법 선택이 이에 속한다. 정보 시스템 연동 기능은 다양한 정보 시스템에 산재해 있는 문서들을 위치에 관계없이 색인하기 위해 요구된다. 일반적으로 이러한 문서들은 다양한 형식으로 저장되어 있으며, 이러한 문서들을 색인하기 위해서는 TXT 형식으로의 변환이 필요하다. 또한, 문서의 특성에 따른 서로 다른 색인 방법의 사용은 검색 효과의 개선에 도움을 준다.

셋째, 검색 결과의 출력을 위해 사용되는 기능들로서, 사용자들이 검색 결과들로부터 적합 문서를 쉽게 발견하도록 도와 준다. 일반적으로 주어진 질의에 대해 검색된 문서의 수가 많기 때문에 사용자가 검색된 모든 문서들을 검토하여 적합 문서를 발견하는 것은 매우 어려운 일이다. 따라서 정보 검색 시스템은 검색된 문서들 중에서 사용자가 원하는 적합 문서를 발견하기 쉬운 형태로 제공하는 것이 필요하다. 즉, 하이라이팅과 문서 요약은 검색된 문서의 내용을 빠르게 판단할 수 있도록 하며, 문서 분류는 동일한 시간 내에 사용자들이 보다 많은 문서들을 검토할 수 있도록 지원한다. 또한, 문서의 정렬 방법에 따라 사용자의 만족도를 높일 수 있다.

넷째, 질의 수정에 관련된 기능들로서 검색어 추천, 적합성 피드백, 유사 문서 검색이 이에 속한다. 일반적으로 사용자들은 광범위한 정보 요구를 지니고 정보 검색 시스템에 접근하여 그 요구들을 질의로 표현한다. 정보 검색 시스템에 입력된 이러한 질의는 매우 임의적이며 부정확하기 때문에, 양질의 검색 결과를 얻는 것은 매우 어려운 일이다. 즉, 인터넷 정보 검색 시스템에 입력되는 질의들을 살펴보면 일반적으로 1～2개의 단어들을 포함하고 있다. 이처럼 모호한 질의들을 처리하여 양질의 검색 결과를 생성하는 일은 매우 어려운 일이며, 사용자는 이러한 질의를 수정하여 재검색을 수행하는 것이 바람직하다.

4.1 질의 작성

4.1.1 불리안 연산

불리안 연산은 사용자의 정보 요구를 표현하기 위해 다수의 색인어를 논리적으로 연결할 수 있는 수단을 제공한다. 대표적인 불리안 연산자로는 AND, OR, NOT이 사용되며, 이러한 연산자들은 각각 교집합, 합집합, 차집합 함수에 의해 구현된다. 한편, 일부 정보 검색 시스템만이 배타적 논리합 연산자 XOR를 지원한다. 이는 대부분의 사용자들이 배타적 논리합 연산을 이해하지 못하며, 또한 배타적 논리합 연산자는 AND, OR, NOT의 조합에 의해 대치될 수 있기 때문이다.

일반적으로 불리안 연산자들 AND, OR, NOT 사이에는 연산에 있어서 우선 순위가 있다. 즉, NOT이 우선적으로 연산되고, 그 다음에 AND와 OR 순서로 연산이 수행된다. 이러한 기본적인 우선 순위를 임의적으로 조정하기 위해서 괄호가 사용되며, 괄호 안의 연산자들이 우선적으로 연산된다. 예를 들어 "(AA OR BB) AND CC"와 "AA OR BB AND CC"는 서로 다른 문서들을 검색한다. 첫 번째 질의는 AA 또는 BB를 포함하면서 CC를 포함하는 문서들을 검색하며, 두 번째 질의는 BB와 CC를 포함하거나 또는 AA를 포함하는 문서들을 검색한다.

불리안 검색의 특별한 형태로서 "M of N" 논리가 사용될 수 있다. 즉, 사용자는 원하는 문서에 포함되어 있을 가능성이 있는 N개의 색인어들을 나열하고, 이들 중에서 M개 이상의 색인어들을 포함하는 문서들을 검색하도록 질의를 작성할 수 있다. 예를 들면, "M of N" 논리를 사용하여 "AA", "BB", "CC" 세 개의 색인어들 중에서 2개 이상의 색인어들을 포함하는 문서들을 검색할 수 있다. "M of N" 논리를 사용한 질의는 AND와 OR 연산자를 사용하여 동일한 질의로 변환될 수 있다. 예를 들면, 위의 예제는 (AA AND BB) OR (AA AND CC) OR (BB AND CC)라는 질의로 변환될 수 있다.

4.1.2 근접 연산

근접 연산은 문서 내에서 색인어들 사이에 허용되는 거리를 제한하기 위해 사용되며, 일반적으로 정보 검색 시스템들은 다음과 같이 ORDER와 NEAR 두 가지 형태의 근접 연산자를 지원한다.

NEAR "*거리*" "*단위*" (*색인어*$_1$, *색인어*$_2$, …, *색인어*$_n$)
ORDER "*거리*" "*단위*" (*색인어*$_1$, *색인어*$_2$, …, *색인어*$_n$)

여기서 인자 "*거리*"는 정수로서 색인어들 사이에 허용되는 거리를 지정하고, 인자 "*단위*"의 값으로는 문자, 단어, 문장 또는 문단들 중의 하나가 사용된다. "*단위*"가 문장 또는 문단일

경우 “*거리*”의 값으로 0이 사용될 수 있으며, 이는 색인어들이 동일한 문장 또는 문단에 존재함을 의미한다. 많은 경우에 문서를 분석하여 문장과 문단을 구분하는 일은 매우 어려운 일이다. 따라서, 근래의 정보 검색 시스템들을 살펴보면 “단위”의 값으로서 단어만이 사용되고 있다.

근접 연산자 NEAR는 색인어들 사이의 거리만을 제한하며, 근접 연산자 ORDER는 색인어들 사이의 거리를 제한함과 동시에 색인어들이 문서에 출현하는 순서를 지정한다. 즉, ORDER 연산자를 사용할 경우, 질의에 나열된 순서대로 색인어들이 출현하는 문서만이 검색된다. 이러한 근접 연산자들의 사용은 검색의 정확성을 향상시킨다. 예를 들어, 사용자가 “반도체 설계”에 대한 문서를 원할 경우, “반도체”와 “설계” 두 개의 색인어들에 대한 AND 연산을 수행하는 대신에 근접 연산을 수행하는 것이 바람직하다. 이는 문서 내에서 두 개 색인어들의 출현 위치가 근접할수록 특정한 개념 “반도체 설계”와 연관되어 있을 가능성이 높기 때문이다.

4.1.3 절단 연산

절단 연산은 검색어의 일부에 임의의 문자 또는 문자열을 포함할 수 있도록 지정한다. 이 결과 절단 연산자를 포함하는 검색어는 다수의 색인어와 일치되며, 정보 검색 시스템은 이러한 다수의 색인어들 중에서 하나 이상을 포함하는 문서를 검색한다. 즉, 절단 연산자를 포함하는 검색어와 일치되는 다수의 색인어들을 OR 연산자로 연결한 질의와 동일한 검색 결과를 제공한다. 예를 들어, 절단 연산자를 포함하는 “COMPUT*”를 검색어로 사용하면, “COMPUT”로 시작하는 모든 색인어가 일치되므로 “COMPUTE”, “COMPUTER”, “COMPUTING” 등의 색인어를 포함하는 문서들이 검색된다.

일반적으로 정보 검색 시스템들은 고정 길이 절단 연산과 가변 길이 절단 연산을 지원한다. 고정 길이 연산자를 포함하는 검색어는 연산자 위치에 임의의 문자를 포함하는 색인어와 일치되며, 가변 길이 연산자를 포함하는 검색어는 연산자 위치에 길이에 관계없이 임의의 문자열을 포함하는 색인어와 일치된다. 또한, 절단 연산은 절단되는 위치에 따라 좌측 절단, 우측 절단, 양측 절단, 중간 절단으로 구분될 수 있다. 좌측 절단은 검색어의 앞쪽을 절단하는 것으로, 다수의 접두어를 갖는 색인어의 검색에 유용하다. 우측 절단은 검색어의 뒤쪽을 절단하는 것으로, 단수/복수 및 다양한 어미를 갖는 색인어의 검색에 유용하다. 양측 절단은 검색어의 앞쪽과 뒤쪽을 동시에 절단함으로써 중간 부분이 일치하는 색인어를 검색한다. 그리고, 중간 절단은 검색어의 중간에 절단 표시를 사용하며, 절단 표시의 앞뒤 문자열과 일치하는 색인어를 검색한다.

4.1.4 자연어 질의

정보 검색 서비스를 제공하여 축적된 사용자 질의를 살펴보면, 불리안, 근접, 절단 연산자를 포함하는 질의가 매우 적으며, 대다수의 질의들이 단순히 하나의 단어 또는 구만을 포함하고 있는 것으로 알려져 있다. 이는 정보 검색 시스템을 이용하는 일반적인 사용자들이 불리안, 근접, 절단 연산에 익숙하지 않으며, 이러한 연산들을 조합하여 효과적인 질의를 작성하는데 어려움을 겪고 있음을 의미한다. 이처럼 질의가 단순할 경우, 사용자의 정보 요구를 만족시킬 수 있는 문서를 검색하는 것은 매우 어려운 일이다.

자연어 질의는 사용자의 정보 요구를 연산자를 사용하지 않고 문장을 작성하거나 단어들을 열거함으로써 작성될 수 있다. 따라서 사용자는 형식에 구애 받지 않고 머리 속에 떠오르는 단어를 생각나는 대로 입력할 수 있기 때문에, 많은 수의 단어가 포함된 질의를 작성할 수 있다. 일반적으로 자연어 질의는 형식화 모듈을 거쳐 정보 검색 시스템의 검색 모듈이 인식할 수 있는 형태로 변환된다. 따라서 자연어 질의를 처리하는 정보 검색 시스템에서 형식화 모듈은 검색 결과의 질을 결정짓는 매우 중요한 요소이다. 한편, 현재까지 개발된 기술로서 부정문을 포함하는 자연어 질의의 형식화는 많은 문제점을 지니고 있다. 예를 들면, “최민식이 출현한 영화 중에서 한석규가 출현하지 않은 영화는?”과 같은 자연어 질의에 대하여, 대다수의 정보 검색 시스템들은 “쉬리”에 대한 문서를 검색한다.

4.1.5 유사어 연산

정보 검색 시스템에 입력되는 문서들은 전자화되어 있어야 하기 때문에, 과거에 생성된 인쇄물에 대해서는 전자화가 선행되어야 한다. 지금까지 생성된 인쇄물은 그 양이 방대하기 때문에, 많은 경우에 인쇄물의 전자화를 위하여 문자 인식 소프트웨어가 활용되고 있다. 그러나, 문자 인식 소프트웨어를 이용하여 인쇄물의 전자화를 수행할 경우, 전자화된 문서는 문자 인식 소프트웨어의 인식 오류로 인한 다수의 철자 오류를 포함하게 되며, 이는 검색을 어렵게 만드는 요인이 된다. 철자 오류의 문제는 수작업에 의해 작성되는 문서에도 존재하며, 특히 웹 문서에서는 문자 오류를 흔히 발견할 수 있다.

유사어 연산은 크게 유사 철자 연산과 유사 발음 연산으로 구분될 수 있다. 유사 철자 연산은 검색어와 유사한 철자를 지닌 색인어를 검색하며, 위에서 언급된 철자 오류를 포함하는 문서의 검색에 유용하게 사용될 수 있다. 예를 들어, 검색어 “정보검색”에 대하여 유사 철자 연산을 수행하면, “정보검삭”, “저보검색”, “정포검색”, “정보검섹” 등과 같이 유사 철자를 지닌 색인어들이 검색된다. 이러한 유사 철자 연산은 많은 경우에 *n*-gram 색인 방법을 이용하여 구현된다. *n*-gram 색인은 언어에 비의존적이기 때문에, 유사 철자 연산은 다양한 언어에서 쉽게 구현될 수 있다.

유사 발음 연산은 검색어의 발음 정보를 분석하여 유사한 발음을 지닌 색인어를 검색하며, 실세계에서 다양한 형태로 응용되고 있다. 1920년대에 미국 국립 문서국은 인구 조사 기록의 효율적인 관리를 위하여 유사 발음을 지닌 영어 인명을 검색할 수 있는 사운덱스(SOUNDEX) 알고리즘을 개발하였다. 이 알고리즘은 유사한 발음을 지닌 영어 인명들에 동일한 코드값을 부여한다. 예를 들어, “smith”와 “smyth”는 철자는 다르지만 발음이 동일하므로 동일한 코드값 “S530”이 부여된다. 이러한 사운덱스 알고리즘은 인구 조사 기록의 관리 뿐만 아니라 족보 연구, 항공사 고객 관리 시스템, 군인들의 신상 기록 카드 관리와 같은 분야에서도 널리 활용되고 있다.

한글은 서로 다른 철자가 서로 다른 발음을 갖는 소리 글자이기 때문에, 유사 발음 연산이 적용될 수 없다. 그러나, 한글의 경우 하나의 외국 단어가 한글로 다양하게 표기되는 문제점을 지니고 있다. 예를 들어, 영어 단어 “Digital”은 “디지털”, “디지탈”, “디지틀”과 같이 다양하게 한글로 표기된다. 이러한 외래어 표기의 다양성은 검색어와 색인어 사이의 불일치 문제를 야기함으로써 정보의 검색을 어렵게 만드는 요인이 되고 있다. 국내에서 개발된 코덱스 알고리즘은 사운덱스 알고리즘의 변형에 의해 개발되었으며, 외국 단어의 다양한 한글 표기들에 동일한 코드를 부여하도록 설계되었다. 즉, 코덱스 알고리즘을 “디지털”, “디지탈”, “디지틀”과 같은 단어들에 동일한 코덱스 코드 “D784”를 부여한다.

4.1.6 색인어 조회

정보 검색 시스템은 입력된 문서들로부터 색인어들을 추출하여 색인 DB를 생성한 후, 색인 DB를 조회하여 질의에 포함된 검색어와 일치되는 색인어의 존재 여부를 확인함으로써 검색을 시작한다. 조회 결과 검색어와 일치되는 색인어가 존재하지 않을 경우, 검색 결과로서 어떠한 문서도 반환되지 않는다. 따라서 사용자는 정보 검색 시스템이 지원하는 색인어 조회 기능을 사용하여 색인 DB 내의 색인어 리스트를 참조한 후, 색인 DB에 존재하는 색인어를 검색어로 사용하여 질의를 작성하는 것이 바람직하다.

색인어 조회 기능은 사용자로부터 입력된 문자열과 가장 근접한 색인어를 색인 DB에서 검색한다. 그리고, 사전적 순서에 근거하여 검색된 색인어와 전후의 색인어들을 각 색인어가 출현한 문서의 수를 의미하는 문서 빈도와 함께 출력한다. 다음은 사용자가 “comput”라는 문자열을 입력했을 때 색인어 조회 결과를 보여준다.

compromise	53
comptroller	18
compulsion	5
compulsive	22
compulsory	4
comput	

computation	265
compute	1245
computen	1
computer	10,800
computerize	18
computes	29

색인어 조회 기능은 절단 연산을 수행할 경우, 출력될 검색 결과의 예측을 가능하도록 해준다. 예를 들어, 사용자가 'compul*'이란 검색어를 입력하면 'compulsion', 'compulsive', 'compulsory'와 같은 색인어들이 검색됨을 예상할 수 있다. 또한, 색인어들의 문서 빈도 정보는 불리안 질의 작성에 매우 유용하게 활용될 수 있다. 예를 들어, 문서 빈도가 높은 색인어를 OR 연산자를 사용하여 질의에 포함시킬 경우, 검색되는 문서의 수가 매우 증가함을 예상할 수 있다. 또한, 문서 빈도가 낮은 색인어를 AND 연산자를 사용하여 질의를 포함시킬 경우, 검색되는 문서의 수가 매우 감소함을 예상할 수 있다.

4.1.7 필드 지정

일반적으로 정보 검색 시스템에 입력되는 문서는 여러 개의 필드들로 구성되어 있으며, 필드 구분이 어려울 경우 문서가 하나의 필드만으로 구성될 수도 있다. 예를 들어, 웹 문서는 “제목”, “본문”, “URL” 등의 필드들로 구성되어 있다. 정보 검색 시스템의 사용자는 검색 대상 필드를 제한하여 검색을 수행함으로써 검색 속도와 검색의 정확도를 개선할 수 있다. 즉, 질의에 검색 필드를 지정하고 검색을 수행함으로써, 검색할 데이터의 양을 감소시키고, 이에 따른 검색 시간의 단축을 얻을 수 있다. 또한 검색 필드를 제한하지 않을 경우 검색되는 문서들 중에서 많은 문서들이 검색 필드의 제한에 의해 검색 결과에서 제외되며, 이에 따라 검색의 정확도가 증가한다.

4.1.8 질의 보관

사용자가 이후의 검색을 위해 작성된 질의에 이름을 붙이고 정보 검색 시스템에 등록하는 기능을 질의 보관이라 한다. 다수의 웹 검색 사용자들은 하나 또는 두 단어만을 포함하는 매우 짧은 질의를 작성하기 때문에, 웹 검색 시스템에서 질의 보관 기능은 중요하게 인식되지 않고 있다. 그러나, 정보의 획득이 매우 중요한 전문가들이 특허, 법률 등과 같은 문서들을 검색하는 경우, 일반적으로 질의는 매우 복잡하며 많은 검색어와 연산자들을 포함하고 있다. 또한, 사용자가 정보 검색 시스템을 이용하는 행동 양식을 살펴보면 동일한 질의를 주기적으로 시스템에 입력하는 경향이 있다. 이러한 경우 사용자는 질의 보관 기능을 사용하여 질의를 정보 검색 시스템에 등록한 후, 필요한 경우에 호출하여 사용함으로써 질의 작성에 소모되는 시간을 절약할 수 있다.

4.2 색인 데이터베이스 생성

4.2.1 정보 시스템 연동

이미 구축되었거나 앞으로 새롭게 구축될 데이터베이스들을 살펴보면, 그들 나름대로의 다양한 이유에 의해서 많은 문서들이 화일 시스템, DBMS, 인트라넷, 인터넷 등의 다양한 위치에 생성되고 있다. 예를 들어, 정보 시스템 개발자들 중의 다수는 이미 검증된 안정성과 성능 때문에 문서 데이터베이스를 ORACLE이나 INFORMIX와 같은 상용 DBMS로 구축하고 있으며, 최근에는 그룹웨어 제품 사용자들이 늘면서 공공 기관, 기업, 연구소 등의 로컬 인트라넷에 많은 문서들이 그룹웨어 소프트웨어를 통해 축적되고 있다. 이러한 정보 시스템들이 관리하고 있는 문서들에 대한 검색을 지원하기 위해서 정보 검색 시스템은 정보 시스템과의 연동 방법을 제공해야 한다.

4.2.2 문서 형식 변환

최근 컴퓨터의 급속한 보급으로 많은 문서들이 아래한글, 워드, 파워포인트 등과 같은 문서 편집기를 사용하여 작성되고 있으며, 이러한 문서 편집기들은 각자 고유의 형식으로 문서들을 저장한다. 또한 웹의 사용이 급증함에 따라 방대한 양의 HTML 형식의 문서들이 작성되고 있으며, 편리한 전자 교환을 위해 PS, PDF와 같은 형식의 문서들이 생성되고 있다. 그러나, 일반적으로 정보 검색 시스템은 TXT 형식의 문서 처리를 기본으로 하고 있다. 따라서 TXT 형식 이외의 문서들에 대한 검색을 지원하기 위해서 정보 검색 시스템은 TXT 형식으로의 문서 형식 변환 기능을 지원해야 한다. 또한, 새로운 형식의 문서들을 처리하기 위해 문서 형식 변환 모듈을 정보 검색 시스템에 추가적으로 삽입할 수 있도록 하는 것이 바람직하다.

4.2.3 색인 방법 선택

정보 검색 시스템은 다양한 색인 방법들을 제공하고 있으며, 이들은 문서로부터 색인어들을 추출하거나 질의로부터 검색어들을 추출하기 위해 활용된다. 영어 문서에 대한 대표적인 색인 방법으로 스테밍을 수행하는 색인 방법과 스테밍을 수행하지 않는 색인 방법을 사용할 수 있으며, 한글 문서에 대표적인 색인 방법으로는 어절 단위 색인법과 형태소 단위 색인법을 사용할 수 있다. 이 외에도 정보 검색 시스템들은 다양한 색인 방법들을 제공하고 있으며, 정보 검색 시스템 관리자는 적절한 색인 방법을 선택함으로써 검색 결과의 질을 크게 향상시킬 수 있다.

4.3 검색 결과 출력

4.3.1 정렬 방식 선택

정렬 방식 선택 기능은 검색 결과들을 어떠한 기준으로 정렬하여 사용자에게 제공할 것인가를 지정한다. 일반적으로 정보 검색 시스템은 질의와 문서에 출현하는 검색어와 색인어들에 가중치를 부여하고, 이를 이용하여 문서와 질의 사이의 유사도를 계산한다. 그리고 유사도를 기준으로 내림차순으로 검색 결과를 정렬하여 사용자에게 제공한다. 사용자는 유사도가 높은 문서들을 우선적으로 검토함으로써 원하는 문서를 발견하는데 소모되는 시간을 단축할 수 있다. 한편, 유사도에 의한 검색 결과 정렬 이외에 문서에 포함된 특정 필드의 값에 의한 검색 결과 정렬이 요구될 수 있다. 예를 들어, 신문 기사에 대한 검색 결과는 최신 기사를 우선적으로 출력하는 것이 사용자의 만족도를 높일 수 있다.

4.3.2 문서 요약

일반적인 정보 검색 서비스들은 질의를 만족하는 문서들을 검색한 후, 검색된 문서의 제목과 더불어 문서의 요약을 사용자에게 제공한다. 사용자들은 검색된 문서의 전문을 확인하지 않고 출력된 문서의 요약만으로 문서의 내용을 추측할 수 있으며, 이러한 추측은 사용자가 문서의 전문을 확인할 것인가의 여부를 선택하는 기준이 된다. 문서의 요약을 생성하는 대표적인 방법들은 다음과 같다. 첫째, 가장 단순한 방법으로 문서의 시작 부분을 요약으로 추출하는 방식이다. 둘째, 문단 추출 방법으로 문서를 문단으로 구분한 후, 질의와 유사도가 가장 높은 문단을 요약으로 추출한다. 셋째, 문장 추출 방법으로서 질의와 유사도가 높은 문장들을 추출한 후, 이들을 조합하여 요약을 생성한다.

4.3.3 하이라이팅

하이라이팅은 검색된 문서의 제목과 요약 또는 전문을 출력할 때, 질의에 포함된 검색어와 일치되는 부분을 진하게 또는 색을 사용하여 강조한다. 진하게 표시된 정도나 다양한 색을 사용함으로써 문서의 검색에 각 검색어가 영향을 미친 정도를 표시할 수도 있다. 하이라이팅은 검색된 문서에 대한 사용자의 적합성 판정에 도움을 주며, 또한 문서 전문의 출력시 검색어의 하이라이팅은 문서의 전체 내용들 중에서 질의에 보다 적합한 부분의 발견에 매우 큰 도움을 준다.

위에서 언급한 바와 같이 사용자가 직접 작성한 질의에 대한 검색 결과에 하이라이팅의 적용은 사용자가 검색 결과의 검토에 소비하는 노력과 시간을 단축시킨다. 그러나, 시소러스를 사용하여 질의를 자동으로 확장하거나 적합성 피드백의 적용에 의해 질의가

변경된 후, 이러한 질의에 의해 검색된 문서들에 하이라이팅의 적용은 그 유용성이 감소된다. 이는 질의에 사용자가 입력하지 않은 검색어들이 포함되어 있으며, 이러한 검색어들과 일치되는 부분도 하이라이팅되기 때문이다.

영어 문서에 대한 하이라이팅은 매우 쉽게 구현될 수 있다. 그러나, 단어와 단어를 구분하는 구분자를 사용하지 않는 한글, 일본어, 중국어, 태국어 등과 같은 경우, 정확한 하이라이팅을 위해서는 단어들을 구분하는 작업이 하이라이팅에 선행되어야 한다. 예를 들어, 검색어가 “핑클”이고 검색된 문서에 “…서핑클럽에 가입한 핑클…”이라는 문자열이 존재할 경우, 일반적으로 사용되는 스트링 매칭에 의한 하이라이팅은 “서핑클럽”의 “핑클”도 하이라이팅한다. 그러나 단어의 구분이 선행될 경우, “서핑”과 “클럽”이 분리되면서 단어로서의 “핑클”만이 하이라이팅된다.

4.3.4 문서 분류

검색 대상이 되는 문서의 양이 방대할 경우 정보 검색 시스템도 입력된 질의에 대해 매우 많은 수의 문서들을 검색하며, 유사도에 따라 정렬된 문서들을 검색 결과로서 사용자에게 제공한다. 그러나, 사용자들은 적합 문서의 발견을 위해 상위 순위 20-30개 정도의 문서만을 검토하는 경향이 있다. 따라서 적합 문서들이 유사도가 낮아서 상위 순위를 차지하지 못 할 경우, 사용자들은 적합 문서들이 검색되지 않은 것으로 판단하게 된다.

위에서 언급된 문제점은 검색 결과 중에서 상위 순위를 차지하는 문서들을 실시간으로 자동 분류하여 사용자에게 제공함으로써 완화될 수 있다. 예를 들며, “보험”이라는 질의에 대하여 정보 검색 시스템은 “자동차 보험”, “비교 견적”, “의료 보험”, “화재 보험” 등과 같은 카테고리를 제시한다. 사용자는 적합 문서의 발견을 위해 정보 요구에 부합하는 카테고리를 선정한 후, 그 카테고리 내의 문서들만을 집중적으로 검토한다. 이러한 검색된 문서의 자동 분류는 문서 분류를 실행하지 않을 경우와 비교하여 사용자가 검토하는 문서들의 수를 증가시킨다.

4.4 질의 수정

4.4.1 검색어 추천

검색 결과에 만족하지 못 할 경우, 사용자는 질의로부터 검색어를 삭제하거나 질의에 새로운 검색어를 추가하는 작업을 수행한다. 검색어 추천은 이러한 질의 수정을 지원하기

위해 사용자가 입력한 질의와 관련된 단어들을 사용자에게 추천하는 기능이다. 이러한 검색어 추천을 위해서 전문가들에 의해 수작업으로 구축된 시소러스가 사용될 수 있다. 시소러스는 단어들 사이의 다양한 관계 즉, 광의어, 협의어, 연관어, 동의어 등을 정의하고 있다. 그러나, 광범위한 분야들에 대한 시소러스를 수작업으로 구축하는 일은 매우 어려운 일이다. 이러한 문제점을 보완하기 위해 문서들을 분석하여 단어들 사이의 연관성을 계산한 후, 질의에 포함된 검색어들과 연관성이 높은 단어들을 추천하는 방법들이 개발되어 왔다.

4.4.2 적합성 피드백

정보 검색 시스템에 입력되는 질의는 매우 임의적이며, 때때로 사용자들은 그들이 지니고 있는 정보 요구조차도 정확하게 표현할 줄 모른다. 이러한 사용자들에게 수작업으로 질의를 개선하는 작업은 매우 어려운 일이다. 이러한 문제점을 해결하기 위한 방법으로 적합성 피드백이 개발되어 왔다. 적합성 피드백은 사용자들에 의해 생성된 적합성 정보를 활용하여, 불완전한 질의를 보완할 수 있는 질의를 자동으로 생성한다. 즉, 사용자는 정보 요구에 근거하여 검색된 문서들을 적합 또는 비적합 문서로 구분하고, 정보 검색 시스템은 이러한 적합성 정보를 활용하여 새로운 질의를 생성한다.

4.4.3 유사 문서 검색

유사 문서 검색은 웹 검색 서비스들이 공통적으로 지원하는 기능들 중의 하나로서, 사용자가 검색된 문서들 중에서 하나를 지정하면 이 문서와 유사한 내용을 지닌 문서들을 검색한다. 유사 문서 검색은 적합성 피드백의 기능을 축소한 것으로 간주될 수 있다. 즉, 적합성 피드백이 다수의 적합 문서와 다수의 부적합 문서를 기반으로 새로운 질의를 생성하는데 비하여, 유사 문서 검색은 한 개의 적합 문서만을 기반으로 새로운 질의를 생성한다.

제 5 장 검색 효과 평가

5.1 테스트 컬렉션

5.1.1 역사 및 종류

정형화된 데이터를 다루는 DBMS 의 성능 평가에는 질의 처리 속도, 저장 공간 사용 효율과 같은 기준들이 사용되는데 비하여, 정보 검색 시스템의 성능 평가에는 추가적으로 검색 효과(retrieval effectiveness)를 사용한다. 즉, 서로 다른 정보 검색 시스템은 서로 다른 검색 효과를 제공하며, 사용자는 보다 높은 검색 효과를 제공하는 정보 검색 시스템을 사용함으로써 정보 발견에 소요되는 시간과 노력을 단축시킬 수 있다. 이러한 검색 효과는 미국 및 유럽을 중심으로 발전해 온 지난 40 여년 동안 정보 검색 연구에서 가장 중요한 평가 기준으로 사용되어 왔다.

정보 검색 시스템의 검색 효과를 향상시키기 위하여 색인어 가중치, 자연어 처리, 적합성 피드백 등을 이용한 다양한 검색 기법들이 개발되고 있다. 정보 검색 분야의 연구에 있어서 특징적인 사항 중의 하나는 직관적 통찰에 의해 개발된 검색 기법들이 많은 경우에 검색 효과의 향상을 가져오지 않는다는 것이다. 이에 대한 예로서 시소러스(Thesaurus)를 사용함으로써 기대했던 만큼의 검색 효과의 향상을 얻는데 실패하여 왔음을 들 수 있다. 따라서 개발중인 검색 기법의 성능을 평가할 수 있는 테스트 컬렉션은 검색 기법의 개발에 있어서 필수적인 요소이다.

정보 검색에 있어서 테스트 컬렉션을 이용한 실험은 오랜 역사를 지니고 있다. 정보 검색에 대한 연구는 Cranfield I 이라고 불리는 색인에 대한 실험과 함께 시작되었으며, 그 후로 30 년이 넘는 동안 실험은 검색 기법의 개발에 있어서 필수적인 요소로 인식되어 왔다. Cranfield II 에서는 컴퓨터에 의한 자동 색인과 사람에 의한 수작업 색인을 비교하였으며, 자동 색인에 대한 긍정적인 연구 결과는 정보 검색에 대한 긍정적인 연구의 계기가 되었다. 1960 년대 말에 생성된 Cranfield 컬렉션은 1400 개의 문서와 225 개의 질문으로 구성되어 있으며, 그 후로 많은 사람들에 의해 활발히 이용되어 왔다. 그 외의 테스트 컬렉션으로는 CACM 컬렉션, NPL 컬렉션 등이 있다.

위에서 언급된 테스트 컬렉션들은 매우 적은 수의 문서들을 대상으로 구축된 소규모 테스트 컬렉션들이다. 따라서 산업체에서는 이러한 소규모 테스트 컬렉션을 사용한 정보 검색 연구 결과들을 신뢰하지 않는 경향을 보여 왔다. 그러나, 미국의 NIST(National Institute of Standards and Technology)의 후원으로 1992 년에 처음으로 개최된 학술 대회 TREC(Text REtrieval Conference)에서 1 백만 건을 초과하는 문서들을 대상으로 대용량 테스트 컬렉션의 구축을 시작하였으며, 이후 매년 테스트 컬렉션에 포함되는 문서들의 수를 증가시키고 있다.

학술 대회 TREC 에서는 이러한 테스트 컬렉션을 사용하여 실험실에서 개발된 시스템들 뿐만 아니라 산업체에서 개발된 상용 시스템들도 평가하여 그 결과를 발표하고 있다.

일본에서도 테스트 컬렉션의 중요성을 인식하여 정보 기관인 NACSIS(National Center for Information Systems)가 주관이 되어 대규모 테스트 컬렉션 구축 사업을 추진 중이다. 또한 NTT Data Corporation 에서 BMIR-J1 과 BMIR-J2 라는 테스트 컬렉션을 개발하였으며, BMIR-J1 은 600 건의 문서와 60 개의 질의로 구성되어 있고, BMIR-J2 는 경제학 및 공학 분야에서 5,080 건의 신문기사와 60 개의 질의를 포함하고 있다.

앞에서 설명된 바와 같이 외국에서는 다양한 테스트 컬렉션들이 개발되어 정보 검색에 대한 연구에 많이 이용되어 왔다. 그러나 이러한 테스트 컬렉션들에 포함된 문서들은 모두 한글과는 특성이 매우 다른 영어로 작성되어 있다. 예를 들면, 영어는 단어의 구분이 분명하여 색인 과정이 단순한데 비하여, 한글은 띄어 쓰기의 자유로움과 조사의 발달로 인하여 색인 과정에 어려움을 지니고 있다. 따라서 한글 문서들로 구성된 테스트 컬렉션은 한글 정보 검색 연구를 위한 필수적인 요소이다.

국내의 경우에도 한글 문서들로 구성된 테스트 컬렉션의 필요성은 인식하고 다음과 같은 다양한 테스트 컬렉션들이 개발되어 왔다. KT 컬렉션은 정보과학회논문지, 한국정보과학회 1993 Proceedings, 정보관리학회지에 수록된 1,053 건의 논문들과 30 개의 매우 단순한 질문을 포함하고 있다. KRIST 컬렉션은 13,315 건의 과기처 연구보고서에 대한 서지 레코드와 30 개의 자연어 질의로 구성되어 있다. HANTEC 은 가장 최근에 개발된 대용량 테스트 컬렉션으로서 일반, 사회 과학, 과학 기술 분야에 속하는 120,000 건의 문서들과 30 개의 질의로 구성되어 있다.

5.1.2 구축 방법

정보 검색용 테스트 컬렉션은 일반적으로 문서 집합, 질의 집합 그리고 각 질의에 대한 적합 문서 리스트로 구성된다. 이들 중 검색의 대상이 되는 문서 집합은 테스트 컬렉션 구축에 있어서 가장 기본적인 요소이다. 문서 집합의 구성에 있어서 고려해야 할 사항은 다양한 분야 및 크기의 문서들로 문서 집합을 구성해야 한다는 것이다. 즉, 정보 검색 기술은 많은 경우에 통계적인 방법이나 언어 처리 기술을 사용하는데, 이들은 모두 문서의 종류에 많은 영향을 받는다. 또한, 문서와 질의의 유사도 계산에 핵심적 역할을 하는 가중치 기법들 중 일부는 특정 크기의 문서들에 높은 유사도를 부여하는 특성을 지니고 있기 때문에 가중치 기법의 성능 평가를 위해서라도 다양한 크기의 문서들로 문서 집합을 구성하는 것이 바람직하다.

예를 들어, HANTEC 테스트 컬렉션의 문서 집합은 다음과 같이 구성되었다. 일반, 사회 과학, 과학 기술의 각 분야별로 40,000건의 문서들을 선정함으로써 문서의 내용이 특정 분야에 편중되지 않도록 하였다. 즉, 일반 분야는 1994년에 발행된 한국일보 기사 22,000건, gov 확장자를 갖는 웹 페이지 9,000건과 com 확장자를 갖는 웹 페이지 9,000건으로 구성되고, 사회 과학 분야는 1994년에 발행된 한국 경제 신문 기사 39,480건, 한국여성개발원이 발행한 정기간행물 여성연구에 게제된 논문 110건과 경북 도의회 회의록 410건으로 구성되며, 과학 기술 분야는 과기처 연구보고서 10,000건, 연구개발정보센터가 발간한 해외과학기술동향 18,000건과 학술 논문 서지 사항 12,000건으로 구성된다. 한편, 이들 문서들은 짧게는 수십 바이트에서 길게는 수십만 바이트까지 매우 다양한 길이를 지니고 있다.

문서 집합이 구성된 다음에는 질의들의 작성이 요구된다. 정보 검색용 테스트 컬렉션에 포함된 질의들은 일반적으로 사용자들이 정보 검색 시스템에 입력하는 질의와 다른 형태를 지니고 있다. 예를 들어, 다음은 HANTEC 컬렉션에 포함된 하나의 질의를 보여준다.

<num> 01
<title> 월드컵 축구 유치
<desc> 한국의 2002년 월드컵 축구 유치 활동 내용
<narr> 한국의 2002년 월드컵 축구 유치를 위한 국내외적인 활동이나 한국개최에 대한 회원국의 반응을 포함한 정보는?
<query> 2002년 월드컵 축구 피파 FIFA 회원국 한국 개최 주최 유치전략 홍보 활동

HANTEC 컬렉션에 포함된 질의들은 <num>, <title>, <desc>, <narr>, <query>의 5개 항목으로 구성되며, 각각은 질의 번호, 질의 제목, 질의 설명, 질의 해설, 질의에 포함된 단어 리스트를 의미한다. 이들 중에서 <title>과 <desc>는 정보 검색 시스템에 입력되어 질의 생성에 이용되는 부분이고, <narr>은 적합 문서의 판별 기준을 기술한 항목이다. <narr> 항목은 적합 문서의 판별 기준을 적합성 판단을 수행하는 사람에게 제공하는 것을 주 목적으로 하고 있으나, 질의 생성에도 이용될 수 있다. <query>는 질의 생성을 도와주기 위한 항목으로 관련된 어절의 집합으로 구성되어 있으며, 앞의 4개 항목에 포함되지 않은 어절이라도 검색을 보충할 단어라면 <query>에 포함된다.

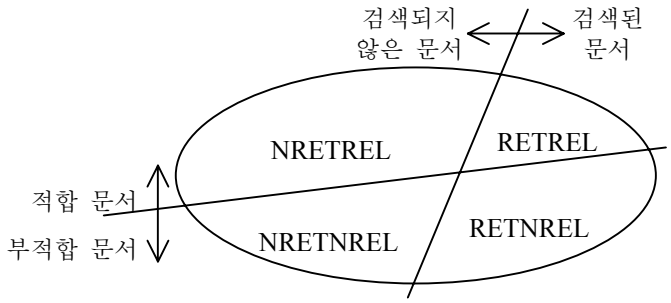
각각의 질의에 대한 적합 문서 리스트의 생성은 테스트 컬렉션 구축에 있어서 가장 중요한 요소이다. 적합 문서 리스트의 생성을 위한 가장 초보적인 방법은 각각의 질의에 대하여 테스트 컬렉션에 포함된 모든 문서들을 읽고 적합성 여부를 판단하는 것이다. 그러나, 이 방법은 문서의 수가 많은 경우에 대단히 많은 시간을 요구하므로 현실적으로 거의 불가능하다.

보다 현실적인 방법으로 서로 다른 특성을 지닌 다수의 검색 시스템을 사용하여 검색을 수행하고, 각각의 시스템에 의해 높은 순위를 부여 받은 문서들에 대해서만 적합성 여부를 판단하는 방법이 제안되었다. 이 방법은 특성이 다른 다수의 시스템들에 의해 검색된 후보 적합 문서들 내에 거의 모든 적합 문서들이 포함되어 있음을 가정하고 있다. 사용자의 적합성 판단 작업이 전체 컬렉션이 아닌 후보 적합 문서들에 국한되므로, 많은 수의 문서들이 컬렉션에 포함된 경우 적합 문서 리스트를 생성할 수 있는 현실적인 방법으로 알려져 있다. 이 방법은 풀링 방법(pooling method)이라고 불리며, 테스트 컬렉션 구축 시 적합 문서 리스트 생성에 효과적인 방법으로 알려져 있다.

5.2 평가 척도

5.2.1 재현율과 정확률

재현율과 정확률은 검색 효과의 평가를 위해 일반적으로 사용되는 척도이다 (Salton et al. 1983). 재현율은 전체 적합 문서에 대한 검색된 적합 문서의 비율을 나타내고, 정확률은 전체 검색 문서에 대한 검색된 적합 문서의 비율이다. 즉, 재현율은 문서 집합에서 사용자가 원하는 문서를 어느 정도 검색하였는가를 나타내고, 정확률은 검색된 문서들 중에서 사용자가 원하는 문서가 얼마나 포함되어 있는가를 나타낸다. 다음 그림은 전체 문서 집합을 검색 여부와 적합성 여부에 따라 구분한 것이다.



이때, 검색 효과 평가 척도 재현율과 적합률은 다음과 같은 수식으로 표현될 수 있다.

$$\text{재현율} = \frac{\text{검색된 적합 문서 수}}{\text{적합 문서 수}} = \frac{\text{RETREL}}{\text{RETREL} + \text{NRETREL}}$$

$$\text{정확률} = \frac{\text{검색된 적합 문서 수}}{\text{검색된 문서 수}} = \frac{\text{RETREL}}{\text{RETREL} + \text{RETNREL}}$$

예를 들면, 전체 문서 집합에서 200 개의 문서가 저장되어 있고, 이 문서 집합 속에 사용자의 질의에 적합한 문서가 5 개 포함되어 있다고 가정하자. 이때 사용자가 검색 시스템을 사

용하여 6 개의 문서를 검색하였고, 검색된 문서 중에서 4 개의 문서가 질의를 만족하는 문서라고 하면, 재현율과 정확률은 각각 0.8 과 0.67 이 된다.

두개의 검색 결과들 A 와 B 의 검색 효과를 재현율과 정확률을 사용하여 비교할 때, 검색 결과 A 의 재현율과 정확률이 모두 검색 결과 B 보다 높으면, 검색 결과 A 의 검색 효과가 보다 우수한 것으로 판단할 수 있다. 그러나, 재현율은 검색 결과 A 가 높고, 정확률은 검색 결과 B 가 높거나, 또는 그 반대인 경우, 사용자에게 재현율과 정확률 중에서 어떠한 평가 척도가 보다 중요한가를 검토해야 한다. 즉, 사용자가 모든 적합 문서들이 검색될 것을 요구할 경우 높은 재현율을 제공하는 검색 결과가 우수하며, 예를 들어 특허 또는 법률 분야의 검색에는 일반적으로 정확률 보다 재현율이 중요시 된다. 한편, 방대한 양의 웹 문서들을 대상으로 하는 웹 검색 시스템의 사용자들은 유사도가 높은 상위 10-20 개 정도의 문서만을 검토하는 경향을 지니고 있기 때문에, 웹 검색 시스템은 재현율을 낮을 지라도 높은 정확률을 제공하도록 설계되는 것이 바람직하다.

5.2.2 평균 재현율과 평균 정확률

사람마다 서로 다른 능력을 지니고 있듯이, 일반적으로 서로 다른 검색 시스템은 서로 다른 질의에 대하여 상대적으로 높은 검색 효과를 제공한다. 즉, 검색 시스템 A 가 질의 q_1 보다 질의 q_2 에 대하여 높은 검색 효과를 제공할 지라도, 이와는 반대로 검색 시스템 B 는 질의 q_1 에 대하여 보다 높은 검색 효과를 제공할 수 있다. 따라서 하나의 질의만을 사용한 검색 시스템의 평가는 부정확하기 때문에, 다수의 질의에 대한 검색 결과들로부터 산출된 평균 재현율과 평균 정확률을 검색 효과의 평가 척도로 이용하는 것이 바람직하다. 평균 재현율과 평균 정확률을 산출하는 방법은 다음에서 설명되는 사용자 지향적 방법과 시스템 지향적 방법으로 구분될 수 있다.

사용자 지향적 방법은 각 질의 i 에 대한 검색 결과로부터 재현율과 정확률을 계산하고 그 평균을 산출하는 것으로 다음과 같은 수식으로 표현될 수 있다.

$$\text{평균재현율}_{\text{사용자지향적}} = \frac{1}{n}\sum_{i=1}^{n} \text{재현율}_i$$

$$\text{평균정확률}_{\text{사용자지향적}} = \frac{1}{n}\sum_{n=1}^{n} \text{정확률}_i$$

여기에서 n 은 전체 질의 수를 나타낸다. 사용자 지향적 방법은 각 질의에 동일한 중요도를 부여하여 평균 재현율과 평균 정확률을 계산하며, 이 방법에 의해 계산된 평균 재현율과 평균 정확률은 사용자가 검색 시스템으로부터 기대할 수 있는 성능을 반영한다.

시스템 지향적 방법에서 평균 재현율은 n 개의 질의에 대해 검색된 전체 적합 문서 수를 n 개의 질의에 대한 전체 적합 문서 수로 나누기를 수행함으로써 계산되고, 평균 정확률은 n 개의 질의에 대해 검색된 전체 적합 문서 수를 n 개의 질의에 대한 검색된 전체 문서 수로 나누기를 수행함으로써 계산되며, 다음과 같은 수식으로 표현될 수 있다.

$$\text{평균재현율}_{\text{시스템지향적}} = \sum_{i=1}^{n} \text{검색된적합문서수}_i \Big/ \sum_{i=1}^{n} \text{적합문서수}_i$$

$$\text{평균정확률}_{\text{시스템지향적}} = \sum_{i=1}^{n} \text{검색된적합문서수}_i \Big/ \sum_{i=1}^{n} \text{검색된문서수}_i$$

시스템 지향적 방법에서 평균 재현율은 적합 문서 한 개당 검색된 적합 문서 수를 반영하며 평균 정확률은 검색된 문서 한 개당 검색된 적합 문서 수를 반영한다.

한편, 시스템 지향적 방법에서 평균 재현율과 평균 정확률은 적은 수의 적합 문서를 갖는 질의보다 많은 수의 적합 문서를 갖는 질의에 보다 많은 영향을 받는다. 예를 들어, 10 개의 적합 문서를 갖는 질의 q_1에 대하여 6 개의 적합 문서와 4 개의 부적합 문서를 검색하고, 2 개의 적합 문서를 갖는 질의 q_2에 대하여 1 개의 적합 문서와 1 개의 부적합 문서를 검색하였다고 가정하자. 질의 q_1의 재현율과 정확률은 모두 0.6 이고, 질의 q_2의 재현율과 정확률은 모두 0.5 이다. 이때 사용자 지향적 방법을 적용한 평균 재현율과 평균 정확률은 모두 0.55 이다. 그러나 시스템 지향적 방법을 적용한 평균 재현율과 평균 정확률은 모두 0.58 로서 보다 많은 수의 적합 문서를 갖는 질의 q_1의 재현율과 정확률에 근접한다.

5.2.3 재현율 수준 정확률

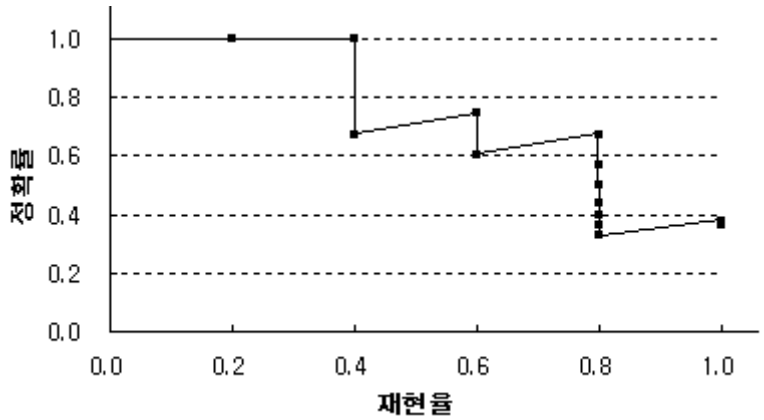
질의와 문서들 사이의 유사도 계산을 수행하지 않는 순수한 불리안 검색 시스템은 검색 결과로서 문서들의 집합을 제공한다. 이 경우 앞에서 설명한 재현율과 정확률을 계산함으로써 검색 효과를 평가할 수 있다. 그러나, 현대의 많은 정보 검색 시스템들은 질의와 문서들 사이의 유사도를 계산하고, 이 값에 따라서 문서들을 정렬하여 유사도가 큰 문서들을 우선적으로 사용자에게 제공한다. 즉, 유사도가 가장 큰 문서의 순위는 1, 다음 문서의 순위는 2 와 같은 식으로 문서들이 출력된다. 따라서, 검색된 문서의 수가 정해 지지 않기 때문에, 재현율과 정확률의 계산에 문제점이 발생한다.

유사도 계산으로 인하여 검색된 문서의 수가 정해지지 않는 시스템에서 재현율과 정확률은 <표 5.1>과 같이 높은 순위의 문서부터 하나씩 적합성 여부를 판정함으로써 계산된다. 그러나 이 경우 하나의 재현율에 다수의 정확률이 대응되거나 특정 재현율에 대응되는 정확률이 존재하지 않는 수가 있다. 예를 들어, <표 5.1>에서 재현율 0.4 에는 2 개의 정확률이 대

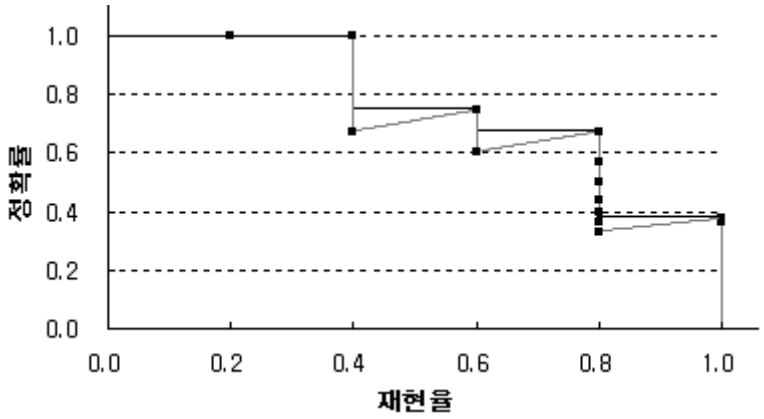
응되며, 재현율 0.8 에는 7 개의 정확률이 대응된다. 또한 재현율 0.3 이나 0.5 에 대응되는 정확률은 존재하지 않는다. <그림 5.1>의 재현율-정확률 그래프는 이러한 문제점을 시각적으로 보여준다.

<표 5.1> 문서 순위 14 위까지의 검색 결과 (전체 적합 문서 수: 5 개)

순위	문서 번호	적합성	재현율	정확률
1	588	○	0.2	1.0
2	589	○	0.4	1.0
3	576	×	0.4	0.67
4	590	○	0.6	0.75
5	986	×	0.6	0.60
6	592	○	0.8	0.67
7	984	×	0.8	0.57
8	988	×	0.8	0.50
9	578	×	0.8	0.44
10	985	×	0.8	0.40
11	103	×	0.8	0.36
12	591	×	0.8	0.33
13	772	○	1.0	0.38
14	990	×	1.0	0.36



<그림 5.1> 표 5.1 의 검색 결과에 대한 재현율-정확률 그래프



<그림 5.2> 표 5.1 의 검색 결과에 보간법을 적용한 재현율-정확률 그래프

이러한 문제점을 해결하기 위해 보간법(interpolation)이 사용되며, 이 기법은 임의의 재현율에 항상 하나의 정확률을 대응시킨다. 지금까지 다양한 보간법들이 개발되었으나, 이들 중에서 가장 보편적으로 사용되는 방법은 <그림 5.2>에서와 같이 여러 개의 수평선으로 구성되는 계단식 그래프를 생성하는 것이다. 이러한 그래프는 가장 높은 재현율 값에서 출발하여, 이 재현율 값에 대응하는 가장 큰 정확률 값에서부터 보다 큰 정확률 값을 제공하는 재현율 값까지 왼쪽으로 수평선을 그음으로써 생성된다. <그림 5.2>의 그래프는 각 재현율 값에 대하여 단지 하나의 정확률 값만을 대응시키며, 또한 모든 재현율 값에 대하여 정확률 값을 대응시킨다.

검색 시스템의 검색 효과를 평가하기 위해서는 테스트 컬렉션에 포함된 모든 질의에 대하여 보간법을 적용한 재현율-정확률 그래프를 생성한 후, 이들을 이용하여 평균적인 재현율-정확률 그래프를 생성하는 일이 필요하다. 그러나 서로 다른 질의로부터 생성된 재현율-정확률 그래프들을 살펴보면, 서로 다른 재현율 값들에 대하여 정확률을 계산하였음을 발견할 수 있다. 예를 들어, <표 5.2>에서 계산된 재현율을 살펴보면 매우 많은 수의 재현율 값이 출현함을 알 수 있으며, 이 값들을 <표 5.1>에서 계산된 재현율 값들과 비교하면 매우 상이함을 알 수 있다.

위에서 언급된 문제점으로 인하여 평균적인 재현율-정확률 그래프를 생성하기 위해서는 재현율 값들의 정규화가 요구된다. 일반적으로 다양한 재현율 값들은 소수점 2 번째 자리에서 내림을 하여 0.0, 0.1, 0.2, …, 1.0 의 11 개 재현율 값으로 정규화 된다. 즉, <표 5.2>에서 순위 1 부터 9 까지의 재현율은 0.0, 순위 10 부터 15 위까지의 재현율은 0.1, 순위 16 위부터 34 위까지의 재현율은 0.2, 순위 35 위부터 54 위까지의 재현율은 0.3, 순위 55 위부터 90 위까지의 재현율은 0.4, 순위 91 위부터 100 위까지의 재현율은 0.5 로 정규화 된다.

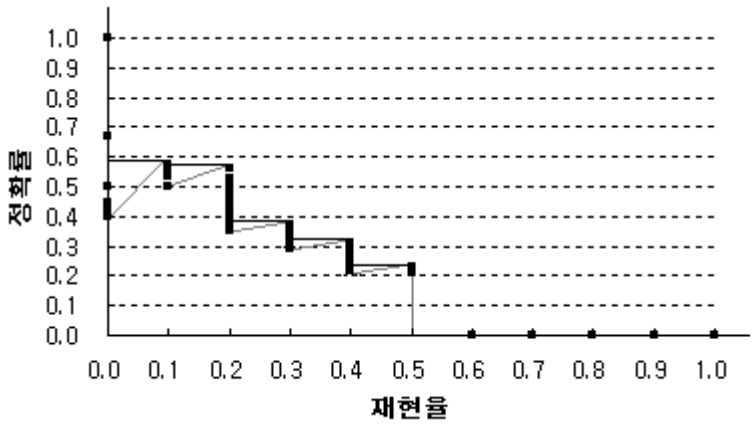
<표 5.2> 문서 순위 100 위까지의 검색 결과 (전체 적합 문서 수: 41 개)

순위	문서 번호	적합성	재현율	정확률	순위	문서 번호	적합성	재현율	정확률
1	WSJ920228-0191	○	0.0244	1.0000	51	WSJ910627-0092	×	0.3659	0.2941
2	WSJ910621-0176	×	0.0244	0.5000	52	WSJ901220-0150	○	0.3902	0.3076
3	WSJ911007-0036	○	0.0488	0.6666	53	WSJ920227-0058	×	0.3902	0.3018
4	WSJ920319-0151	×	0.0488	0.5000	54	WSJ911111-0054	×	0.3902	0.2962
5	WSJ910730-0088	×	0.0488	0.4000	55	WSJ900524-0054	○	0.4146	0.3090
6	WSJ911115-0065	○	0.0732	0.5000	56	WSJ910226-0011	×	0.4146	0.3036
7	WSJ920213-0099	×	0.0732	0.4285	57	WSJ900420-0017	×	0.4146	0.2982
8	WSJ910621-0145	○	0.0976	0.5000	58	WSJ900601-0045	×	0.4146	0.2931
9	WSJ910614-0036	×	0.0976	0.4444	59	WSJ910617-0056	×	0.4146	0.2881
10	WSJ911205-0060	○	0.1220	0.5000	60	WSJ920121-0054	×	0.4146	0.2833
11	WSJ910624-0071	○	0.1463	0.5454	61	WSJ910226-0109	×	0.4146	0.2787
12	WSJ911029-0023	×	0.1463	0.5000	62	WSJ911122-0153	×	0.4146	0.2742
13	WSJ920116-0130	○	0.1707	0.5384	63	WSJ900821-0111	×	0.4146	0.2698
14	WSJ911120-0064	○	0.1951	0.5714	64	WSJ910903-0145	×	0.4146	0.2656
15	WSJ911219-0025	×	0.1951	0.5333	65	WSJ911120-0033	×	0.4146	0.2615
16	WSJ910806-0091	○	0.2195	0.5625	66	WSJ910724-0063	×	0.4146	0.2576
17	WSJ900904-0058	×	0.2195	0.5294	67	WSJ911112-0055	×	0.4146	0.2537
18	WSJ911119-0047	×	0.2195	0.5000	68	WSJ920316-0081	×	0.4146	0.2500
19	WSJ910607-0133	○	0.2439	0.5263	69	WSJ900817-0048	×	0.4146	0.2464
20	WSJ910805-0082	×	0.2439	0.5000	70	WSJ911021-0102	×	0.4146	0.2429
21	WSJ920306-0058	○	0.2683	0.5238	71	WSJ900716-0137	×	0.4146	0.2394
22	WSJ920218-0155	×	0.2683	0.5000	72	WSJ910820-0029	×	0.4146	0.2361
23	WSJ910822-0045	×	0.2683	0.4782	73	WSJ911220-0103	×	0.4146	0.2329
24	WSJ901224-0099	×	0.2683	0.4583	74	WSJ911224-0012	×	0.4146	0.2297
25	WSJ911127-0153	×	0.2683	0.4400	75	WSJ910725-0088	×	0.4146	0.2267
26	WSJ910219-0150	×	0.2683	0.4230	76	WSJ920302-0119	○	0.4390	0.2368
27	WSJ910618-0082	×	0.2683	0.4074	77	WSJ920130-0095	×	0.4390	0.2338
28	WSJ910708-0061	○	0.2927	0.4285	78	WSJ911204-0093	×	0.4390	0.2308
29	WSJ900706-0047	×	0.2927	0.4137	79	WSJ911018-0152	×	0.4390	0.2278
30	WSJ911118-0001	×	0.2927	0.4000	80	WSJ900920-0078	○	0.4634	0.2375
31	WSJ920317-0149	×	0.2927	0.3870	81	WSJ910410-0040	×	0.4634	0.2346
32	WSJ910213-0015	×	0.2927	0.3750	82	WSJ920303-0073	×	0.4634	0.2317
33	WSJ910917-0005	×	0.2927	0.3636	83	WSJ900427-0007	×	0.4634	0.2289
34	WSJ910422-0165	×	0.2927	0.3529	84	WSJ910906-0021	×	0.4634	0.2262
35	WSJ920227-0147	○	0.3171	0.3714	85	WSJ900711-0022	×	0.4634	0.2235
36	WSJ911028-0059	×	0.3171	0.3611	86	WSJ910329-0038	×	0.4634	0.2209
37	WSJ910423-0184	×	0.3171	0.3513	87	WSJ900806-0026	×	0.4634	0.2184
38	WSJ900709-0085	×	0.3171	0.3421	88	WSJ911021-0038	×	0.4634	0.2159
39	WSJ911105-0088	○	0.3415	0.3589	89	WSJ910212-0092	×	0.4634	0.2135
40	WSJ920114-0073	○	0.3659	0.3750	90	WSJ911204-0020	○	0.4878	0.2222
41	WSJ910311-0051	×	0.3659	0.3658	91	WSJ900907-0190	○	0.5122	0.2308
42	WSJ900913-0069	×	0.3659	0.3571	92	WSJ900423-0002	×	0.5122	0.2283
43	WSJ910730-0064	×	0.3659	0.3488	93	WSJ910815-0100	×	0.5122	0.2258
44	WSJ910606-0015	×	0.3659	0.3409	94	WSJ900611-0152	×	0.5122	0.2234
45	WSJ900925-0025	×	0.3659	0.3333	95	WSJ910128-0100	×	0.5122	0.2211
46	WSJ910111-0022	×	0.3659	0.3260	96	WSJ911121-0046	×	0.5122	0.2188
47	WSJ900402-0191	×	0.3659	0.3191	97	WSJ900529-0064	×	0.5122	0.2165
48	WSJ920225-0149	×	0.3659	0.3125	98	WSJ911204-0010	×	0.5122	0.2143
49	WSJ911113-0147	×	0.3659	0.3061	99	WSJ910213-0010	×	0.5122	0.2121
50	WSJ911114-0059	×	0.3659	0.3000	100	WSJ911106-0053	×	0.5122	0.2100

<표 5.3> 표 5.2 의 검색 결과에

재현율 정규화와 보간법을 적용한 재현율-정확률

재현율	정확률
0.0	1.0000
0.1	0.5714
0.2	0.5625
0.3	0.3750
0.4	0.3090
0.5	0.2308
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000



<그림 5.3> 표 5.2 의 검색 결과에
재현율 정규화와 보간법을 적용한 재현율-정확률 그래프

한편, <표 5.2>의 검색 결과에 대하여 재현율-정확률 그래프를 생성하기 위해서는 0.6 부터 1.0 까지의 재현율에 대한 정확률의 계산이 요구된다. 학술 대회 TREC 에서는 상위 100 위 이내에서 검색되지 않은 적합 문서들의 순위를 무한대로 가정하기 때문에, 0.6 부터 1.0 까지의 재현율에 대한 정확률을 모두 0 으로 계산한다. 따라서 <표 5.2>의 검색 결과에 재현율 정규화와 보간법을 적용할 경우, 0.0 부터 1.0 까지의 11 개 재현율 값들에 대한 정확률 값들은 <표 5.3>과 같이 계산될 수 있으며, <그림 5.3>과 같은 재현율-정확률 그래프가 생성된다.

5.2.4 문서 수준 정확률

정확률은 전체 검색 문서에 대한 검색된 적합 문서의 비율로서 정의되며, 따라서 정확률을 계산하기 위해서는 검색된 문서의 수가 결정되어야 한다. 유사도 계산으로 인하여 검색된 문서의 수가 결정되지 않는 시스템에서는 검색된 문서의 수를 임의로 지정함으로써 정확률을 계산할 수 있다. 즉, 문서 수준 *n* 에서의 정확률은 상위 *n* 개의 문서들에 포함된 적합 문서들의 비율로서 정의되며, 다음과 같은 수식으로 표현될 수 있다.

문서 수준 정확률 = 상위 *n* 개의 문서들에 포함된 적합 문서 수 / 문서 수준 *n*

일반적으로 검색 결과의 검색 효과 평가를 위해서 여러 개의 문서 수준에 대한 정확률을 측정하며, 학술 대회 TREC 에서는 9 개의 문서 수준 5, 10, 15, 20, 30, 100, 200, 500, 1000 에 대한 문서 수준 정확률을 측정하고 있다. 예를 들어, <표 5.2>의 100 개의 검색 결과에 대하여 TREC 에서 사용되는 9 개의 문서 수준에 대한 정확률은 <표 5.4>과 같이 계산된다.

<표 5.4> 9 개의 문서 수준에 대한 정확률

문서 수준	정확률
5	0.4000
10	0.5000
15	0.5333
20	0.5000
30	0.4000
100	0.2100
200	0.1050
500	0.0420
1000	0.0210

<표 5.2>의 검색 결과에 포함된 문서의 수는 100 개이다. 따라서 문서 수준 5, 10, 15, 20, 30, 100 에서의 정확률은 위의 식을 사용하여 쉽게 계산될 수 있다. 예를 들어, 상위 30 위 이내에 포함된 적합 문서의 수가 12 개이기 때문에, 문서 수준 30 에서의 정확률은 0.4 로 계산된다. 그러나, 검색 결과가 상위 100 위까지만을 포함하고 있기 때문에, 문서 수준 200, 500, 1000 에서의 정확률을 계산하는데 있어 문제점이 발생한다. 학술 대회 TREC 에서는 이러한 경우에 대하여 문서 순위 101 부터 1000 까지 적합 문서가 검색되지 않는다고 가정하고 문서 수준 정확률을 계산한다. 따라서 상위 1000 위 이내에 포함된 적합 문서의 수가 21 개이기 때문에, 문서 수준 1000 에서의 정확률은 0.021 로 계산된다.

<표 5.5> 적합 문서가 검색된 순위에서 계산된 정확률

	문서번호	검색순위	정확률		문서번호	검색순위	정확률
1	WSJ920228-0191	1	1.0000	22	WSJ900720-0157	∞	0.0000
2	WSJ911007-0036	3	0.6666	23	WSJ900802-0150	∞	0.0000
3	WSJ911115-0065	6	0.5000	24	WSJ900820-0109	∞	0.0000
4	WSJ910621-0145	8	0.5000	25	WSJ900907-0137	∞	0.0000
5	WSJ911205-0060	10	0.5000	26	WSJ900910-0146	∞	0.0000
6	WSJ910524-0071	11	0.5454	27	WSJ900917-0025	∞	0.0000
7	WSJ920116-0130	13	0.5384	28	WSJ901221-0077	∞	0.0000
8	WSJ911120-0064	14	0.5714	29	WSJ901221-0180	∞	0.0000
9	WSJ910927-0091	16	0.5625	30	WSJ910130-0144	∞	0.0000
10	WSJ910607-0133	19	0.5263	31	WSJ910215-0049	∞	0.0000
11	WSJ920306-0058	21	0.5238	32	WSJ910325-0003	∞	0.0000
12	WSJ910708-0061	28	0.4285	33	WSJ910531-0053	∞	0.0000
13	WSJ920227-0147	35	0.3714	34	WSJ910624-0071	∞	0.0000
14	WSJ911105-0088	39	0.3589	35	WSJ910715-0053	∞	0.0000
15	WSJ920114-0073	40	0.3750	36	WSJ910715-0127	∞	0.0000
16	WSJ901220-0150	52	0.3076	37	WSJ910806-0091	∞	0.0000
17	WSJ900524-0054	55	0.3090	38	WSJ910926-0069	∞	0.0000
18	WSJ920302-0119	76	0.2237	39	WSJ920110-0094	∞	0.0000
19	WSJ900920-0078	80	0.2375	40	WSJ920116-0018	∞	0.0000
20	WSJ911204-0020	90	0.2222	41	WSJ920128-0088	∞	0.0000
21	WSJ900907-0190	91	0.2308				

5.2.5 적합 문서들에 대한 평균 정확률

각각의 문서에 순위가 부여된 검색 결과의 검색 효과를 평가하기 위하여 각각의 적합 문서가 검색된 순위에서 정확률을 계산하고 이들에 대한 평균을 구하는 방법이 사용될 수 있다. 예를 들어, <표 5.2>의 검색 결과를 생성하기 위해 사용된 질의를 만족하는 적합 문서들은 전체 문서 집합에 41 개가 존재하며, 이들 중에서 21 개가 상위 100 위 이내에서 검색되었다. <표 5.5>는 41 개의 적합 문서 각각에 대하여 검색된 순위에서 계산된 정확률을 보여주며, 적합 문서들에 대한 평균 정확률은 이들에 대한 평균 0.2317 로 계산된다.

상위 100 위 이내에서 검색된 적합 문서들에게는 검색 순위가 부여되기 때문에, 이들 문서들이 검색된 순위에서의 정확률은 쉽게 계산될 수 있다. 예를 들어, 적합 문서 ‘ WSJ911205-0060’ 의 검색 순위는 10 위이며, 상위 10 위 이내에서 검색된 적합 문서들의 수가 5 개이기 때문에, 적합 문서 ‘ WSJ911205-0060’ 이 검색된 순위에서의 정확률은 5/10=0.5 이다. 그러나, 상위 100 위 이내에서 검색되지 않은 20 개의 적합 문서들에게는 검색 순위가 부여되지 않기 때문에, 이들 문서들이 검색된 순위에서의 정확률을 계산하는데 있어

문제점이 발생한다. 학술 대회 TREC 에서는 상위 100 위 이내에서 검색되지 않은 적합 문서들의 순위를 무한대로 가정하며, 따라서 이러한 적합 문서들이 검색된 순위에서의 정확률은 모두 0 으로 계산된다.

5.2.6 *R*-정확률

R-정확률에서 *R* 은 적합 문서들의 수를 의미하며, 다음과 같이 상위 *R* 개의 문서들에 포함된 적합 문서들의 비율로서 정의된다.

R-정확률 = 상위 *R* 개의 문서들에 포함된 적합 문서 수 / *R*

예를 들어, <표 5.2>의 검색 결과를 생성하기 위해 사용된 질의를 만족하는 적합 문서들은 전체 문서 집합에 41 개가 존재하며, 상위 41 위 이내에서 검색된 적합 문서들의 수는 15 개이기 때문에, *R*-정확률은 15/41=0.0.3659 로 계산된다.

한편, 적합 문서들의 수가 100 개를 초과하는 경우에 *R*-정확률의 계산에 있어 문제점이 발생한다. 학술 대회 TREC 에서는 이러한 경우에 대하여 문서 순위 101 부터는 적합 문서가 검색되지 않는다고 가정하고 *R*-정확률을 계산한다. 예를 들어, 적합 문서의 수가 200 개일 경우, 상위 100 위 이내에 포함된 적합 문서의 수가 21 개이고 101 위부터는 적합 문서가 검색되지 않는다고 가정하였기 때문에, *R*-정확률은 21/200=0.105 로 계산된다.

제 6 장 영어 색인어 추출

전통적으로 문서로부터 색인어들을 추출하는 색인 작업은 훈련된 사서나 주제 전문가에 의해 수작업으로 수행되어 왔다. 그러나, 1950 년대에 색인 대상이 되는 문서 수의 급속한 증가로 인하여 수작업 색인은 많은 시간과 비용을 요구하게 되었다. 이러한 문제점을 해결하기 위해 컴퓨터를 이용한 자동 색인 기법들이 연구되었으며, 자동 색인 기법들은 컴퓨터에 입력된 문서의 텍스트를 분석하여 이 문서의 내용을 대표할 수 있는 단어 또는 단어구들의 추출을 목적으로 하고 있다.

자동 색인에 대한 최초의 연구 결과는 1957 년에 룬(Luhn)에 의해 발표되었으며, 이후 룬의 연구 결과는 다양한 자동 색인 기법의 기초가 되었다. 룬은 문서로부터 주제어들을 추출하기 위한 기준으로서 문서에 출현하는 단어의 출현 빈도를 사용하였다. 즉, 문서에 매우 빈번히 출현하는 고빈도의 단어들과 매우 드물게 출현하는 저빈도의 단어들은 주제어로서 가치가 없다고 생각하였다. 따라서 최고 한계 빈도와 최저 한계 빈도를 정하고, 이 두개의 한계 빈도 내에 속하는 중간 빈도의 단어들을 색인어로 선정할 것을 제안하였다. 그러나, 룬은 두 한계 빈도의 구체적인 산출 방법은 제시하지 않았다.

1958 년 박센데일(Baxendale)은 기능어를 제외한 모든 단어를 색인어로 선정할 것을 제안하였다. 여기에서 기능어는 관사, 전치사, 접속사, 대명사 등 주제적 의미를 지니지 않은 단어들을 총칭한다. 일반적으로 텍스트에 포함된 기능어를 색인어 선정에서 제외할 경우, 색인할 단어들의 수가 절반 이상 줄어드는 것으로 알려져 있다. 이러한 생각은 대부분의 현대 정보 검색 시스템들에 채택되어 불용어 제거라고 불리는 색인어 추출 과정의 일부로 사용되고 있다. 본 장에서는 영어 문서 또는 질의로부터 색인어를 추출하는 3 단계 과정 즉, 어휘 분석, 불용어 제거, 스테밍에 대하여 기술한다.

6.1 어휘 분석

어휘 분석은 입력된 문자열을 토큰들로 변환하는 과정으로 질의 처리와 자동 색인에 요구되는 기본적인 과정이다. 어휘 분석기의 설계에 있어서 첫번째로 고려할 사항은 토큰들의 형태를 정의하는 것이다. 영어 색인어 추출에 있어서 토큰은 연속적인 알파벳 a-z, A-Z 문자열로 쉽게 정의될 수 있으나, 이러한 정의는 다양한 문제점들을 발생시킬 수 있다. 따라서 토큰의 형태를 정의하기 위해서는 다음과 같은 사항들이 고려되어야 한다.

- 숫자: 많은 경우에 연속된 숫자는 색인어로서 부적합하기 때문에, 연속된 숫자는 토큰으로 고려되지 않는다. 그러나, 특정 문서 집합에서 숫자는 중요한 의미를 지닐 수 있다.

예를 들면, 신문 기사 데이터베이스에서 어떠한 사건의 발생 일자는 검색에 도움을 줄 수 있다. 또한, 기술을 다루는 문서들에서 알파벳 문자와 숫자로 구성된 토큰들은 색인어로서 가치가 높다. 예를 들면, 비타민에 대한 문서들은 “B6”, “B12”과 같은 중요한 토큰들을 포함한다.

- 하이픈: 다수의 영어 단어들이 하이픈으로 연결되어 있을 경우, 각각의 단어들을 하나의 토큰으로 인식할 것인지 또는 하이픈으로 연결된 전체를 하나의 토큰으로 취급할 것인지를 결정해야 한다. 예를 들면, “state–of–the–art”라는 스트링이 주어졌을 때, “state–of–the–art”에 포함된 단어들 “state”, “of”, “the”, “art” 각각을 하나의 토큰으로 인식할 것인지 또는 “state–of –the–art”라는 하나의 토큰으로 취급할 것인지를 결정해야 한다. 토큰이 하이픈을 포함하는 것이 적합한 경우로는 “F–16”, “MS–DOS”와 같은 예를 들 수 있다.

- 구두점: 하이픈과 유사하게 다양한 구두점들이 토큰의 일부분으로 인식되는 것이 적합한 경우들이 존재한다. 예를 들면, 마침표는 컴퓨터 시스템의 파일 이름 작성에 이용되거나(예, COMMAND.COM), 어떠한 사건을 지칭할 때(예, 6.25 전쟁, 5.18 광주 항쟁) 사용될 수 있으며, 슬래시는 상품의 이름 등에 사용될 수 있다(예, OS/2). 또한, 숫자를 색인어로 간주한다면, 콤마와 마침표를 포함한 숫자가 인식되어야 한다(예, 1,234,567.890).

6.2 불용어 제거

정보 검색에 대한 연구가 시작된 초기부터 영어에서 빈번하게 사용되는 “the”, “of”, “and”, “to” 등과 같은 단어들은 색인어로서 가치가 없는 것으로 인식되어 왔다. 그 이유는 대부분의 문서들은 이러한 단어들을 포함하고 있으며, 따라서 질의로서 이러한 단어들의 사용은 데이터베이스에 포함된 대부분의 문서를 검색하기 때문이다. 이처럼 대다수의 문서들에 빈번히 사용되어 검색에 도움이 되지 않는 단어들은 “불용어(stopword)”라고 지칭되며, 일반적으로 관사, 전치사, 접속사는 불용어 목록에 포함된다. 예를 들면, 일반적인 영어 문서들에 포함된 100 만개의 단어를 조사한 결과, “the”와 “of” 두개의 단어가 전체 단어의 10%를 차지하였으며, 그 다음으로 “and”, “to”, “a”, “in” 네개의 단어가 전체 단어의 10%를 차지하였다.

불용어 제거는 색인 파일의 크기를 상당히 감소시키기 때문에, 많은 정보 검색 시스템들은 관사, 전치사, 접속사 이외에 동사, 부사, 형용사의 일부를 불용어 목록에 포함한다. 적절한 불용어 제거는 색인 파일의 크기를 40% 이상 감소시키는 것으로 알려져 있다. 또한, 불용어 제거는 검색을 수행하는 시간을 단축시킨다. 색인 파일에는 각각의 색인어에 대하여 그 색인어를 포함하고 있는 문서들의 식별자가 저장된다. 어떠한 색인어가 매우 많은 수의

문서에 출현할 경우, 그 색인어에 대한 문서 식별자 리스트가 매우 커지게 된다. 따라서 커다란 크기의 문서 식별자 리스트를 갖는 불용어를 질의 처리에서 배제함으로써, 입출력 시간과 유사도 계산 시간을 감소시키고, 전체 질의 처리 시간을 단축시킬 수 있다.

색인 파일의 크기를 감소시키고 검색 시간을 단축시킬 수 있음에도 불구하고, 불용어 제거는 재현률의 감소라는 문제를 발생시킨다. 예를 들면, 사용자가 “to be or not to be”라는 구를 검색할 경우, 전치사와 접속사를 불용어로서 제거한다면 단지 “be”라는 단어만이 검색에 이용된다. 이때 “be”라는 단어만으로는 “to be or not to be”라는 구를 포함하는 문서를 검색하는 것은 거의 불가능에 가깝다. 이러한 경우를 염두에 두고 일부 정보 검색 시스템들은 불용어 제거를 수행하지 않고 있다.

6.3 스테밍

동일한 의미를 갖는 단어들의 다양한 변형들을 하나의 색인어로 변환하는 스테밍은 1960 년대부터 정보 시스템의 자동화에 적용되었다. 예를 들면, 영어 스테밍은 문서 또는 질의에 포함된 “computers”, “computing”, “compute”, “computed”, “computable”, “computation” 등의 단어들을 “compute”라는 색인어로 변환한다. 정보 검색 시스템에서 스테밍의 사용은 저장 공간의 사용을 감소시키며 검색 속도의 개선에 기여한다. 이는 스테밍이 다수의 단어들을 하나의 색인어로 변환하기 때문에, 각각의 단어들 모두를 색인어로 취급하는 방법에 비하여 색인어 수를 50% 이상 감소시키기 때문이다. 또한 스테밍은 검색 결과의 질을 향상시킨다. 예를 들면, 질의로 “compute”를 입력하였을 경우, “computers”, “computing”, “computed”, “computable”, “computation” 를 포함하는 문서들을 검색할 수 있다.

스테머는 과도 스테밍(overstemming)과 과소 스테밍(understemming)이라 불리는 2 가지 종류의 오류를 발생시킬 수 있다. 과도 스테밍은 단어에 포함된 문자들을 너무 많이 제거하기 때문에, 서로 연관성이 없는 단어들의 매칭을 발생시키며, 따라서 질의에 부적합한 문서의 검색을 야기한다. 과소 스테밍은 단어에 포함된 문자들을 너무 적게 제거하기 때문에, 서로 연관성이 있는 단어들이 매칭 되지 않으며, 따라서 질의에 적합한 문서가 검색되지 않는 현상을 발생시킨다. 즉, 과도 스테밍은 정확률을 희생하여 재현율을 증가시키며, 과소 스테밍은 재현율을 희생하여 정확률을 증가시킨다. 많은 웹 검색 시스템들은 스테밍을 수행하지 않고 있다. 과도 스테밍과 과소 스테밍의 이러한 특성으로 인하여, 재현율보다 정확률을 중요시하는 웹 정보 검색 시스템들은 스테밍을 수행하지 않는 경향을 지니고 있다.

현재 널리 사용되고 있는 스테머들은 최장 일치법을 순환적으로 사용하여 접사를 제거한다. 이러한 스테머들은 순환 최장 일치 스테머(iterative longest match stemmer)로 분류될 수

있으며, Lovins(1968)에 의해 최초로 개발된 이후 Salton(1968), Dawson(1974), Porter(1980), Paice(1990) 등에 의해 연구가 지속되었다. 이들 중에서 Porter 에 의해 개발된 스테머가 가장 널리 활용되고 있기 때문에, 다음에서는 Porter 의 스테머에 대해 자세히 다루기로 한다.

Porter 의 스테머는 접두사는 제거하지 않고 접미사만을 제거하거나, 새로운 스트링으로 대치한다. Porter 의 스테머는 스테밍을 위한 다양한 규칙을 포함하고 있으며, 각각의 규칙은 다음과 같은 4 개의 항목에 의해 정의된다.

- 규칙 번호
- 접미사
- 대치 스트링
- 스템 상태

규칙 번호는 규칙에 할당되는 고유 번호로서, 다수의 규칙이 적용 조건을 만족할 경우 작은 번호의 규칙이 우선적으로 적용된다. 주어진 단어에 어떠한 규칙이 적용되기 위해서는 2 가지 조건이 만족되어야 한다. 첫째, 단어가 규칙에 정의된 접미사를 포함해야 한다. 둘째, 단어로부터 접미사를 제거한 스템이 스템 상태에서 지정된 조건을 만족해야 한다. 이러한 조건들을 만족할 경우, 단어에 포함된 접미사는 규칙에 정의된 대치 스트링으로 치환된다. 만약 접미사가 NULL 로 지정되어 있으면 단어에 대치 스트링을 연결하고, 대치 스트링이 NULL 로 지정되어 있으면 단어에 포함된 접미사를 제거한다.

한편, 스템 상태의 조건을 지정하기 위하여 다음과 같은 기호들이 사용된다. 첫째, m 은 스템의 크기는 나타낸다. 어근의 크기는 단순히 알파벳의 수를 의미하지 않으며, 모음-자음의 쌍 하나를 크기 1 로 계산한다. 여기에서 모음은 5 개의 문자 a, e, i, o, u 그리고 자음에 연이어 나타나는 y 를 말한다. 예를 들면, “relation”이라는 단어가 있을 때, 이 단어는 “el”, “at”, “on”이라는 3 개의 모음-자음의 쌍을 포함하기 때문에 어근의 크기가 3 으로 계산된다. 둘째, *v* 는 스템이 모음을 포함하고 있음을 의미한다. 셋째, *o 는 스템의 마지막이 자음-모음-자음의 순으로 끝나며, 마지막 자음이 w, x, y 가 아님을 의미한다.

Porter 는 위에서 설명된 기호를 이용하여 스테밍 규칙들을 정의하였다. 이러한 규칙들은 여러 단계로 구분되며, 각 단계에서 하나의 규칙만이 적용된다. Porter 스테머는 주어진 단어에 각각의 단계를 차례대로 적용한다. 즉, 단계 1a, 단계 1b, 단계 1b1, 단계 1c, 단계 2, 단계 3, 단계 4, 단계 5a, 단계 5b 를 차례대로 적용한다. 단. 단계 1b1 은 단계 1b 의 106 또는 107 번 규칙이 사용되었을 경우에만 적용된다. 다음에서는 Porter 스테머의 단계별로 구분된 규칙들에 대하여 기술한다.

• **단계 1a**

101	sses	ss	NULL	caresses→caress
102	ies	i	NULL	ponies→poni
103	ss	ss	NULL	carress→carress
104	s	NULL	NULL	cats→cat

• 단계 1b

105	eed	ee	m>0	feed→feed agreed→agree
106	ed	NULL	*v*	plastered→plaster bled→bled
107	ing	NULL	*v*	motoring→motor sing→sing

• 단계 1b1

108	at	ate	NULL	conflat(ed) →conflate
109	bl	ble	NULL	troubl(ing) →trouble
110	iz	ize	NULL	siz(ed) →size
111	bb	b	NULL	
112	dd	d	NULL	
113	ff	f	NULL	
114	gg	g	NULL	
115	mm	m	NULL	
116	nn	n	NULL	tann(ed) →tan
117	pp	p	NULL	hopp(ing) →hop
118	rr	r	NULL	
119	tt	t	NULL	
120	ww	w	NULL	
121	xx	x	NULL	
122	NULL	e	m=1 and *o	fail(ing) →fail fil(ing) →file

• 단계 1c

123	y	i	*v*	happy→happi sky→sky

• 단계 2

203	ational	ate	m>0	relational→relate
204	tional	tion	m>0	conditional→condition
205	enci	ence	m>0	valenci→valence
206	anci	ance	m>0	hesitanci→hesitance
207	izer	ize	m>0	digitizer→digitize
208	abli	able	m>0	conformabli→conformable
209	alli	al	m>0	radicalli→radical
210	entli	ent	m>0	differentli→different
211	eli	e	m>0	vileli→vile
213	ousli	ous	m>0	analogousli→analogous

214	ization	ize	m>0	vietnamization→vietnamize
215	ation	ate	m>0	predication→predicate
216	ator	ate	m>0	operator→operate
217	alism	al	m>0	feudalism→feudal
218	iveness	ive	m>0	decisiveness→decisive
219	fulnes	ful	m>0	hopefulness→hopeful
220	ousness	ous	m>0	callousness→callous
221	aliti	al	m>0	formaliti→formal
222	iviti	ive	m>0	sensitiviti→sensitive
223	biliti	ble	m>0	sensibiliti→sensible

• 단계 3

301	icate	ic	m>0	triplicate→triplic
302	ative	NULL	m>0	formative→form
303	alize	al	m>0	formalize→formal
304	iciti	ic	m>0	electriciti→electric
305	ical	ic	m>0	electrical→electric
308	ful	NULL	m>0	hopeful→hope
309	ness	NULL	m>0	goodness→good

• 단계 4

401	al	NULL	m>1	revival→reviv
402	ance	NULL	m>1	allowance→allow
403	ence	NULL	m>1	inference→infer
405	er	NULL	m>1	airliner→airlin
406	ic	NULL	m>1	gyroscopic→gyroscop
407	able	NULL	m>1	adjustable→adjust
408	ible	NULL	m>1	defensible→defens
409	ant	NULL	m>1	irritant→irrit
410	ement	NULL	m>1	replacement→replac
411	ment	NULL	m>1	adjustment→adjust
412	ent	NULL	m>1	dependent→depend
413	sion	s	m>1	
414	tion	t	m>1	adoption→adopt
415	ou	NULL	m>1	homologou→homolog
416	ism	NULL	m>1	communism→commun
417	ate	NULL	m>1	activate→activ
418	iti	NULL	m>1	angulariti→angular
419	ous	NULL	m>1	homologous→homolog
420	ive	NULL	m>1	effective→effect
421	ize	NULL	m>1	bowdlerize→bowdler

• 단계 5a

501	e	NULL	m>1	probate→probat rate→rate
502	e	NULL	m=1 and not *o	cease→ceas

• **단계 5b**

503	ll	l	m>1	controll→control roll→roll

위에서 설명된 Porter 스테머를 사용하여 영어 단어 “relations”를 스테밍하는 과정은 다음과 같다. (1) 단계 1a: 규칙 104 를 적용하여 “relations”를 “relation”으로 변환 (2) 단계 1b: 적용되는 규칙 부재 (3) 단계 1b1: 규칙 106 또는 107 이 적용되지 않았기 때문에 단계 1b1 을 무시 (4) 단계 1c: 적용되는 규칙 부재 (5) 단계 2: 규칙 215 를 적용하여 “relation”을 “relate”로 변환 (5) 단계 3: 적용되는 규칙 부재 (6) 단계 4: 적용되는 규칙 부재 (7) 단계 5a: 규칙 501 을 적용하여 “relate”를 “relat”로 변환 (7) 단계 5b: 적용되는 규칙 부재. 따라서 Porter 스테머는 “relations”을 “relat”로 변환함을 알 수 있다.

제 7 장 렉시콘 구조

다수의 페이지들로 구성된 책에서 모든 페이지들에 대한 검토 없이 원하는 내용을 기술하고 있는 페이지를 신속하게 발견하기 위하여 일반적으로 책 뒷부분에 첨부되어 있는 색인이 이용된다. 이러한 페이지 색인은 본문으로부터 추출된 색인어와 이 색인어가 출현하는 페이지 번호들의 쌍들로 구성된다. 그리고, 페이지 색인에 포함된 색인어들은 오름차순으로 정렬되어 있기 때문에, 사용자들은 사전에서 단어를 찾듯이 원하는 색인어를 쉽게 검색할 수 있다.

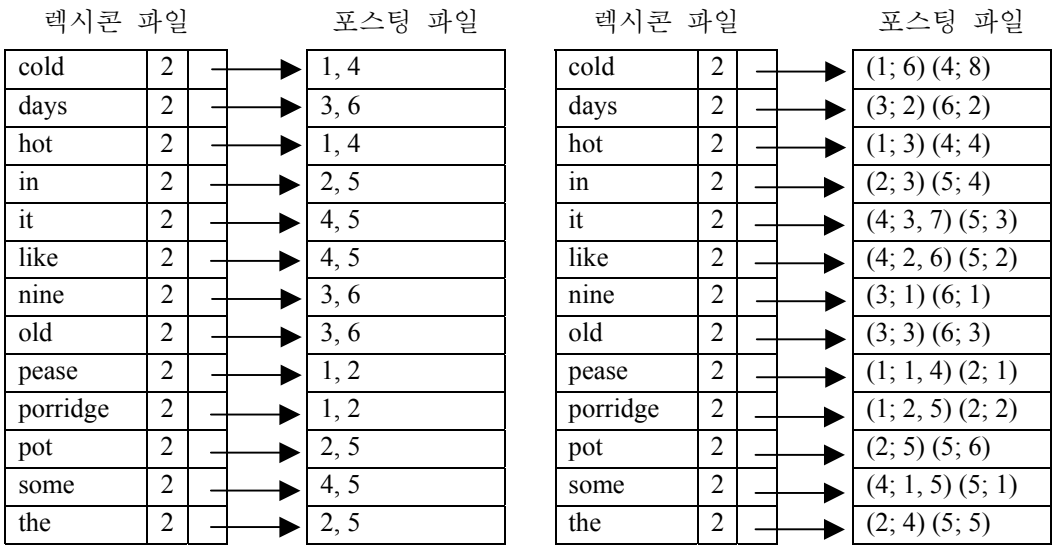
한편, 수 많은 문서들로부터 사용자 질의를 만족하는 문서들을 검색하는 정보 검색 시스템도 페이지 색인과 유사한 형태의 문서 색인을 생성한다. 이러한 문서 색인은 문서로부터 추출된 색인어와 이 색인어가 출현한 위치 정보의 쌍들로 구성되며, 문서로부터 추출된 색인어들과 색인어들의 위치 정보는 각각 렉시콘(Lexicon)과 포스팅(Posting)으로 지칭된다.

일반적으로 렉시콘과 포스팅은 별도의 파일에 저장되기 때문에, 렉시콘 파일에는 문서로부터 추출된 색인어와 더불어 문서 빈도 및 포스팅에 대한 포인터가 저장된다. 포스팅은 근접 연산을 지원하지 않을 경우 색인어가 출현한 문서 번호들만으로 구성되며, 근접 연산을 지원할 경우 색인어가 출현한 문서 번호들과 문서 내에서 색인어가 출현한 위치 번호들로 구성된다. 예를 들어, <그림 7.1>은 <표 7.1>의 문서들로부터 생성된 렉시콘 파일과 포스팅 파일을 보여준다. 정보 검색 시스템은 렉시콘에서 질의에 포함된 검색어를 발견한 후, 포스팅에 대한 포인터를 통하여 검색어와 관련된 포스팅에 접근함으로써, 질의 처리에 필요한 정보들을 획득한다.

본 장에서는 저장 공간을 효율적으로 사용하는 다양한 렉시콘 구조에 대하여 기술한다. 렉시콘 구조는 크게 동적 렉시콘과 정적 렉시콘으로 구분될 수 있다. 동적 렉시콘은 색인어의 삽입과 삭제가 자유로운 구조를 지니고 있으며, 정적 렉시콘은 최초에 렉시콘이 생성된 이후에 색인어의 삽입과 삭제가 어려운 구조를 지니고 있다. 또한, 동적 렉시콘과 정적 렉시콘은 모두 메모리 상주 여부에 따라 메모리 기반 렉시콘과 디스크 기반 렉시콘으로 구분될 수 있다.

<표 7.1> 예제 문서

문서 번호	문서 내용
1	Pease porridge hot, pease porridge cold
2	Pease porridge in the pot
3	Nine days old
4	Some like it hot, some like it cold
5	Some like it in the pot
6	Nine days old



(a) 근접 연산을 지원하지 않을 경우 (b) 근접 연산을 지원할 경우

<그림 7.1> 렉시콘 파일과 포스팅 파일

7.1 동적 렉시콘

동적 렉시콘은 최초에 렉시콘이 생성된 이후에도 색인어의 삽입과 삭제가 자유로운 구조를 지니고 있으며, 일반적으로 AVL 트리와 B+ 트리와 같은 트리 자료 구조를 이용하여 구현된다. 문서로부터 추출된 색인어의 수가 적을 경우 전체 렉시콘의 메모리 상주가 바람직하며, 렉시콘의 메모리 상주는 렉시콘 탐색에 소비되는 시간을 단축시킬 수 있다. 메모리 기반의 자료 구조인 AVL 트리는 노드의 삽입과 삭제로 인하여 균형을 상실하는 이진 트리의 문제점을 개선한 방법으로서, 임의의 노드에 대한 좌우 서브 트리의 깊이 차이가 1이하인 범위 내에서 트리의 균형을 유지한다. 따라서 AVL 트리는 색인어의 삽입 및 삭제와 더불어 효율적인 검색을 지원한다.

문서로부터 추출된 색인어의 수가 많아서 전체 렉시콘의 메모리 상주가 불가능할 경우, 렉시콘을 디스크에 저장하고 색인어를 효율적으로 삽입, 삭제 및 검색하는 자료 구조가 요구되며, B+ 트리 자료 구조는 이러한 요구 사항을 만족한다. B+ 트리는 항상 완전 균형을 이루고 있으며, 색인어 수가 매우 많을 경우에도 3-4 정도의 낮은 높이를 유지한다. 따라서, 대용량 동적 렉시콘의 관리에 적합한 자료 구조이다. 또한, 연결 리스트로 구성된 리프 노드들은 색인어들에 대한 순차적 접근을 지원하기 때문에, 색인어 조회와 같은 연산을 효율적으로 수행할 수 있다.

7.2 메모리 기반 정적 렉시콘

7.2.1 고정 길이 색인어 필드

일반적으로 렉시콘은 색인어, 문서 빈도, 포스팅 위치와 같은 3개 필드들을 지닌 레코드들로 구성되며, 또한 색인어를 통하여 이들 레코드들에 효율적으로 접근할 수 있는 구조를 지녀야 한다. 문서 빈도는 색인어를 포함하는 문서 수로서 정수형 필드이고, 포스팅 위치는 색인어가 출현한 문서 번호 등에 대한 정보를 지니고 있는 포스팅에 대한 포인터로서 문서 빈도 필드와 마찬가지로 정수형 필드이다. 문서 빈도와 포스팅 위치 필드의 길이로는 4 바이트가 사용된다.

<그림 7.2>는 고정 길이 색인어 필드를 사용하는 메모리 기반 정적 렉시콘을 보여준다. 이 렉시콘 구조는 색인어 필드의 길이를 고정시키기 때문에 전체 레코드의 길이도 고정된다. 또한, 레코드들이 색인어 필드를 기준으로 오름차순으로 정렬되어 저장되기 때문에, 색인어에 의한 레코드 접근은 이진 탐색에 의해 효율적으로 이루어진다.

고정 길이 색인어 필드를 사용하는 메모리 기반 정적 렉시콘은 가장 단순한 형태의 렉시콘 구조로서, 구현이 쉽기 때문에 적은 수의 문서들을 대상으로 하는 검색 시스템에 적합한 구조이다. 그러나 색인어 필드의 길이가 최대 길이의 색인어에 의해 결정되기 때문에, 많은 양의 저장 공간이 낭비된다. 예를 들어, 색인어의 최대 길이가 20바이트라고 가정하면, 모든 레코드들의 길이는 28바이트가 되며, 렉시콘에 1백만 개의 색인어들을 저장할 경우 약 28M 바이트의 저장 공간이 요구된다. 그러나, 색인어의 평균 길이가 8바이트라고 가정하면, 약 12M 바이트의 저장 공간이 낭비됨을 알 수 있다.

<그림 7.2> 고정 길이 색인어 필드를 사용하는 메모리 기반 정적 렉시콘

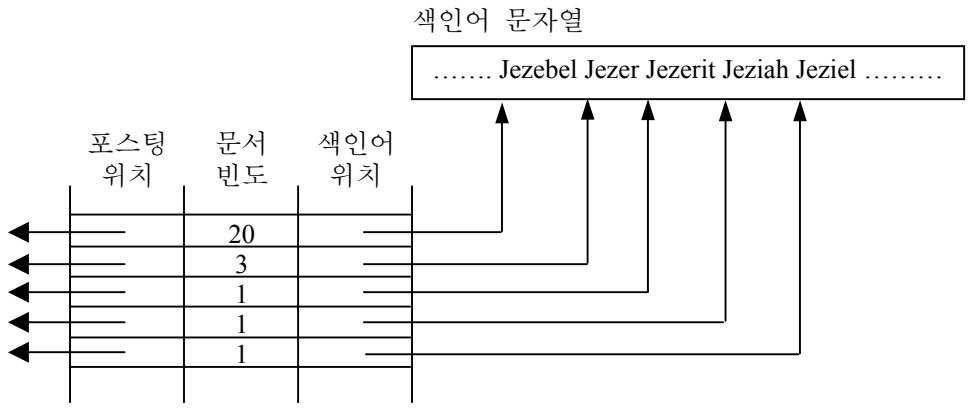
7.2.2 가변 길이 색인어 필드

가변 길이 색인어 필드를 사용하는 메모리 기반 정적 렉시콘은 고정 길이 색인어 필드의 사용으로 인하여 저장 공간이 낭비되는 문제점을 개선한 렉시콘 구조이다. <그림 7.3>은 가변 길이 색인어 필드를 사용하는 메모리 기반 정적 렉시콘을 보여준다. 이 렉시콘 구조에서 색인어는 고정 길이 색인어 필드에 저장되지 않고, 색인어 위치 필드의 값이 지정하는 곳에 저장된다. 이때 모든 색인어들은 하나의 연속된 문자열 형태로 저장되며, 또한 이 색인어들을 지시하는 레코드들과 동일한 순서로 저장된다. 따라서, n번째 색인어의 길이는 $n+1$번째 색인어의 위치 값으로부터 n번째 색인어의 위치 값을 뺀 값이 된다.

색인어 위치 필드는 4 바이트 길이의 정수형으로 고정 길이이기 때문에, 색인어 위치, 문서 빈도, 포스팅 위치 필드들로 구성된 레코드의 길이도 고정된다. 또한, 고정 길이

색인어 필드를 사용하는 렉시콘 구조와 마찬가지로 레코드들이 색인어 위치 필드의 값이 지정하는 색인어 필드를 기준으로 오름차순으로 정렬되어 저장된다. 따라서, 이 렉시콘 구조에서도 색인어에 의한 레코드 접근은 이진 탐색에 의해 이루어진다.

한편, 가변 길이 색인어 필드를 사용하는 메모리 기반 정적 렉시콘은 모든 색인어들을 하나의 연속된 문자열 형태로 저장하기 때문에, 고정 길이 색인어 필드를 사용할 경우 공백 문자들로 인하여 발생하는 저장 공간 낭비의 문제점을 개선한다. 예를 들어, 색인어 위치 정보, 문서 빈도, 포스팅 위치 필드의 크기가 모두 4바이트이고 색인어의 평균 길이가 8바이트라고 가정하면, 렉시콘에 1백만 개의 색인어를 저장할 경우 약 20M 바이트의 저장 공간이 요구된다. 즉, 고정 길이 색인어 필드를 사용할 경우에 비하여 8M 바이트의 저장 공간이 절약된다.



<그림 7.3> 가변 길이 색인어 필드를 사용하는 메모리 기반 정적 렉시콘

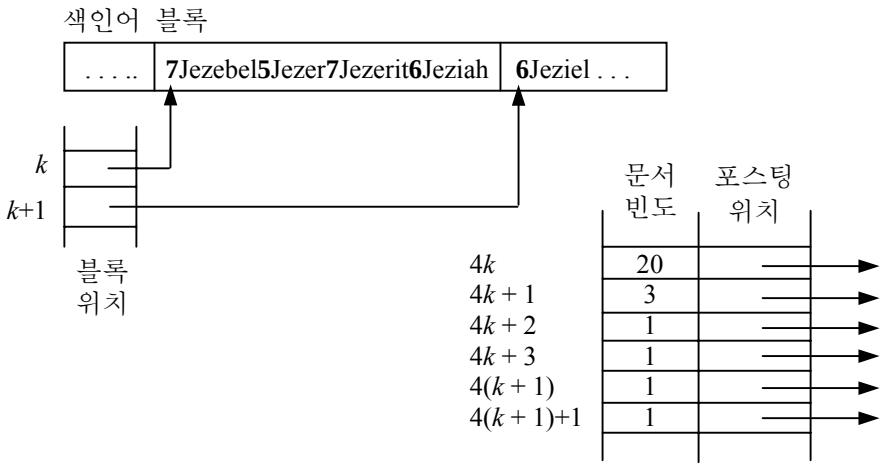
7.2.3 색인어 블록화

가변 길이 색인어 필드를 사용하는 렉시콘 구조에서 색인어들을 블록 단위로 분할하여 저장함으로써 저장 공간을 절약할 수 있다. 즉, <그림 7.4>와 같이 4개의 색인어를 하나의 블록으로 지정한 후, 색인어들의 위치를 저장하는 대신에 블록들의 위치만을 기록한다. 이때 블록 내에서 색인어들을 구분하기 위하여 각 색인어의 길이가 색인어 이전에 저장되어야 한다.

이 렉시콘 구조에서 원하는 색인어에 대한 포스팅을 획득하는 과정은 다음과 같이 3단계로 구분될 수 있다. 첫번째 단계에서 블록들에 대한 이진 탐색을 수행하여 원하는

색인어를 포함할 가능성이 있는 블록을 검색한다. 두 번째 단계에서는 발견된 색인어 블록에 포함된 색인어들에 대한 순차 탐색을 수행하여 원하는 색인어를 검색한다. 마지막으로 원하는 색인어가 k번째 블록의 n번째 색인어일 경우, 포스팅 위치 필드를 포함하는 $4k+n$번째 레코드를 통하여 포스팅을 획득할 수 있다.

한편, 색인어들의 블록화는 저장 공간의 절약을 제공한다. 예를 들어, 문서 빈도, 포스팅 위치, 블록 위치 필드의 크기가 모두 4바이트이고 색인어의 평균 길이가 8바이트라고 가정하자. 렉시콘에 1백만 개의 색인어를 저장할 경우, 문서 빈도와 포스팅 위치의 저장을 위하여 각각 4M 바이트, 블록 위치의 저장을 위하여 1M 바이트, 색인어 문자열과 색인어 문자열 길이의 저장을 위하여 9M 바이트가 요구되며, 전체적으로 약 18M 바이트의 저장 공간이 요구된다. 즉, 색인어들의 블록화를 수행하지 않은 경우에 비하여 2M 바이트의 저장 공간이 절약된다.



<그림 7.4> 색인어 블록화를 수행한 메모리 기반 정적 렉시콘

7.2.4 색인어 블록의 압축

프론트 코딩(Front Coding)은 정렬된 상태로 저장된 문자열들을 압축하는 기법이다. 정렬된 상태로 저장된 문자열들을 살펴보면 이웃하고 있는 색인어들은 문자열의 시작 부분이 동일함을 알 수 있다. 프론트 코딩은 동일한 부분의 문자열을 생략하고, 서로 다른 부분의 문자열만을 저장한다. 프론트 코드로 변환된 색인어는 이전 문자열과 중복된 문자열의 길이, 이전 문자열과 중복되지 않은 문자열의 길이, 이전 문자열과 중복되지 않은 문자열과 같은 3개의 항목으로 구성된다.

<표 7.2>의 두 번째 열은 전체 색인어 리스트에 대하여 프론트 코딩을 적용한 결과를 보여준다. 첫번째 색인어 “ Jezebel”은 이전 색인어가 존재하지 않으므로, 색인어 문자열의 길이 7과 색인어 문자열 “ Jezebel”로 변환된다. 두 번째 색인어 “ Jezer”은 첫번째 색인어와 중복된 문자열 “ Jeze”을 포함하므로, 중복된 문자열의 길이 4, 중복되지 않은 문자열의 길이 1, 그리고 중복되지 않은 문자열 “ r”로 변환된다. 마찬가지로, 세 번째 색인어 “ Jezerit”는 두 번째 색인어와 중복된 문자열 “ Jezer”을 포함하므로, 중복된 문자열의 길이 5, 중복되지 않은 문자열의 길이 2, 그리고 중복되지 않은 문자열 “ it”로 변환된다.

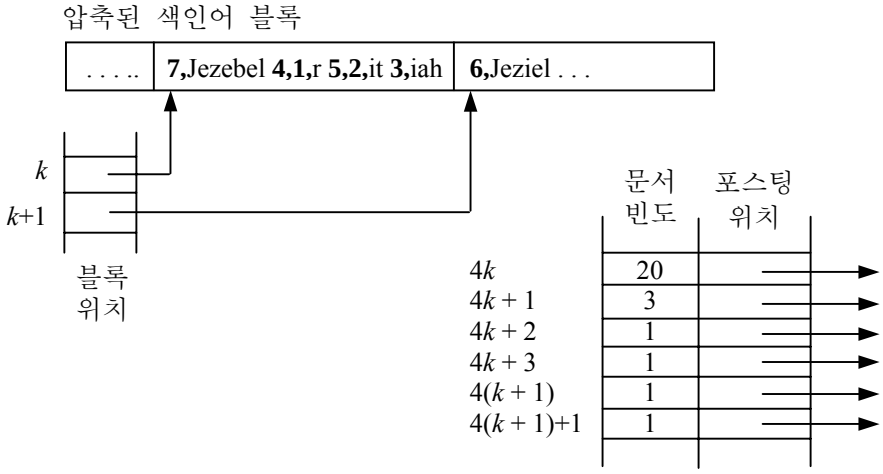
<표 7.2> 프론트 코딩 예제

색인어	전체 프론트 코딩	블록 프론트 코딩
Jezebel	7, Jezebel	7, Jezebel
Jezer	4, 1, r	4, 1, r
Jezerit	5, 2, it	5, 2, it
Jeziah	3, 3, iah	3, iah
Jeziel	4, 2, el	6, Jeziel
Jezliah	3, 4, liah	3, 4, liah
Jezoar	3, 3, oar	3, 3, oar
Jezrahiah	3, 6, rahiah	3, rahiah
Jezreel	4, 3, eel	, 7, Jezreel
Jezreelites	7, 4, ites	7, 4, ites
Jibsam	1, 5, ibsam	1, 5, ibsam
Jidlaph	2, 5, dlaph	2, dlaph

전체 색인어 리스트에 프론트 코딩을 적용하는 방법은 저장 공간의 절약이라는 측면에서 매우 효과적인 방법이나, 색인어 검색 시에 다음과 같은 문제점을 발생시킨다. 렉시콘에 존재하는 색인어들은 오름차순으로 정렬되어 있기 때문에, 프론트 코딩이 적용되지 않았을 경우 원하는 색인어를 검색하기 위하여 효율적인 이진 탐색이 사용될 수 있다. 그러나, 전체 색인어 리스트에 프론트 코딩이 적용되었을 경우, n번째 색인어의 문자열을 얻기 위해서는 첫번째부터 n번째까지의 색인어들에 대한 디코딩이 수행되어야 하기 때문에, 이진 탐색은 순차 탐색 보다도 많은 탐색 시간을 소비한다.

위에서 언급된 문제로 인하여 색인어들을 블록 단위로 분할하고, 블록 내의 색인어 리스트에 대하여 프론트 코딩을 적용하는 방법이 사용되며, <표 7.2>의 세 번째 열은 그 예를 보여준다. 첫번째부터 세 번째 색인어까지는 전체 프론트 코딩과 동일한 방법으로 코드가 생성되나, 블록의 마지막인 네 번째 색인어에 대해서는 문자열의 길이를 저장하지 않는다. 이는 네 번째 색인어의 문자열 길이를 두 번째 블록의 시작 위치를 이용하여 계산할 수 있기 때문이다. 두 번째 이후의 블록들에 대해서는 첫번째 블록과 동일한 방법으로 프론트 코딩을 적용한다.

<그림 7.5>는 색인어들의 블록화를 수행한 후, 블록 내의 색인어 리스트를 프론트 코딩을 사용하여 압축한 렉시콘 구조를 보여준다. 이 렉시콘 구조는 색인어 블록들이 압축되어 있다는 점을 제외하고는 7.2.3절에서 설명된 색인어 블록화를 수행한 렉시콘 구조와 동일하다. 또한 두 번째 단계에서 색인어 블록의 순차적 탐색 이전에 디코딩이 수행되는 점을 제외하면, 원하는 색인어에 대한 포스팅을 획득하는 과정도 색인어 블록화를 수행한 렉시콘 구조에서와 동일하다.



<그림 7.5> 색인어 블록을 압축한 메모리 기반 정적 렉시콘

한편, 색인어 블록을 프론트 코딩을 사용하여 압축함으로써 7.2.3절에서 설명된 색인어 블록화만을 수행한 렉시콘 구조와 비교하여 추가적인 저장 공간을 절약할 수 있다. 예를 들어, 문서 빈도, 포스팅 위치, 블록 위치 필드의 크기가 모두 4바이트이고, 색인어의 평균 길이와 압축된 색인어의 평균 길이가 각각 8바이트와 4바이트라고 가정하자. 하나의 블록에 포함된 4개 색인어들의 문자열 저장을 위해 필요한 공간은 8+ 4+ 4+ 4=20 바이트이므로 색인어 문자열들의 평균 길이는 5바이트이다. 그리고, 하나의 블록 내에 저장된 길이 정보는 총 6개이므로 하나의 색인어가 갖는 평균 길이 정보는 1.5바이트이다. 따라서,

렉시콘에 1백만 개의 색인어를 저장할 경우, 문서 빈도와 포스팅 위치의 저장을 위하여 각각 4M 바이트, 블록 위치의 저장을 위하여 1M 바이트, 색인어 문자열과 길이 정보의 저장을 위하여 6.5M 바이트가 요구되며, 전체적으로 약 15.5M 바이트의 저장 공간이 요구된다. 즉, 색인어 블록을 압축하지 않은 경우에 비하여 2.5M 바이트의 저장 공간이 절약된다.

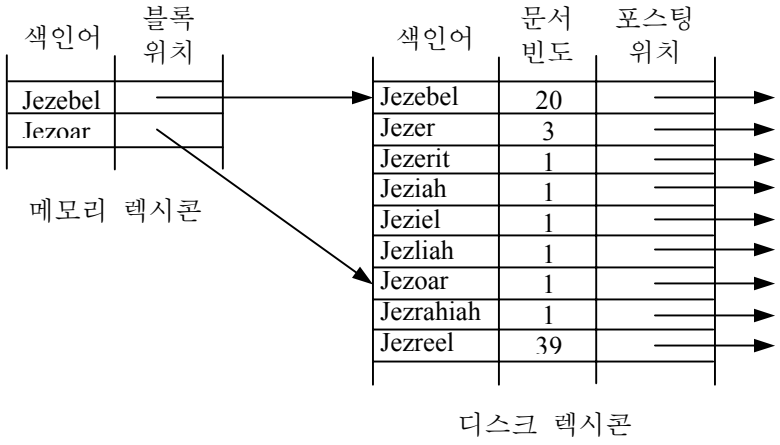
7.3 디스크 기반 정적 렉시콘

문서로부터 추출된 색인어의 수가 많아서 전체 렉시콘의 메모리 상주가 불가능할 경우, 디스크에 렉시콘을 저장하고 이 렉시콘에 포함된 색인어에 대한 효율적인 검색을 지원해야 한다. <그림 7.6>은 이러한 방법들 중의 하나인 2단계 렉시콘을 보여준다. 이 렉시콘 구조에서 디스크 렉시콘에는 색인어, 문서 빈도, 포스팅 위치 필드로 구성된 레코드들이 색인어 필드를 기준으로 오름차순으로 정렬된 후, 블록 단위로 분할되어 저장된다. 그리고 메모리 렉시콘에는 각 블록의 첫번째 색인어와 블록의 시작 위치로 구성된 레코드들이 색인어 필드를 기준으로 오름차순으로 정렬되어 저장된다.

디스크 렉시콘의 블록 크기는 효율적인 디스크 입출력을 위해서 운영 체제의 입출력 크기와 동일하게 설정되는 것이 바람직하며, 따라서 UNIX 운영 체제를 사용할 경우 적합한 블록 크기는 4096바이트이다. 또한, <그림 7.6>에서는 편의상 고정 길이 색인어 필드를 사용하는 렉시콘과 동일한 구조로 디스크 렉시콘이 표현되어 있으나, 보다 효율적인 메모리 기반 정적 렉시콘의 구조를 응용하여 디스크 렉시콘을 구현할 수 있다.

2단계 렉시콘 구조에서 원하는 색인어에 대한 포스팅을 획득하는 과정은 다음과 같이 2단계로 구분될 수 있다. 첫번째 단계에서 메모리 렉시콘에 대한 이진 탐색을 수행하여 원하는 색인어를 포함할 가능성이 있는 블록을 검색한다. 두 번째 단계에서는 발견된 색인어 블록 전체를 메모리에 적재한 후, 블록에 포함된 색인어들에 대한 이진 탐색을 수행하여 원하는 색인어를 검색한다.

한편, 2단계 렉시콘 구조를 사용하면 매우 적은 양의 메모리 공간만으로도 매우 많은 수의 색인어를 포함하는 렉시콘의 생성이 가능함을 알 수 있다. 예를 들어, 디스크 렉시콘의 블록의 크기가 4096바이트이고, 메모리와 디스크 렉시콘에 하나의 색인어를 저장하기 위해 평균적으로 20바이트가 필요하다고 가정하자. 이때 하나의 블록에는 약 200개의 색인어가 저장될 수 있다. 따라서, 렉시콘에 1천만 개의 색인어를 저장할 경우 약 5,000개의 블록이 생성되며, 메모리 렉시콘의 크기는 약 100K 바이트가 된다.



<그림 7.6> 디스크 기반 정적 렉시콘

제 8 장 색인 생성

색인 생성은 문서들로부터 추출된 색인어들과 이 색인어들이 출현한 문서의 문서 번호와 색인어 위치 등을 탐색이 용이한 형태로 파일에 저장하며, 일반적으로 색인 생성의 결과로서 렉시콘 파일과 포스팅 파일로 구성되는 역파일이 생성된다. 문서들로부터 추출된 색인어들은 렉시콘 파일에 저장되며, 색인어들이 출현한 문서의 문서 번호, 색인어 위치는 포스팅 파일에 저장된다. 정보 검색 시스템의 질의 처리기는 렉시콘을 탐색하여 질의에 포함된 색인어를 발견한 후, 포스팅 파일에 접근하여 이 색인어에 연결된 포스팅을 획득한다.

포스팅 생성을 효율적으로 수행하는 방법들은 크게 렉시콘과 포스팅을 동시에 생성하는 방법과 렉시콘 생성 후 포스팅을 생성하는 방법으로 구분될 수 있다. 전자는 문서들에 대한 색인어 추출을 1회만 수행하며, 포스팅의 생성을 위하여 연결 리스트를 이용한다. 후자는 문서들에 대한 색인어 추출을 2회 이상 수행한다. 즉, 첫번째 색인어 추출의 결과로서 정적 렉시콘과 색인어들에 대한 통계 정보를 생성하고, 이들을 이용하여 두 번째 이후의 색인어 추출 시에 포스팅을 생성한다.

8.1 렉시콘 생성

7장에서 설명된 바와 같이 렉시콘 구조는 크게 동적 렉시콘과 정적 렉시콘으로 구분될 수 있으며, 이들 각각은 메모리 상주 여부에 따라 메모리 기반 렉시콘과 디스크 기반 렉시콘으로 구분될 수 있다. 일반적으로 정보 검색 시스템이 활용되는 분야의 특성을 고려하여, 이에 적합한 렉시콘 구조가 사용된다. 예를 들어, 문서로부터 추출된 색인어 수가 매우 많으며, 최초에 색인이 생성된 이후에 문서의 삽입과 삭제가 없는 경우에는 디스크 기반 정적 렉시콘을 사용하는 것이 바람직하다.

한편, 다양한 형태의 렉시콘 구조와 관계없이 이러한 다양한 렉시콘 구조들의 초기 생성은 모두 메모리 기반 동적 렉시콘을 활용하여 다음과 같이 효율적으로 이루어질 수 있다.

1. 문서로부터 색인어들을 추출하고, 각각의 색인어에 대하여 메모리 기반 동적 렉시콘을 탐색한다. 렉시콘에 색인어가 존재하지 않을 경우, 이 색인어를 렉시콘에 삽입하고, 문서 빈도를 1로 지정한다. 또한, 렉시콘에 색인어가 존재할 경우, 이 색인어의 문서 빈도를 1만큼 증가시킨다.

2. 메모리 기반 동적 렉시콘에 지속적으로 색인어를 삽입함으로써 메모리가 부족할 경우, 그때까지 생성된 색인어들을 임시 렉시콘 파일에 오름차순으로 저장하고, 이들이 사용하던 메모리를 모두 반환한다.

3. 모든 문서들로부터 색인어들을 추출할 때까지 과정 1과 2를 반복함으로써, 다수의 임시 렉시콘 파일들을 생성한다. 이때 동일한 색인어가 여러 개의 임시 렉시콘 파일에 중복되어 저장될 수 있다.

4. 색인어들이 오름차순으로 저장되어 있는 임시 렉시콘 파일들을 하나의 렉시콘 파일로 합병하고, 이 렉시콘 파일을 활용하여 다양한 형태의 렉시콘 구조를 생성한다.

8.2 렉시콘과 포스팅의 동시 생성

포스팅 생성에 앞서 렉시콘이 생성되어 있지 않을 경우, 8.1절에서 설명된 렉시콘 생성 방법을 포스팅을 생성할 수 있도록 확장함으로써 렉시콘과 포스팅을 동시에 생성할 수 있다. 즉, 렉시콘 생성 방법에서와 동일하게 문서로부터 추출된 색인어들의 관리를 위하여 메모리 기반 동적 렉시콘을 사용하고, 추가적으로 문서 번호, 출현 빈도 필드들로 구성된 포스팅 레코드들의 관리를 위하여 연결 리스트를 사용한다.

1. 문서로부터 색인어들을 추출하여 색인어, 문서 번호, 출현 빈도 필드들로 구성된 레코드들을 생성하고, 각각의 레코드를 다음과 같이 처리한다.

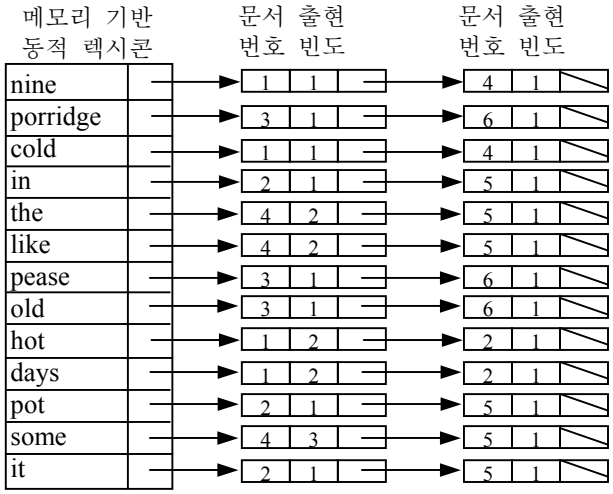
   1.1 메모리 기반 동적 렉시콘에 색인어가 존재하지 않을 경우 이 색인어를 렉시콘에 삽입하고, <그림 8.1>과 같이 색인어와 포스팅 레코드를 연결하여, 연결 리스트 구조의 포스팅을 생성한다.

   1.2 메모리 기반 동적 렉시콘에 색인어가 존재할 경우, 색인어가 지시하는 포스팅에 포스팅 레코드만을 연결한다. 이때 문서 번호를 사용하여 정렬된 순으로 포스팅 레코드들이 접근될 수 있도록, 새롭게 추가되는 포스팅 레코드는 포스팅의 마지막에 연결된다.

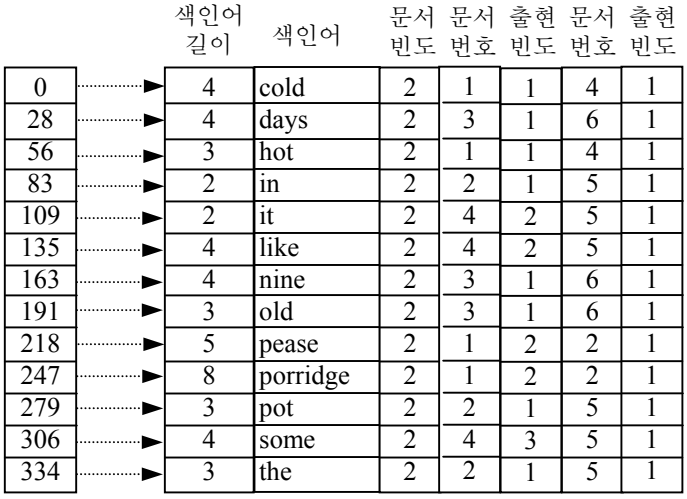
2. 지속적인 색인어 삽입과 포스팅 레코드 연결로 인하여 메모리가 부족할 경우, 그때까지 생성된 색인어들과 포스팅 레코드들을 <그림 8.2>와 같은 저장 구조를 사용하여 임시 파일에 저장하고, 이들이 사용하던 메모리를 모두 반환한다. <그림 8.2>의 저장 구조를 살펴보면, 색인어들과 포스팅 레코드들은 각각 문자열과 문서 번호를 기준으로 정렬되어 있음을 알 수 있다.

3. 모든 문서들로부터 색인어들을 추출할 때까지 과정 1과 2를 반복함으로써, 다수의 임시 파일들을 생성한다. 일반적으로 하나의 색인어는 많은 수의 문서에 출현하기 때문에, 동일한 색인어가 여러 개의 임시 파일에 중복되어 저장될 수 있다.

4. 문자열과 문서 번호를 기준으로 정렬되어 임시 파일에 저장되어 있는 색인어들과 포스팅 레코드들을 합병하여 최종적으로 하나의 렉시콘 파일과 하나의 포스팅 파일을 생성한다.



<그림 8.1> 연결 리스트를 이용한 포스팅 생성



<그림 8.2> 임시 렉시콘과 임시 포스팅의 저장 구조

8.3 렉시콘 생성 후 포스팅 생성

8.2절에서 설명된 렉시콘과 포스팅을 동시에 생성하는 방법은 임시 파일들의 생성으로 인하여 대용량의 문서들에 대한 색인을 생성할 경우, 많은 양의 임시 저장 공간을 요구하는 단점을 지니고 있다. 이에 비하여 렉시콘 생성 후 포스팅을 생성하는 방법들, 즉 문서 분할 방법과 렉시콘 분할 방법은 임시 저장 공간을 사용하지 않는 장점을 지니고 있다. 그러나, 문서 분할 방법은 문서에 대한 색인어 추출을 2회 수행하며, 렉시콘 분할 방법은 문서에 대한 색인어 추출을 2회 이상 수행하는 특성을 지니고 있다. 따라서, 형태소 분석기를 이용하여 한글 문서로부터 색인어를 추출할 경우, 색인 생성에 많은 시간이 소비되는 단점을 지니고 있다.

8.3.1 문서 분할

문서 분할 방법은 문서들에 대한 색인어 추출을 2회 수행한다. 첫번째 색인어 추출 과정에서는 전체 문서들에 대한 정적 렉시콘을 생성할 뿐만 아니라, 전체 문서들을 다수의 문서 블록들로 분할하고, 각각의 문서 블록들에 대하여 블록 단위 정적 렉시콘을 생성한다. 이 과정은 8.1절에서 설명된 렉시콘 생성 방법을 다음과 같이 수정함으로써 효율적으로 수행될 수 있다. 단, 문서 번호와 출현 빈도 필드들로 구성되는 포스팅 레코드의 크기가 8바이트이고, 메모리 상주 포스팅에 일정한 양의 메모리를 할당한 후에도, 블록 단위 정적 렉시콘을 메모리에 상주시킬 수 있음을 가정한다.

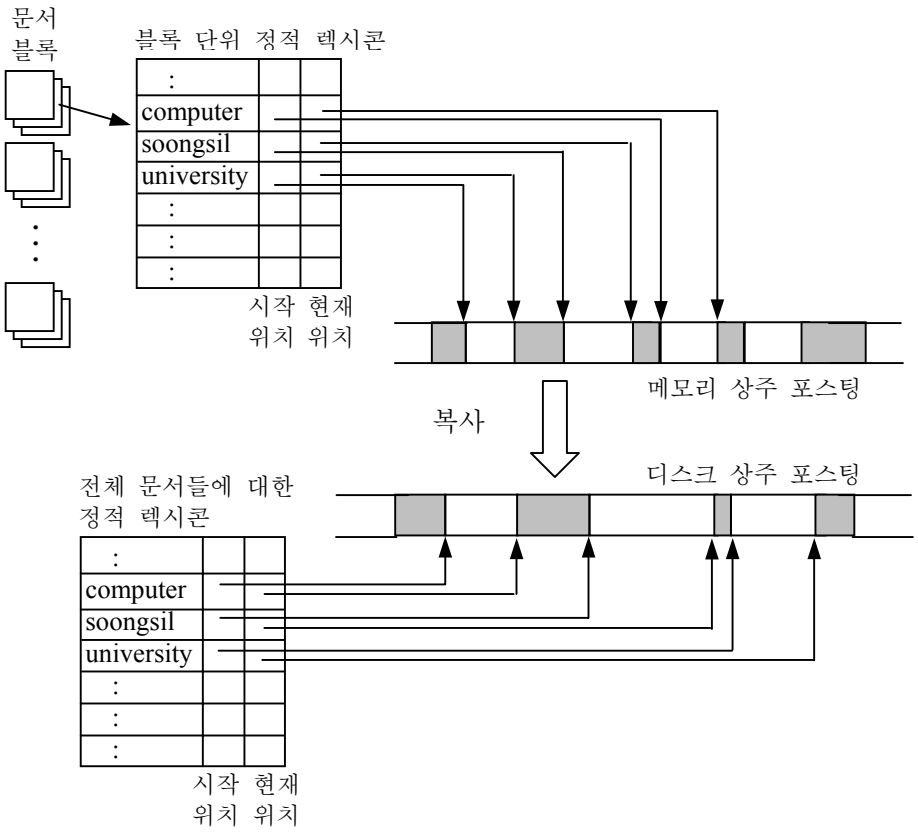
1. 문서로부터 색인어들을 추출하고, 각각의 색인어에 대하여 메모리 기반 동적 렉시콘을 탐색한다. 렉시콘에 색인어가 존재하지 않을 경우, 이 색인어를 렉시콘에 삽입하고, 문서 빈도를 1로 지정한다. 또한, 렉시콘에 색인어가 존재할 경우, 이 색인어의 문서 빈도를 1만큼 증가시킨다.

2. 하나의 문서로부터 추출된 색인어들에 대한 처리가 완료될 때마다, 메모리 기반 동적 렉시콘에 삽입된 모든 색인어들의 문서 빈도를 합산한 값에 8을 곱한다. 이 값이 메모리 상주 포스팅에 할당된 메모리 양에 근접할 경우, 지금까지 처리된 문서들을 하나의 문서 블록으로 분할한다. 또한, 현재의 메모리 기반 동적 렉시콘을 변환하여 블록 단위 정적 렉시콘 파일을 생성하고, 메모리 기반 동적 렉시콘이 사용하던 메모리를 모두 반환한다.

3. 모든 문서들로부터 색인어들을 추출할 때까지 과정 1과 2를 반복함으로써, 다수의 블록 단위 정적 렉시콘 파일들을 생성한다. 이때 동일한 색인어가 여러 개의 블록 단위 정적 렉시콘 파일에 중복되어 저장될 수 있다.

4. 색인어들이 오름차순으로 저장되어 있는 블록 단위 정적 렉시콘 파일들을 합병하여 전체 문서들에 대한 정적 렉시콘 파일을 생성한다.

두 번째 색인어 추출 과정에서는 메모리 상주 렉시콘을 위한 메모리를 할당한 후, 첫번째 색인어 추출 과정에서 생성된 블록 단위 정적 렉시콘과 전체 문서에 대한 정적 렉시콘을 기반으로 각각의 문서 블록에 대하여 다음을 반복적으로 수행함으로써 <그림 8.3>과 같이 포스팅을 생성한다. 첫째, 블록 단위 정적 렉시콘을 메모리에 적재하고, 문서 빈도를 이용하여 메모리 상주 포스팅에서 각각의 색인어에 대한 포스팅의 시작 위치를 계산한다. 둘째, 문서 블록에 포함된 문서들로부터 색인어들을 추출하여, 메모리 상주 포스팅을 생성한다. 마지막으로, 메모리 상주 포스팅을 디스크 상주 포스팅으로 복사한다.



<그림 8.3> 문서 분할을 이용한 색인 생성

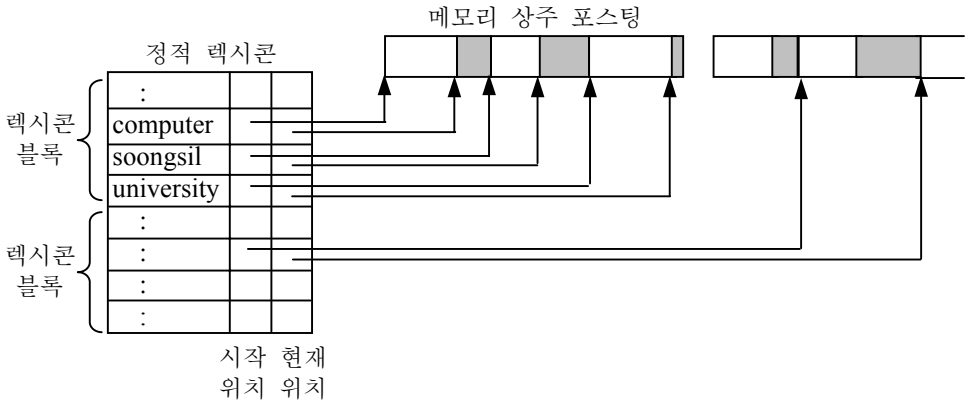
8.3.2 렉시콘 분할

렉시콘 분할 방법은 문서들에 대한 색인어 추출을 2회 이상 수행한다. 첫번째 색인어 추출 과정에서는 8.1절에서 설명된 렉시콘 생성 방법을 이용하여 전체 문서들에 대한 정적 렉시콘을 생성한다. 정적 렉시콘은 각각의 색인어에 대한 문서 빈도를 포함하고 있기 때문에, 이 색인어에 대한 포스팅을 저장하기 위해 요구되는 메모리 공간을 계산할 수 있다. 즉, 포스팅 레코드가 문서 번호와 출현 빈도 필드들로 구성될 경우, 임의의 색인어에 대한 포스팅의 크기는 이 색인어의 문서 빈도에 8을 곱한 값이 된다.

렉시콘 분할 방법은 <그림 8.4>와 같이 메모리 상주 포스팅에 일정한 양의 메모리를 할당하고, 첫번째 색인어 추출 과정에서 생성된 렉시콘을 다수의 렉시콘 블록들로 분할한다. 이때 하나의 렉시콘 블록에 포함된 색인어들의 문서 빈도를 합한 값에 8을 곱한 값은 메모리 상주 포스팅에 할당된 메모리 양을 초과할 수 없다.

렉시콘 분할 방법은 첫번째 색인어 추출이외에 렉시콘 블록 수만큼의 색인어 추출을 추가로 수행한다. 즉, 각각의 렉시콘 블록에 대하여 다음을 반복적으로 수행함으로써 <그림 8.4>와 같이 포스팅을 생성한다. 첫째, 렉시콘 블록을 메모리에 적재하고, 문서 빈도를 이용하여 메모리 상주 포스팅에서 각각의 색인어에 대한 포스팅의 시작 위치를 계산한다. 둘째, 모든 문서들로부터 렉시콘 블록에 포함된 색인어들을 추출하여, 메모리 상주 포스팅을 생성한다. 마지막으로, 메모리 상주 포스팅을 디스크에 저장한다.

문서 분할 방법은 문서에 대한 색인어 추출을 2회 수행하며, 렉시콘 분할 방법은 문서에 대한 색인어 추출을 2회 이상 수행하는 특성을 지니고 있다. 즉, 렉시콘 분할 방법은 정적 렉시콘 생성을 위해 1번의 색인어 추출을 수행하며, 포스팅 생성을 위하여 렉시콘 블록 수만큼의 색인어 추출을 수행한다. 일반적으로 색인어 추출의 횟수로 인하여 렉시콘 분할 방법은 문서 분할 방법에 비하여 색인 생성에 많은 시간을 소비한다. 그러나, 렉시콘 분할 방법은 모든 렉시콘 블록에 대한 포스팅 생성을 동시에 수행할 수 있는 장점을 지니고 있다.



<그림 8.4> 렉시콘 분할을 이용한 색인 생성

제 9 장 불리안 질의 처리

불리안 검색(Boolean Retrieval)은 구현이 용이하고 빠른 검색 시간을 제공하기 때문에, 대부분의 상용화된 정보 검색 시스템들은 불리안 검색을 지원하고 있다. 불리안 검색에서 문서는 색인어들의 집합으로 표현되며, 질의는 검색어와 논리 연산자 AND, OR, NOT 으로 구성된 불리안 수식이다. 예를 들어, “정보 검색 시스템”에 관련된 문서를 검색하기 위한 불리안 질의는 다음과 같이 작성될 수 있다.

“정보 AND 검색 AND 시스템”

불리안 검색은 질의로 주어진 불리안 수식을 만족하는 문서들을 사용자에게 제공하며, 따라서 위 질의의 검색 결과에 포함된 문서들은 질의에 출현하는 모든 검색어들을 포함하고 있다. 그러나, 문서 내에서 검색어들은 질의에 출현한 순서와 무관하게 출현하며, 또한 인접하여 출현하지 않을 수도 있다. 예를 들어, “패턴에 대한 *정보* 없이 *시스템* 내의 비정상적인 움직임 감지를 통해 바이러스를 *검색*하는”과 같은 문장을 포함하는 문서가 검색될 수 있다.

일반적인 불리안 질의는 <그림 7.1>에서 설명된 렉시콘 파일과 포스팅 파일을 이용하여 쉽게 처리될 수 있다. 즉, 불리안 질의에 포함된 검색어들을 렉시콘에서 탐색한 후, 포스팅에 대한 포인터를 통하여 검색어와 관련된 포스팅들을 획득한다. 그리고, 불리안 질의에 지정된 연산자에 따라 포스팅들의 합집합, 교집합, 차집합을 생성한다. 마지막으로 모든 불리안 연산이 수행된 후, 그 결과에 포함된 문서 번호들을 사용자에게 제공한다. 9.1 절과 9.2 절에서는 불리안 질의들 중에서 논리곱 질의와 논리곱 정규형 질의의 효율적인 처리 방법에 대하여 설명한다.

한편, 일반적으로 불리안 질의에 포함된 검색어에 절단 연산을 적용할 수 있다. 절단 연산자는 검색어의 일부에 임의의 문자 또는 문자열을 포함할 수 있도록 지정한다. 절단 연산자의 위치에 따라 좌측 절단, 우측 절단, 양측 절단, 중간 절단으로 구분될 수 있다. 우측 절단과 중간 절단은 7장에서 설명된 렉시콘 구조만을 사용하여 처리될 수 있다. 그러나, 좌측 절단과 양쪽 절단은 별도의 자료 구조를 필요로 한다. 9.3절에서는 절단 연산자를 효율적으로 처리하는 방법에 대하여 설명한다.

9.1 논리곱 질의

논리곱 질의는 질의 포함된 모든 검색어들이 AND 연산자로 연결된 질의를 의미한다. 구글(Google) 등과 같은 다수의 웹 검색 시스템들은 검색창에 입력된 검색어들이 모두 포함되어 있는 문서들만을 검색한다. 즉, 연산자 없이 검색창에 입력된 검색어들을 논리곱 질의

로 간주하고 있다. 따라서, 이러한 시스템에서 논리곱 질의 처리의 효율성은 검색 서비스의 제공에 소비되는 비용을 좌우하는 매우 중요한 요소이다.

논리곱 질의의 처리 결과는 질의에 포함된 검색어들의 연산 순서와 관계없이 동일하다. 그러나, 논리곱 질의의 처리에 소비되는 메모리 양과 시간은 질의에 포함된 검색어들의 연산 순서에 의존적이며, 따라서 효율적인 논리곱 질의의 연산을 위해서는 논리곱 질의에 포함된 검색어들의 연산 순서를 결정하는 작업이 선행되어야 한다. 다음은 논리곱 질의를 효율적으로 연산하는 방법을 기술하고 있다.

단계 1: 렉시콘을 탐색하여 질의에 포함된 각각의 검색어 t 에 대한 문서 빈도 f_t 와 포스팅 p_t 의 위치를 구한 후, 검색어들을 문서 빈도 f_t 의 오름차순으로 정렬한다. 이후의 질의 처리에 있어서 문서 빈도가 작은 검색어의 포스팅이 우선적으로 처리된다.

단계 2: 가장 작은 문서 빈도 f_t를 갖는 검색어의 포스팅 p_t을 읽고, 이 포스팅에 포함된 문서 번호들을 검색 결과 후보들을 집합 C 에 포함시킨다. 검색 결과 후보들은 검색 결과에 포함될 가능성이 있는 문서 번호들로서, 최종 검색 결과에는 이 후보들 이외의 문서 번호들은 포함될 수 없으며, 일반적으로 검색어의 포스팅이 처리될 때마다 검색 결과 후보들의 수는 감소한다.

단계 3: 각각의 나머지 검색어 t에 대하여 다음을 수행한다.

(a) 검색어 t에 대한 포스팅 p_t를 읽는다.

(b) 검색 결과 후보 집합 C 에 포함된 각각의 문서 번호 d 가 포스팅 p_t 에 포함되어 있지 않으면, 검색 결과 후보 집합 C로부터 문서 번호 d를 제거한다.

(c) 검색 결과 후보 집합 C 가 공집합이 되면, 나머지 검색어에 대한 처리를 중단하며, 이때 질의를 만족하는 문서는 존재하지 않는다.

논리곱 질의의 처리에 있어서 문서 빈도가 작은 검색어를 우선적으로 처리함으로써 다음과 같은 2 가지 장점을 얻을 수 있다. 첫째, 질의를 처리하는 과정 중에 임시로 사용되는 메모리 양을 최소화할 수 있다. 위에서 설명된 논리곱 질의의 처리 방법을 살펴보면, 검색 결과 후보들의 수가 논리곱 질의에 포함된 검색어들의 처리 과정에서 증가하지 않음을 알 수 있다. 따라서 최초의 검색 결과 후보들을 가장 작은 문서 빈도를 갖는 검색어의 포스팅에 포함된 문서 번호들로 지정함으로써 검색 결과 후보들의 관리에 소비되는 메모리의 양을 최소화할 수 있다.

둘째, 문서 빈도의 오름차순으로 검색어를 처리함으로써 논리곱 질의의 처리 시간을 단축시킬 수 있다. 단계 3 의 (b)에서는 검색 결과 후보 집합과 검색어가 가리키는 포스팅에 포함된 문서 번호 집합과의 교집합을 생성한다. 교집합을 생성하는 방법으로서 병합(merge) 연산 또는 조회(look-up) 연산을 이용할 수 있다. 병합 연산에 소비되는 시간은 연산에 참여하는 2 개 집합의 크기의 합과 비례한다. 즉, $|C|$개의 검색 결과 후보들과 f_t개의 문서 번호를

포함하고 있는 포스팅의 병합을 위해서는 $|C| + f_t$ 번의 비교 연산이 수행된다. 예를 들어, $|C| = 60$ 이고 $f_t = 60{,}000$ 이라고 가정하면, 60,060 번의 비교 연산이 수행된다.

포스팅에 포함된 문서 번호들은 오름차순으로 정렬되어 있기 때문에, 교집합 생성을 위하여 병합 연산 대신에 포스팅에 포함된 문서 번호들을 이진 탐색하여 각각의 검색 결과 후보를 조회하는 방법을 사용할 수 있다. $|C| = 60$ 이고 $f_t = 60{,}000$ 이라고 가정하면, 각각의 검색 결과 후보에 대하여 $\log_2 60{,}000 \approx 16$ 번의 비교 연산이 수행되며, 따라서 조회 연산을 이용한 교집합 생성에는 총 $60 \times 16 = 960$ 번의 비교 연산이 수행된다. 일반적으로 검색 결과 후보 집합의 크기와 포스팅에 포함된 문서 번호 집합의 크기에 차이가 클 경우, 조회 연산을 이용한 교집합 생성이 보다 효율적이다.

한편, 문서 빈도가 큰 검색어의 우선적 처리와 비교하여 문서 빈도가 작은 검색어의 우선적 처리는 검색 결과 후보의 수를 보다 빠르게 감소시킨다. 심지어 어떠한 경우에는 논리곱 질의에 포함된 검색어들을 모두 처리하기도 전에 검색 결과 후보들의 수가 0 이 될 수 있으며, 이 경우 나머지 검색어의 처리는 불필요하며 질의를 만족하는 문서는 존재하지 않는다.

9.2 논리곱 정규형 질의

논리곱 정규형(conjuctive normal form) 질의는 다음과 같이 검색어들을 OR 연산자로 연결한 논리합 절을 다시 AND 연산자로 연결한 질의를 의미한다.

(정보 OR 데이터 OR 자료) AND
(검색 OR 서치) AND
(시스템 OR 엔진)

논리곱 정규형 질의는 논리곱 질의와 유사한 특성을 지니고 있기 때문에, 이들의 처리 방법도 논리곱 질의의 처리 방법과 유사하다. 즉, 논리곱 정규형 질의의 처리 결과는 질의에 포함된 논리합 절들의 처리 순서와 관계없이 동일하다. 그러나, 논리곱 정규형 질의의 처리에 소비되는 메모리 양과 시간은 질의에 포함된 논리합 절들의 처리 순서에 의존적이며, 따라서 효율적인 논리곱 정규형 질의의 연산을 위해서는 논리곱 정규형 질의에 포함된 논리합 절들의 처리 순서를 결정하는 작업이 선행되어야 한다.

논리합 절들의 처리 순서를 결정하기 위해서는 논리합 절들의 크기, 즉 이들의 처리 결과에 포함된 문서 번호들의 수를 알아야 한다. 논리합 절들의 크기는 모든 논리합 절들을 우선적으로 처리함으로써 정확히 계산될 수 있으며, 또한 논리합 절에 포함된 검색어들의 문서 빈도 합으로 추측될 수도 있다. 논리합 절들의 처리 순서가 결정된 이후의 과정은 검색어가 가리키는 포스팅 대신에 논리합 절의 처리 결과가 사용된다는 점을 제외하면 논리곱 질의의 처리 과정과 동일하다.

9.3 절단 연산

9.3.1 문자열 대조

문서들로부터 추출된 색인어들의 수가 많지 않으면, 전체 색인어들이 메모리에 상주되어 있는 메모리 기반 렉시콘을 사용할 수 있다. 이처럼 전체 색인어들이 메모리에 상주되어 있는 경우, 절단 연산자를 포함하고 있는 검색어와 렉시콘에 포함되어 있는 각각의 색인어에 대하여 효율적인 문자열 대조 알고리즘을 적용할 수 있다. 좌측 절단 또는 양쪽 절단 연산자를 포함하고 있는 검색어에 대해서는 렉시콘에 포함된 모든 색인어들에 대하여 문자열 대조 알고리즘을 적용해야 한다. 그러나, 렉시콘에 포함된 색인어들이 정렬되어 있고, 중간 절단 또는 우측 절단을 포함하고 있는 검색어에 대해서는 렉시콘에 포함된 일부 색인어들에 대해서만 문자열 대조 알고리즘을 적용할 수 있다. 즉, 렉시콘을 이진 탐색하여 검색어의 처음 문자열 부분과 일치하는 색인어들을 검출한 후, 이 색인어들에 대해서만 문자열 대조 알고리즘을 적용한다.

9.3.2 *n*-Gram 색인

이 방법은 렉시콘에 포함된 색인어에 문자열의 처음과 끝을 표시하기 위하여 기호 '$'를 추가하고, '$'가 추가된 문자열로부터 *n*-Gram 들을 생성한다. 즉, 색인어 "labor"로부터 6개의 2-Gram "$l", "la", "ab", "bo", "or", "r$"가 생성된다. 그리고, 이러한 *n*-Gram 들은 렉시콘에 포함된 색인어에 접근하기 위한 경로로서 사용된다. 예를 들면, <그림 9.1>의 (b)는 (a)에 주어진 렉시콘에 대하여 생성될 수 있는 2-Gram 색인의 일부를 보여준다. 일반적으로 (b)의 색인어 번호 리스트는 압축되어 저장될 수 있으며, 평균 약 6 비트로 1 개 색인어 번호를 저장할 수 있다 (Witten, Moffat, Bell, 1994)

렉시콘과 렉시콘에 대한 2-Gram 색인이 <그림 9.1>과 같을 때, 절단 연산자를 포함한 검색어 "lab*r"와 부합되는 색인어를 발견하기 위해서는 우선적으로 2-Gram 색인에 대해 다음의 질의가 수행되어야 한다.

$l AND la AND ab AND r$

위 질의의 수행 결과로서 렉시콘에 포함된 5 개의 색인어 "laaber", "labor", "laborator", "labour", "lavacaber"가 검색된다. 그러나, 검색된 5 개의 색인어 모두가 검색어 "lab*r"에 부합되지 않음에 유의해야 한다. 따라서, 검색된 색인어 모두를 검색어 "lab*r"와 대조하여 부합되지 않는 색인어를 제거하는 과정이 요구된다.

번호	색인어
1	Abhor
2	bear
3	laaber
4	labor
5	laborator
6	labour
7	lavacaber
8	slab

(a) 렉시콘

2-Gram	색인어 번호
$a	1
$b	2
$l	3, 4, 5, 6, 7
$s	8
aa	3
ab	1, 3, 4, 5, 6, 7, 8
bo	4, 5, 6
la	3, 4, 5, 6, 7, 8
or	1, 4, 5
ou	6
r$	1, 2, 3, 4, 5, 6, 7
ra	5
ry	5
sl	8

(b) 2-Gram 색인

<그림 9.1> 2-Gram 색인을 이용한 절단 연산자의 처리

9.3.3 회전 렉시콘

회전 렉시콘의 사용은 앞 절에서 설명된 색인어에 대한 2-Gram 색인보다 많은 저장 공간을 요구하나, 절단 연산에 소비되는 시간을 단축시킨다. 회전 렉시콘은 각 색인어의 문자열의 처음에 기호 '$'를 추가하고, '$'가 추가된 문자열을 시계 방향으로 회전시켜 이 문자열에 포함된 문자 수만큼의 회전 색인어를 생성한다. 즉, 색인어 "labor"에 대하여 6 개의 회전 색인어 "$labor", "labor$", "abor$l", "bor$la", "or$lab", "r$labo"들이 생성된다. <그림 9.2>의 (b)는 (a)에 주어진 렉시콘에 대하여 생성될 수 있는 회전 색인어들의 일부를 보여준다.

렉시콘에 포함된 모든 색인어들로부터 생성된 회전 색인어들이 정렬되어 있을 때, 중간 또는 우측 절단 연산자를 포함한 검색어와 부합되는 색인어를 발견하는 방법은 다음과 같다. 우선적으로 검색어 문자열의 처음에 기호 '$'를 추가하고, 절단 연산자가 문자열의 오른쪽 끝에 위치할 때까지 검색어를 회전시킨다. 즉, 절단 연산자를 포함한 검색어 "lab*r"는 문자열 "r$lab*"로 변환된다. 그리고, 절단 연산자를 제외한 나머지 문자열 "r$lab"과 처음 부분이 일치하는 회전 색인어들을 이진 탐색한다.

<그림 9.2>의 (b)에서 첫 번째 열은 단지 회전 색인어의 문자열 형태를 보여주기 위한 것이며, 실제의 회전 렉시콘은 <그림 9.2>의 (a)와 같은 렉시콘과 (b)의 두 번째 열과 같은 색인어 번호와 회전 횟수 쌍의 배열로 구성된다. 이는 색인어 번호와 회전 횟수 정보를 이용하여 회전 색인어를 생성할 수 있기 때문이다. 예를 들면, 15 번째 색인어 번호와 회전 정보 쌍 (7, 6)로부터 회전 색인어 "aber$lavac"는 다음과 같이 생성될 수 있다. 렉시콘으로부터

색인어 번호 7 에 해당하는 색인어 “lavacaber”을 얻은 후, 문자열의 처음에 기호 ‘$’를 추가한다. 그리고, 기호 ‘$’가 추가된 문자열 “$lavacaber”을 시계 방향으로 6 번 회전한다.

검색어의 처음에 절단 연산자를 포함하고 있는 검색어는 기호 ‘$’를 추가하지 않고, 절단 연산자가 우측에 위치할 때까지 문자열을 회전시킨다. 즉, 검색어 “*aber”는 “aber*”로 변환된다. 또한, 검색어의 처음과 끝 모두에 절단 연산자를 포함하고 있는 검색어도 기호 ‘$’를 추가하지 않고, 절단 연산자가 모두 우측에 위치할 때까지 문자열을 회전시킨다. 즉, 검색어 “*ab*”는 “ab**”로 변환된다. 그리고, 절단 연산자를 제외한 나머지 문자열과 처음 부분이 일치하는 회전 색인어들을 이진 탐색한다. 한편, 검색어 “ab*r*”와 같이 회전에 의해서 절단 연산자 모두를 우측에 위치시킬 수 없는 경우, 검색어와 부합하는 색인어의 발견은 2 단계로 이루어진다. 첫 번째 단계에서는 가장 긴 문자열 즉, “$ab”를 이용하여 검색어와 부합할 가능성이 있는 후보 색인어들을 선정하고, 두 번째 단계에서 후보 색인어 모두를 검색어 “ab*r*”와 대조하여 부합되지 않는 색인어를 제거한다.

번호	색인어
1	Abhor
2	bear
3	laaber
4	labor
5	laborator
6	labour
7	lavacaber
8	slab

(a) 렉시콘

회전 색인어의 문자열 형태	색인어 번호와 회전 횟수
$abhor	(1, 0)
$bear	(2, 0)
$laaber	(3, 0)
$labor	(4, 0)
$laborator	(5, 0)
$labour	(6, 0)
$lavacaber	(7, 0)
$slab	(8, 0)
aaber$l	(3, 2)
ab$sl	(8, 3)
aber$la	(3, 3)
aber$lavac	(7, 6)
abhor$	(1, 1)
abor$l	(4, 2)
aborator$l	(5, 2)
abour$l	(6, 2)
r$abho	(1, 5)
r$bea	(2, 4)
r$laabe	(3, 6)
r$labo	(4, 5)
r$laborato	(5, 9)
r$labou	(6, 6)
r$lavacabe	(7, 9)
slab$	(8, 1)

(b) 회전 색인어

<그림 9.2> 회전 렉시콘을 이용한 절단 연산자의 처리

제 10 장 벡터 공간 모델

10.1 유사도 계산

벡터 공간 모델(Salton, 1983)은 문서 또는 질의의 내용을 표현하는 색인어들을 추출하고 문서 또는 질의에서의 중요도에 따라 추출된 색인어들에 가중치를 부여함으로써, 문서와 질의를 가중치가 부여된 색인어들의 벡터로 표현한다. 즉, 벡터 공간 모델에서 문서 D 와 질의 Q 는 다음과 같은 벡터 형태로 표현된다.

$$D = \{(t_1, w_{d1}), (t_2, w_{d2}), \ldots , (t_n, w_{dn})\}$$
$$Q = \{(t_1, w_{q1}), (t_2, w_{q2}), \ldots , (t_n, w_{qn})\}$$

문서 색인어 가중치 w_{di}는 문서 D에서 i번째 색인어 t_i의 가중치를 의미하고, 질의 색인어 가중치 w_{qi}는 질의 Q에서 i번째 색인어 t_i의 가중치를 의미한다. 문서 또는 질의에 나타나지 않는 색인어들에 대해 가중치 값 0이 할당된다.

불리안 검색이 질의를 만족하는 문서들의 집합을 생성하는데 비하여, 벡터 공간 모델에 의한 검색은 질의와 문서 사이의 유사도를 계산하고, 유사도 값에 따라 내림차순으로 문서들을 정렬하여 일정 수의 상위 문서들을 사용자에게 제공한다. 벡터 공간 모델에서 문서 D와 질의 Q의 유사도는 다음과 같이 문서 벡터와 질의 벡터의 내적으로 표현된다.

$$Sim(D,Q) = \sum_{i=1}^{n} (w_{di} \times w_{qi})$$

위의 식에서 알 수 있듯이 문서와 질의 사이의 유사도 값은 문서 및 질의 색인어 가중치에 의해 결정되기 때문에, 색인어 가중치를 산출하는 기법은 벡터 공간 모델을 사용하는 정보 검색 시스템의 검색 효과에 영향을 미치는 매우 중요한 요소이다.

10.2 색인어 가중치 산출

문서 색인어 가중치 w_{di}와 질의 색인어 가중치 w_{qi}를 산출하기 위해서 다음과 같은 세 가지 구성 요소가 사용된다. 첫째, 출현 빈도 요소는 문서 또는 질의 내에서 자주 출현하는 색인어에 보다 높은 가중치를 부여한다. 둘째, 장서 빈도 요소는 전체 문서들 중에서 적은 수의 문서에 출현하는 색인어에 보다 높은 가중치를 부여한다. 마지막으로 정규화 요소는 데이터베이스 내의 모든 문서 벡터들의 길이를 일치시키는 요소로서, 작은 크기의 문서들이 유사도 계산에 있어서 불공정하게 취급되는 것을 피하도록 한다.

<표 10.1>은 벡터 공간 모델에서 색인어 가중치 산출을 위해 널리 사용된 수식을 구성 요소별로 보여준다. 문서와 질의로부터 추출된 색인어들의 가중치를 산출하는 식은 앞에서 언급된 세가지 구성 요소의 조합으로 생성된다. 예를 들어, *lnc · ltc* 기법은 문서 색인어 가중치 산출에는 *lnc* 기법을 적용하고, 질의 색인어 가중치 산출에는 *ltc* 기법을 적용함을 의미한다. 즉, 문서 색인어 가중치는 색인어 출현 빈도의 로그 값을 코사인 정규화함으로써 계산되고, 질의 색인어 가중치는 색인어 출현 빈도와 역 문헌 빈도를 곱한 값을 코사인 정규화함으로써 계산되며, 이들을 식으로 표현하면 다음과 같다.

• lnc (문서)

$$w_{di} = \frac{\log(tf_{di}) + 1.0}{\sqrt{\sum_{i=1}^{n} [\log(tf_{di}) + 1.0]^2}}$$

• ltc (질의)

$$w_{qi} = \frac{\left(\log(tf_{qi}) + 1.0\right) \cdot \log\left(\frac{N}{n_i}\right)}{\sqrt{\sum_{i=1}^{n} \left[\left(\log(tf_{qi}) + 1.0\right) \cdot \log\left(\frac{N}{n_i}\right)\right]^2}}$$

<표 10.1> 색인어 가중치 부여 기법의 구성 요소

출현 빈도(term frequency)		
b	1.0	색인어의 출현 빈도를 무시하고 벡터를 구성하는 색인어에 1 의 가중치 부여
n	*tf*	문서나 질의 내에서 색인어의 출현빈도
a	$0.5 + 0.5\frac{tf}{max\ tf}$	보강된 정규화 출현 빈도 (*tf* 를 *max tf* 로 나누고, 그 결과가 0.5~1.0 의 값을 갖도록 정규화)
l	$\ln tf + 1.0$	색인어의 출현 빈도에 로그 함수 적용
장서 빈도(collection frequency)		
n	1.0	색인어 출현 빈도만으로 가중치 생성
t	$\ln\frac{N}{n}$	색인어의 출현 빈도와 역문서 빈도를 곱한다. (*N* 은 전체 문서들의 수이며, *n* 은 그 색인어를 포함하고 있는 문서들의 수이다.)
정규화(normalization)		
n	1.0	출현 빈도와 장서 빈도만으로 유도된 가중치를 사용
c	$\frac{1}{\sqrt{\sum_{vector} w_i^2}}$	유클리디안 벡터 길이를 이용한 코사인 정규화

10.3 색인어 가중치 산출 기법 분석

10.3.1 문서 형태 분류

출현 빈도 벡터 길이(*tf*-vector length)는 출현 빈도의 합으로 정의된다. t_i가 색인어의 출현 빈도이고, n이 벡터를 구성하는 색인어들의 수일 때, 출현 빈도 벡터 길이는 $\sum_{i=1}^{n} tf_i$ 이다. 예를 들면, 문서 d_1이 색인어와 가중치의 쌍으로 다음과 같이 표현되었다고 가정하자.

$$d_1=\{(t_1,1), (t_2,2), (t_3,3), (t_4,4), (t_5,5)\}$$

이때 문서 d_1의 출현 빈도 벡터 길이는 15(=1+ 2+ 3+ 4+ 5)이다. 많은 현실적인 문서 집합에서 문서들은 매우 다양한 출현 빈도 벡터 길이를 갖는다. 문서들은 출현 빈도 벡터 길이에 따라 다음과 같이 3가지 형태로 분류될 수 있다.

- 짧은 출현 빈도 벡터 길이 (short *tf*-vector length)
- 중간 출현 빈도 벡터 길이 (median *tf*-vector length)
- 긴 출현 빈도 벡터 길이 (long *tf*-vector length)

한편, 많은 경우에 긴 문서뿐만 아니라 짧은 문서들도 단일 주제가 아닌 다중 주제를 기술하는 것으로 알려져 왔다. 따라서 문서들은 문서가 다루는 주제들의 수에 따라 다음과 같이 2가지 형태로 분류될 수 있다.

- 단일 주제 (single topic)
- 다중 주제 (multiple topic)

10.3.2 색인어 가중치 산출 기법의 특성

출현 빈도 벡터 길이를 정규화하지 않는 가중치 기법이 사용되었을 때, 긴 출현 빈도 벡터 길이의 문서는 짧은 출현 빈도 벡터 길이를 갖는 문서에 비하여 높은 검색 가능성을 갖는다. 예를 들면, 문서 d_2, d_3가 다음과 같이 표현되어 있다고 가정하자.

$$d_2=\{(t_1,1), (t_2,1), (t_3,1), \ldots, (t_n,1)\}$$
$$d_3=\{(t_1,2), (t_2,2), (t_3,2), \ldots, (t_n,2)\}$$

출현 빈도 벡터 길이에 대한 정규화 요소를 포함하지 않는 가중치 기법 *lnn*(ln *tf*+1.0)이 사용된다면, $d_{3.lnn}$의 색인어 가중치는 $d_{2.lnn}$의 색인어 가중치보다 높다.

$$d_{2.lnn}=\{(t_1,1), (t_2,1), (t_3,1), \ldots, (t_n,1)\}$$
$$d_{3.lnn}=\{(t_1,1.69), (t_2, 1.69), (t_3, 1.69), \ldots, (t_n, 1.69)\}$$

따라서 질의 q_1={(t_1,w_1), (t_2,w_2), (t_3,w_3), …, (t_n,w_n)}에 대한 $d_{2.lnn}$, $d_{3.lnn}$의 유사도는 다음과 같이 계산될 수 있다.

$$Sim(d_{2.lnn}, q_1) = w_1 + w_2 + ... + w_n = \sum_{i=1}^{n} w_i$$

$$Sim(d_{3.lnn}, q_1) = 1.69 \cdot w_1 + 1.69 \cdot w_2 + ... + 1.69 \cdot w_n = 1.69 \cdot \sum_{i=1}^{n} w_i$$

즉, 긴 출현 빈도 벡터 길이를 갖는 문서 d_3의 유사도가 짧은 출현 빈도 벡터 길이를 갖는 문서 d_2 유사도의 1.69배이다. 그러나 출현 빈도 벡터 길이를 고려할 때, 문서 d_2와 d_3는 거의 동일한 문서로 간주될 수 있다.

일반적으로 검색의 목적하에서 모든 문서는 동등하게 취급되어야 한다. 그러나 앞에서 설명된 바와 같이 출현 빈도 벡터 길이에 대한 정규화가 색인어 가중치 산출 기법에 고려되지 않는다면, 검색 시스템은 긴 출현 빈도 벡터 길이를 갖는 문서를 선호한다. 벡터 공간 모델의 색인어 가중치 산출 기법에서 사용되는 코사인 정규화는 출현 빈도 벡터 길이를 정규화하는 특성을 지니고 있다. 예를 들어, 코사인 정규화 요소를 포함하고 있는 *lnc* 기법을 사용하여 색인어 가중치를 산출하면, 문서 d_2, d_3는 다음과 같이 동일한 벡터로서 표현된다.

$$d_{1.lnc} = \left\{\left(t_1, \frac{1}{\sqrt{n}}\right), \left((t_2, \frac{1}{\sqrt{n}}\right), \ldots, \left(t_n, \frac{1}{\sqrt{n}}\right)\right\}$$

$$d_{2.lnc} = \left\{\left(t_1, \frac{1}{\sqrt{n}}\right), \left((t_2, \frac{1}{\sqrt{n}}\right), \ldots, \left(t_n, \frac{1}{\sqrt{n}}\right)\right\}$$

코사인 정규화 요소를 포함하는 색인어 가중치 산출 기법은 많은 경우에 코사인 정규화 요소를 포함하지 않는 색인어 가중치 산출 기법보다 높은 검색 효과를 제공하는 것으로 알려져 있다. 그러나, 코사인 정규화는 다중 주제를 다루는 적합 문서의 검색을 어렵게 하는 특성을 지니고 있다. 이는 사용자 질의와 관련된 적합 색인어의 가중치가 비적합 색인어 가중치에 의해 감소되기 때문이다. 예를 들면, 문서 d_4가 단일 주제만을 다루고, 문서 d_5는 문서 d_4가 기술하는 주제를 포함하여 여러 개의 주제를 다룬다고 가정하자. 즉 문서 d_4, d_5는 다음과 같이 표현될 수 있다.

d_4={(t_1,1), …, (t_m,1), (t_{m+1},0), …, (t_n,0)}

d_5={(t_1,1), …, (t_m,1), (t_{m+1},1), …, (t_n,1)}

색인어 가중치 산출 기법 *lnc*가 적용된다면, 문서 d_4, d_5는 다음과 같이 변환된다.

$$d_{4.lnc} = \left\{ \left(t_1, \frac{1}{\sqrt{m}} \right), \ldots, \left(t_m, \frac{1}{\sqrt{m}} \right), (t_{m+1}, 0), \ldots, (t_n, 0) \right\}$$

$$d_{5.lnc} = \left\{ \left(t_1, \frac{1}{\sqrt{n}} \right), \ldots, \left(t_m, \frac{1}{\sqrt{n}} \right), \left(t_{m+1}, \frac{1}{\sqrt{n}} \right), \ldots \left(t_n, \frac{1}{\sqrt{n}} \right) \right\}$$

따라서 질의 q_2={(t_1,w_1), …, (t_m,w_m), (t_{m+1},0), …, (t_n,0)}에 대한 $d_{4.lnc}$, $d_{5.lnc}$의 유사도는 다음과 같다.

$$Sim(d_{4.lnc}, q_2) = \frac{1}{\sqrt{m}} \bullet w_1 + \frac{1}{\sqrt{m}} \bullet w_2 + \ldots + \frac{1}{\sqrt{m}} \bullet w_m = \frac{1}{\sqrt{m}} \bullet \sum_{i=1}^{m} w_i$$

$$Sim(d_{5.lnc}, q_2) = \frac{1}{\sqrt{n}} \bullet w_1 + \frac{1}{\sqrt{n}} \bullet w_2 + \ldots + \frac{1}{\sqrt{n}} \bullet w_m = \frac{1}{\sqrt{n}} \bullet \sum_{i=1}^{m} w_i$$

n이 항상 m보다 크기 때문에, 단일 주제 문서 d_4가 다중 주제 문서 d_5 보다 높은 순위를 부여 받는다. 그러나, 문서 d_4, d_5는 질의 q_2에 대하여 같은 양의 정보를 포함하고 있다. 이처럼 바람직하지 않은 결과가 생성되는 이유는 문서 d_5에 포함되어 있는 적합 색인어 t_1-t_m의 가중치가 비적합 색인어 t_{m+1}-t_n의 가중치에 의해 감소되기 때문이다.

*a*로 표기되는 보강된 정규화 출현 빈도는 출현 빈도 *tf*를 최대 출현 빈도 max*tf*로 정규화한다. 이러한 최대 정규화는 특정 경우에 출현 빈도 벡터 길이를 정규화할 수 있다. 예를 들면, 문서 d_6, d_7이 다음과 같이 표현되어 있다고 가정하자.

$$d_6 = \{(t_1,1), (t_2,1), \ldots, (t_n,1)\}$$
$$d_7 = \{(t_1,2), (t_2,2), \ldots, (t_n,2)\}$$

색인어 가중치 산출 기법 *ann*(0.5+0.5•*tf*/max*tf*)이 적용된다면, 문서 d_6, d_7은 다음과 같이 동일한 벡터로 표현된다.

$$d_{6.ann} = \{(t_1,1), (t_2,1), \ldots, (t_n,1)\}$$
$$d_{7.ann} = \{(t_1,1), (t_2,1), \ldots, (t_n,1)\}$$

그러나, 이러한 최대 정규화는 많은 경우에 출현 빈도 벡터 길이를 정규화할 수 없다. 예를 들면, 문서 d_8, d_9가 다음과 같이 표현되어 있다고 가정하자.

$$d_8 = \{(t_1,1), (t_2,1), \ldots, (t_{100},1)\}$$
$$d_9 = \{(t_1,2), (t_2,1), \ldots, (t_{100},1)\}$$

색인어 가중치 산출 기법 *ann*이 적용된다면, 다음과 같이 $d_{8.ann}$의 색인어 가중치가 $d_{9.ann}$의 색인어 가중치보다 크다.

$$d_{8.ann} = \{(t_1,1), (t_2,1), \ldots, (t_n,1)\}$$
$$d_{9.ann} = \{(t_1,1), (t_2,0.75), \ldots, (t_n,0.75)\}$$

따라서 임의의 질의에 대하여 문서 d_8이 문서 d_9 보다 높은 순위를 부여 받을 가능성이 크다. 그러나, 색인어 t_1의 출현 빈도만이 서로 다르기 때문에, 문서 d_8, d_9는 거의 동일한 문서로 간주될 수 있다. 한편, 색인어 가중치 산출 기법 *lnc*가 적용된다면, 문서 d_8, d_9는 다음과 같이 변환될 수 있다.

$$d_{8.lnc}=\{(t_1,0.1), (t_2,0.1), \ldots, (t_n,0.1)\}$$
$$d_{9.lnc}=\{(t_1,0.17), (t_2,0.99), \ldots, (t_n,0.99)\}$$

색인어 t_1의 가중치를 제외한다면 $d_{8.lnc}$의 색인어 가중치는 $d_{9.lnc}$의 색인어 가중치와 거의 동일하기 때문에, 문서 d_8과 d_9는 임의의 질의에 대하여 유사한 수준의 유사도를 제공한다.

최대 정규화는 출현 빈도 벡터 길이의 정규화에 있어 문제점을 지니고 있다 할지라도, 다중 주제 문서의 검색이라는 측면에서 코사인 정규화에 비하여 장점을 지니고 있다. 예를 들면, 문서 d_{10}이 단일 주제를 다루고, 문서 d_{11}이 문서 d_{10}이 다루는 단일 주제를 포함한 여러 개의 주제를 기술한다고 가정하자. 즉, 문서 d_{10}, d_{11}은 다음과 같이 표현될 수 있다.

$$d_{10}=\{(t_1,1), \ldots, (t_m,1), (t_{m+1},0), \ldots, (t_n,0)\}$$
$$d_{11}=\{(t_1,1), \ldots, (t_m,1), (t_{m+1},1), \ldots, (t_n,1)\}$$

색인어 가중치 산출 기법 *ann*이 적용된다면, 문서 d_{10}, d_{11}은 다음과 같이 변환된다.

$$d_{10.ann}=\{(t_1,1), \ldots, (t_m,1), (t_{m+1},0), \ldots, (t_n,0)\}$$
$$d_{11.ann}=\{(t_1,1), \ldots, (t_m,1), (t_{m+1},1), \ldots, (t_n,1)\}$$

따라서 질의 $q_3=\{(t_1,w_1), \ldots, (t_m,w_m), (t_{m+1},0), \ldots, (t_n,0)\}$에 대하여, 문서 d_{10}, d_{11}은 다음과 같이 동일한 유사도를 부여받는다.

$$Sim(d_{10.ann}, q_3) = Sim(d_{11.ann}, q_3) = \sum_{i=1}^{m} w_i$$

비록 일반적이지 않을지라도, 위의 예는 코사인 정규화의 문제점이 최대 정규화에 의해 완화될 수 있음을 보여준다.

지금까지 검색되는 문서 형태에 영향을 주는 코사인 정규화, 최대 정규화와 같은 색인어 가중치 산출 기법의 중요한 특성을 설명하였다. 이러한 특성에 근거하여 색인어 가중치 산출 기법은 다음과 같이 분류될 수 있다.

- 부류 C: 코사인 정규화를 수행하는 가중치 기법들
- 부류 M: 최대 정규화를 수행하고 코사인 정규화를 수행하지 않는 가중치 기법들
- 부류 N: 코사인 정규화와 최대 정규화 모두를 수행하지 않는 가중치 기법들

앞에서 설명된 바와 같이, 서로 다른 부류에 속하는 색인어 가중치 산출 기법들은 서로 다른 형태의 문서들을 검색할 수 있다. 이를 확인하기 위하여 서로 다른 색인어 가중치 산출 기법을 사용하는 두개의 검색 실행에 의해 공통적으로 검색되는 문서들의 수를 조사하였다. <표 10.2>는 같은 부류에 속하는 색인어 가중치 산출 기법들에 의해 보다 많은 수의 문서들이 공통적으로 검색됨을 보여준다. 또한 부류 N, 부류 M에 의해 공통적으로 검색되는 문서 수가 부류 C, 부류 N에 의해 공통적으로 검색되는 문서 수와 부류 C, 부류 M에 의해 공통적으로 검색되는 문서 수보다 많으며, 따라서 코사인 정규화의 사용 여부가 서로 다른 집합의 문서들을 검색하는데 중요한 역할을 함을 알 수 있다.

<표 10.2> 서로 다른 색인어 가중치 산출 기법을 사용하는 두개의 검색 실행에 의해 공통적으로 검색되는 문서들의 수 (WSJ.D2: 상위 순위 200개 문서가 100개의 질의에 의해 검색됨)

	lnc•ltc(C)	*anc•ltc*(C)	*lnn•ntc*(N)	*ltn•ntc*(N)	*ann•ntc*(M)
anc•ltc(C)	**15183**	—	—	—	—
lnn•ntc(N)	9911	8200	—	—	—
ltn•ntc(N)	10573	9032	**15937**	—	—
ann•ntc(M)	10106	10395	13069	13310	—
atn•ntc(M)	10301	10301	11745	13733	**16069**

[참고문헌]

① Lee, J.H. (1995), Combining Multiple Evidence from Different Properties of Weighting Schemes, ACM SIGIR Conference on Research and Development in Information Retrieval, 180-188.

② Harman, D. (1993), Overview of the 1st text retrieval conference, Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 36-48.

③ Salton, G. & McGill, M.J. (1983), Introduction to Modern Information Retrieval, McGraw-Hill, Inc..

④ Singhal, C. & Buckley, C. & Mitra, M. (1996), Pivoted Document Length Normalization, ACM SIGIR Conference, 21-29.

제 11 장 적합성 피드백

일반적으로 사용자들은 광범위한 정보 요구를 지니고 정보 검색 또는 정보 필터링 시스템에 접근하여 그 요구들을 질의로서 표현한다. 이러한 초기의 질의는 매우 임의적이며, 때때로 사용자들은 그들이 지니고 있는 문제점조차도 정확하게 표현할 줄 모른다. Belkin(1982)은 사용자들의 이러한 상태를 비정상적인 지식의 상태라고 불렀고, Bookstein(1983)은 불확실성은 정보 검색 과정의 본질적인 것이라고 말하였다. 이처럼 질의가 부정확할 경우, 정보 검색 및 정보 필터링 시스템으로부터 양질의 결과를 얻는 것은 매우 어려운 일이며, 따라서 불완전한 초기 질의는 보완되어야 한다.

적합성 피드백은 보다 많은 적합 문서 그리고 보다 적은 비적합 문서를 검색할 수 있도록 불완전한 초기 질의를 보완하여 질의를 자동으로 재생성하는 기법이다. 지금까지 많은 정보 학자들에 의해 다양한 적합성 피드백 방법들이 검증되어 왔으며, 각 방법의 장단점이 논의되어 왔다. 적합성 피드백에 대부분의 연구들은 질의를 가중치가 부여된 검색어들의 벡터로서 표현하는 벡터 질의 환경에서 수행되었으며, 이러한 연구들은 새로운 질의의 재구성 과정에서 적합 문서들에 출현한 검색어들의 가중치를 높이고, 부적합 문서들에 출현한 검색어들의 가중치를 낮춘다는 원칙을 기초로 하고 있다.

사용자의 정보 요구를 주제어와 그들간의 논리적인 관계로서 표현하는 불리안 검색 모델을 기반으로 하는 기존의 불리안 정보 검색 시스템은 구현이 용이하고 짧은 검색 시간을 제공하기 때문에, 지금까지 정보 검색 분야에서 널리 사용되고 있다. 그러나 정보 검색 분야에서 수행된 대부분의 연구는 불리안 검색 모델의 한계성을 극복하기 위해 질의를 가중치가 부여된 검색어들의 벡터로서 표현하는 벡터 질의 환경에서 수행되었으며, 따라서 적합성 피드백에 관한 대부분의 연구들도 불리안 질의가 아닌 벡터 질의 수정을 목표로 수행되었다. 지금까지 불리안 질의의 수정을 위한 적합성 피드백에 관한 연구는 매우 적었으며, 정보 검색 또는 정보 필터링에 있어서 불리안 모델의 중요도를 고려할 때 앞으로도 매우 적을 것으로 예상된다. [Dill80, Dill83, Salt84, Salt85]은 불리안 질의 수정을 위한 적합성 피드백과 관련된 연구 논문들이다.

벡터 질의 환경에서 개발된 적합성 피드백 방법들은 질의 벡터를 수정하는 방법들과 확률 검색에 근거한 방법들로 구분될 수 있다. 이 두 가지 방법들은 새로운 질의를 생성하기 위한 초기 질의의 사용 유무에 차이점을 두고 있다. 즉, 질의 벡터를 수정하는 방법들은 초기 질의를 수정하는 형태를 취하고 있으며, 확률 검색에 근거한 방법들은 초기 질의와 무관하게 적합성 정보를 기반으로 새로운 질의를 생성한다. 확률 검색에 근거한 적합성 피드백을 이해하기 위해서는 확률 검색 모델에 대한 이해가 선행되어야 하기 때문에, 확률 검색에 근거한 적합성 피드백은 14 장에서 확률 검색 모델에 대한 기술에 이어서 설명되며, 본 장에

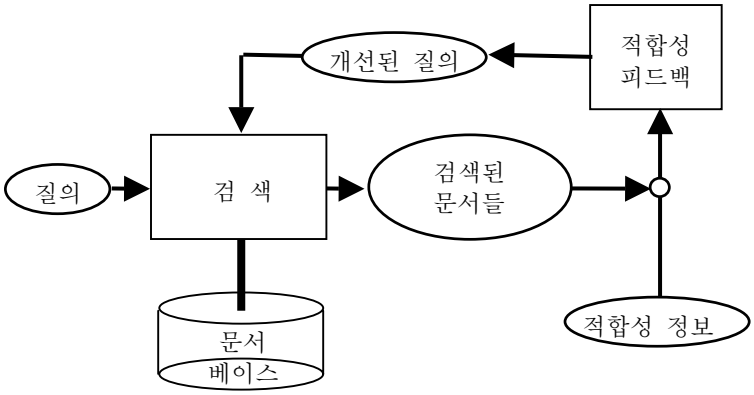
서는 적합성 피드백의 활용 분야와 질의 벡터를 수정하는 적합성 피드백들에 대하여 기술한다.

11.1 활용 분야

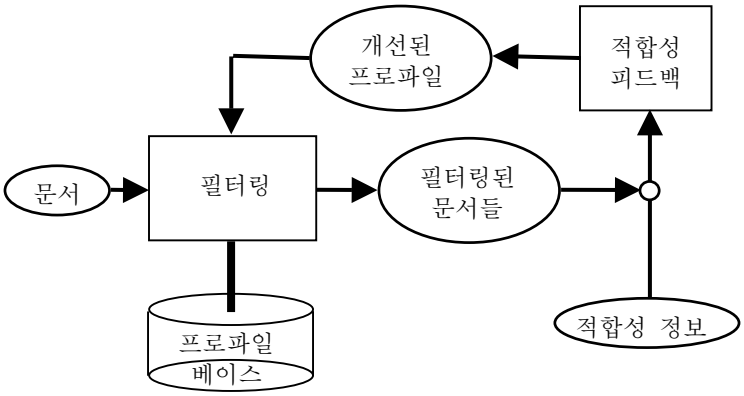
지난 5 년여 동안 국내에서는 정보 검색 시스템에 대한 관심이 지속적으로 증가되어 왔으며, 많은 상용 정보 검색 시스템들이 국내 기술로 개발되어 다양한 분야에서 활용되고 있다. 이와는 대조적으로 국내의 정보 필터링 시스템에 대한 수요는 정보검색 시스템에 비하여 상대적으로 적은 편이며, 따라서 이에 대한 연구 및 개발도 매우 부족한 실정이다. 그러나 정보화 사회의 발전과 발맞추어 정보 필터링 시스템에 대한 수요는 지속적으로 증가할 것이 예상된다.

<그림 11.1>과 <그림 11.2>는 각각 정보 검색 시스템과 정보 필터링 시스템의 구성도를 보여준다. 정보 검색 시스템[Salt83]은 축적된 문서들의 집합으로부터 사용자의 정보 요구를 만족하는 문서들을 검색하여 사용자에게 전달하고, 정보 필터링 시스템[Belk92, Bell96, Call96]은 사용자의 정보 요구를 질의 DB 에 수록해 두고, 새로운 문서가 들어오게 되면 질의 DB 에 저장된 질의와 비교하여 질의를 만족하는 문서를 사용자에게 전달한다. 이때 사용자는 생각하고 있던 정보 요구에 근거하여 전달된 문서들을 적합 또는 비적합 문서로 구분할 수 있으며, 이러한 적합 또는 비적합 문서들의 집합을 주어진 정보 요구에 대한 적합성 정보라고 한다.

적합성 피드백은 사용자들에 의해 생성된 적합성 정보를 활용하여, 불완전한 초기 질의를 보완할 수 있는 질의를 자동으로 생성한다. 즉, 적합성 피드백은 정보 검색 시스템 또는 정보 필터링 시스템에서 사용자가 원하는 문서들을 검색하기 위한 질의의 작성을 도와준다. 그러나 적합성 피드백에 관한 연구가 1960 년대부터 30 년 이상 수행되어 왔음에도 불구하고, 아직까지 상용적으로 활발히 활용되지 못하고 있는 실정이다. 그러나 최근 들어 적합성 피드백에 대한 효용성이 입증되고 있으며, 이를 이용하는 상용 정보 검색 또는 정보 필터링 시스템이 등장하고 있는 추세이다.



<그림 11.1> 정보 검색 시스템 구조



<그림 11.2> 정보 필터링 시스템 구조

11.2 질의 벡터 수정

Maron & Kuhns(1960)는 1960 년에 발표된 논문에서 사용자 질의와 밀접하게 관련된 검색어를 질의에 추가하는 질의 수정을 수행함으로써 보다 많은 적합 문서들을 검색할 수 있음을 언급하였다. 적합성 피드백의 초기 연구는 이러한 이론을 기반으로 벡터 공간 모델에서 시작되었다. 정보 검색 시스템 SMART 는 벡터 공간 모델을 기반으로 구현되었으며, 1960 년대부터 코넬 대학에서 개발되어 왔다. 초기의 SMART 시스템에서 적합성 피드백을 주제로 하는 많은 연구가 수행되었다[Salt71, Ide71b]. 이러한 연구의 결과로서 Rocchio 는 질의 벡터 수정에 관한 실험 결과들을 발표하였으며[Rocc66, Rocc71], 이 방법은 과거 많은 연구 결과들을 통해 검색 효과의 개선에 있어서 긍정적인 평가를 받아왔다. 한편, Rocchio 방법 이외의 질의 벡터 수정 방법은 Ide 에 의해 연구되었다[Ide71a].

질의 벡터를 수정하는 적합성 피드백 방법들은 질의를 확장하고 검색어 가중치를 재산정하는 2 개의 과정으로 구성되어 있다. 질의 확장은 적합 문서에 출현할 가능성이 높은 검색어들을 질의 벡터에 추가하는 과정이며, 검색어 가중치 재산정 과정은 질의 확장 과정 이후에 질의 벡터에 포함된 검색어들의 가중치를 재산정하는 과정이다. 다음에서는 이러한 2 개의 과정에 대하여 설명한다.

11.2.1 질의 확장

질의를 확장하는 방법으로는 다음과 같은 4 가지 방법이 사용되어 왔다. 첫째, 질의를 확장하지 않는 방법으로, 질의 벡터 수정의 결과로서 초기 질의에 포함된 검색어들의 가중치만이 재산정된다. 둘째, 적합 문서에 출현하는 모든 색인어들을 검색어로서 질의에 추가하는 방법이다. 셋째, 적합 문서에 출현하고, 적합 문서 내에서 문서 빈도가 높은 색인어들을 검색어로서 질의에 추가하는 방법이다. 마지막으로, 적합 문서들의 합 벡터에서 가중치가 높은 색인어들을 검색어로서 질의에 추가하는 방법이다.

예를 들어, 초기 질의 “정보”에 의해 검색된 적합 문서 집합에 총 3 개의 색인어 ‘정보’, ‘검색’, ‘시스템’이 존재하고, 적합 문서 빈도는 ‘정보’, ‘시스템’, ‘검색’의 순서로, 적합 문서 합 벡터에서 가중치는 ‘정보’, ‘검색’, ‘시스템’의 순서로 높게 나타난다고 가정하자. 이때 질의를 확장하는 각각의 방법들은 다음과 같이 초기 질의에 검색어를 추가한다.

- 적합 문서에 출현하는 모든 색인어 추가: 초기 질의에 ‘검색’과 ‘시스템’을 추가하므로, 확장된 질의는 3 개의 검색어 ‘정보’, ‘검색’, ‘시스템’을 포함한다.
- 적합 문서 빈도가 높은 색인어 추가: 1 개의 검색어만을 추가한다고 가정하면, 초기 질의에 ‘시스템’을 추가하므로, 확장된 질의는 2 개의 검색어 ‘정보’, ‘시스템’을 포함한다.

- 적합 문서 합 벡터에서 가중치가 높은 색인어 추가: 1 개의 검색어만을 추가한다고 가정하면, 초기 질의에 '검색'을 추가하므로, 확장된 질의는 2 개의 검색어 '정보', '검색'을 포함한다.

대용량 정보 검색 시스템에서 시스템 응답 시간은 검색 대상 문서들의 수 보다 질의에 포함된 검색어들의 수에 의해 많은 영향을 받는다. 따라서 질의에 추가된 검색어의 수는 시스템 성능을 좌우하는 중요한 요소가 될 수 있다. Cranfield 테스트 컬렉션을 사용한 Harman(1988)의 실험에서 검색된 적합 문서 내에 출현하는 모든 색인어를 사용하는 것 보다 선택적으로 사용하는 방법이 더욱 향상된 검색 결과를 보였다. 즉, 검색된 적합 문서들에 출현하는 색인어들을 다양한 방법으로 정렬하여 상위 20 개의 색인어를 질의 확장에 사용했을 경우 가장 우수한 검색 효과를 제공하였다.

한편, 6 가지 테스트 컬렉션을 사용한 Salton & Buckley(1990)의 실험에서는 대부분의 테스트 컬렉션에서 적합 문서에 출현하는 모든 색인어들을 추가하는 방법이 색인어를 선택적으로 추가하는 방법 보다 우수한 검색 효과를 제공하였다. 그러나, 이러한 2 가지 질의 확장 방법이 제공하는 검색 효과의 차이는 매우 근소하였다. 또한, Salton & Buckley 는 적합성 피드백에 의한 검색 효과 개선을 결정하는 요소로서 초기 질의에 포함된 검색어 수, 초기 질의가 제공하는 검색 효과, 검색 대상 문서 집합의 특성을 언급하였다. 즉, 초기 질의에 포함된 검색어의 수가 적고, 초기 질의가 제공하는 검색 효과가 낮을수록, 그리고 광범위한 내용의 문서 집합보다 전문적이고 기술적인 내용의 문서 집합의 검색에서 높은 검색 효과 개선을 얻을 수 있었다.

11.2.2 검색어 가중치 재산정

벡터 공간 모델에서 적합성 피드백은 질의 확장 과정 이후에 질의에 포함된 검색어의 가중치를 재산정한다. 즉, 가중치 재산정 이전의 질의 $q=(w_1, w_2, ..., w_n)$에 대하여, 적합 문서에 빈번하게 출현하는 검색어에 대한 가중치를 증가시키고, 비적합 문서에 빈번하게 출현하는 검색어에 대한 가중치를 감소시킴으로써 다음과 같은 새로운 질의 벡터를 생성한다.

$$q' = (w_1', w_2', ..., w_n')$$

여기에서 w_i'는 검색어 t_i의 변화된 검색어 가중치를 나타낸다.

Rocchio 는 다음과 같은 식을 이용하여 질의 벡터에 포함된 검색어들의 가중치를 재산정하였다[Rocc66, Rocc71].

$$Q_{new} = Q_{old} + \frac{1}{n_{rel}} \sum_{r=1}^{n_{rel}} D_r - \frac{1}{n_{nonrel}} \sum_{n=1}^{n_{nonrel}} D_n$$

Q_{new}	수정 후 질의 벡터
Q_{old}	수정 전 질의 벡터
D_r	적합 문서 d_r의 벡터
D_n	비적합 문서 d_n의 벡터
n_{rel}	적합 문서의 수
n_{nonrel}	비적합 문서의 수

위 식에서 수정 후 질의 벡터 Q_{new}는 수정 전 질의 벡터 Q_{old}에 적합 문서의 평균 벡터와 비적합 문서의 평균 벡터의 차이를 더한 값이다. 즉, 적합 문서에 포함된 검색어의 경우에는 가중치를 증가시키고, 비적합 문서에 포함된 검색어의 경우에는 가중치를 감소시킴으로써 질의 벡터를 재형성한다.

Ide 는 SMART 시스템을 사용하여 Rocchio 의 연구를 확장하는 실험을 수행하고, 그 결과를 논문으로 발표하였다[Ide71a]. Ide 는 Rocchio 방법이 검색 효과에 긍정적인 결과를 제공함을 재확인하였으며, 또한 Rocchio 방법을 변형한 다음과 같은 3 가지 방법을 개발하였다. 첫째, Rocchio 방법으로부터 적합 문서와 비적합 문서의 수에 대한 정규화 요소를 제거한 다음과 같은 식을 사용하였다.

$$Q_{new} = Q_{old} + \sum_{r=1}^{n_{rel}} D_r - \sum_{n=1}^{n_{nonrel}} D_n$$

둘째, 첫번째 방법과 유사하나, 비적합 문서를 이용하지 않는 다음과 같은 식을 사용하였다.

$$Q_{new} = Q_{old} + \sum_{r=1}^{n_{rel}} D_r$$

셋째, 가장 높은 유사도를 갖는 비적합 문서의 벡터만을 이용하는 다음과 같은 식을 사용하였다.

$$Q_{new} = Q_{old} + \frac{1}{n_{rel}} \sum_{r=1}^{n_{rel}} D_r - T_{nonrel}$$

여기에서 T_{nonrel} 는 최상위 순위를 갖는 부적합 문서의 벡터이다. 위의 세가지 방법들에 대한 성능 평가 실험은 검색 효과 측면에서 평균적으로 커다란 차이를 보이지 않았다.

제 12 장 유사도 계산

정보 검색 시스템의 중요한 기능들 중의 하나는 문서와 질의 사이의 관련 정도를 나타내는 유사도를 계산하고, 계산된 유사도에 따라 문서에 순위를 부여하는 것이다. 높은 유사도, 즉 높은 순위를 부여 받은 문서일수록 질의에 대한 만족도가 크며, 사용자는 높은 순위의 문서를 우선적으로 검토함으로써 필요한 정보를 얻는데 소모되는 시간을 최소화할 수 있다.

정보 검색 모델은 질의와 문서 사이의 유사도 계산 방법에 대한 이론적 근거를 제공한다. 지금까지 유사도 계산을 정확하게 수행하는 정보 검색 모델에 대한 많은 연구가 수행되어 왔으며, 이러한 연구들에 의해 개발된 벡터 공간 모델, 추론 네트워크 모델, 2-포아송 모델 등과 같은 정보 검색 모델들은 우수한 검색 효과를 제공한다고 알려져 있다. 일반적으로 다수의 정보 검색 모델들에서 다음과 같이 문서 D 는 가중치가 부여된 색인어들의 벡터로 표현되며, 질의 Q 또한 가중치가 부여된 검색어들의 벡터로 표현된다.

$$D = \{(t_1, w_{d1}), (t_2, w_{d2}), \ldots, (t_n, w_{dn})\}$$
$$Q = \{(t_1, w_{q1}), (t_2, w_{q2}), \ldots, (t_n, w_{qn})\}$$

여기에서 색인어 가중치 w_{di}는 문서 D에서 i번째 색인어 t_i의 가중치를 의미하고, 검색어 가중치 w_{qi}는 질의 Q에서 i번째 검색어 t_i의 가중치를 의미한다. 그리고, 문서 D와 질의 Q 사이의 유사도는 다음과 같이 문서 벡터와 질의 벡터의 내적으로 표현된다.

$$Sim(D,Q) = \sum_{i=1}^{n} (w_{di} \times w_{qi})$$

위의 식에서 알 수 있듯이 문서와 질의 사이의 유사도 값은 색인어 및 검색어 가중치에 의해 결정되기 때문에, 이러한 가중치를 산출하는 식은 정보 검색 모델의 가장 중요한 요소들 중의 하나이다. 일반적으로 서로 다른 정보 검색 모델은 색인어 및 검색어 가중치를 산출하기 위하여 서로 다른 식을 사용한다.

한편, 질의와 문서가 위에서 설명된 벡터로 표현되어 있을 때, 정보 검색 시스템은 각각의 문서 벡터와 질의 벡터의 내적으로서 유사도를 계산하고, 이러한 유사도에 따라 문서에 순위를 부여할 수 있다. 그러나, 이러한 유사도 계산 방법을 사용할 경우, 질의 처리 시간은 정보 검색 시스템에 입력된 문서들의 수와 비례하며, 따라서 검색 대상 문서의 수가 증가할수록 질의 처리 시간이 증가하는 문제점을 지니고 있다. 이러한 문제점을 개선하기 위하여 렉시콘 파일과 포스팅 파일로 구성되는 역파일을 이용하여 질의와 문서 사이의 유사도를 계산하는 방법들이 개발되었으며, 본 장에서는 이러한 방법들에 대하여 기술한다.

12.1 기본적 누산기 방법

누산기는 유사도를 저장하는 저장 공간, 즉 변수를 의미하며, 각각의 문서에 대하여 1 개의 누산기가 할당된다. 다음은 기본적 누산기 방법을 단계별로 기술하고 있다. 단계 1 은 각각의 문서에 누산기를 할당하고 그 값을 초기화하며, 단계 2 는 역화일로부터 검색어에 대한 포스팅 리스트을 읽어 유사도를 계산한다. 마지막으로 단계 3 에서 누산기의 값에 따라 문서들을 정렬한 후, 사용자가 원하는 수만큼의 상위 문서들을 검색 결과로서 반환한다.

1. **누산기 생성**: 문서 집합에 포함된 각각의 문서 d 에 대하여 누산기 A_d 를 생성하고, 그 값을 0 으로 초기화한다. 즉,

$$A_d \leftarrow 0$$

2. **유사도 계산**: 질의에 포함된 각각의 검색어 (t, w_{qt})에 대해 다음을 수행한다.
   2.1 역화일로부터 검색어 t 에 대한 포스팅 리스트를 읽는다.
   2.2 포스팅 리스트에 포함된 각각의 포스팅 (d, w_{dt})에 대하여, 검색어 t 와 문서 d 사이의 부분 유사도 $w_{qt} \times w_{dt}$ 를 계산하여 이를 문서 d 의 누산기에 합산한다. 즉,

$$A_d \leftarrow A_d + w_{qt} \times w_{dt}$$

3. **검색 결과 생성**: 문서들을 누산기 값에 따라 정렬한 후, 사용자가 원하는 수만큼의 상위 문서들을 검색 결과로서 반환한다.

<그림 12.1>은 기본적 누산기 방법을 사용한 유사도 계산 과정을 그림으로 보여주고 있다. 기본적 누산기 방법은 문서 집합에 포함된 각각의 문서에 대하여 1 개의 누산기를 할당하기 때문에, 누산기로 인하여 소비되는 메모리의 양은 문서 집합에 포함된 문서들의 수에 비례한다. 그러나, <그림 12.1>에서도 알 수 있듯이 일반적으로 유사도 계산 단계의 종료 이후에도 초기화 값 0 을 그대로 유지하고 있는, 즉 불필요한 많은 수의 누산기들이 존재한다.

기본적 누산기 방법에서 누산기 생성 단계를 생략하고, 유사도 계산 과정 중에 필요에 따라 누산기를 생성하고 초기화함으로써 불필요한 누산기의 할당으로 인한 메모리의 낭비를 줄일 수 있다. 즉, 기본적 누산기 방법의 단계 1 과 2 를 다음과 같이 변경함으로써 유사도 계산 단계의 종료 이후에도 초기화 값 0 을 그대로 유지하고 있는 누산기의 생성을 방지할 수 있다.

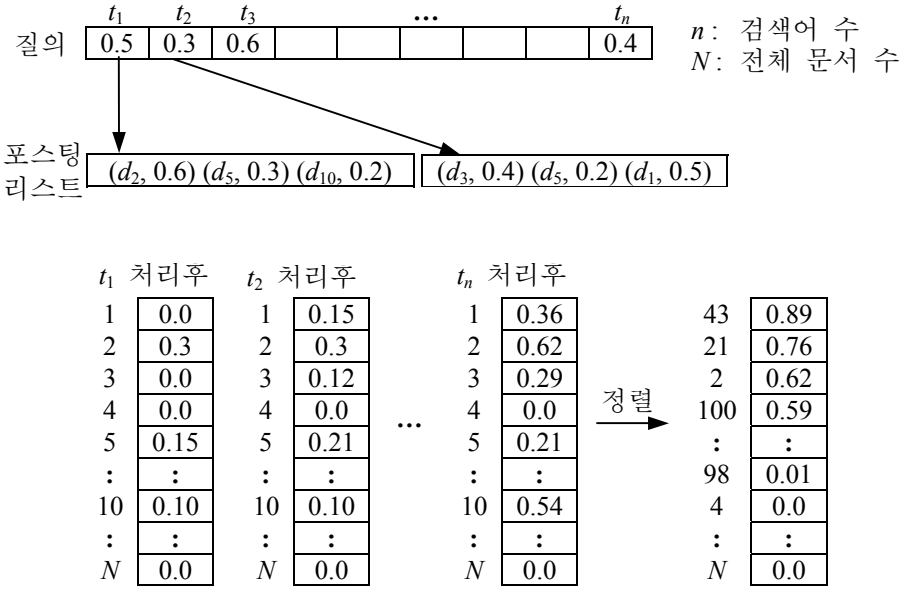
누산기 생성 및 유사도 계산: 질의에 포함된 각각의 검색어 (t, w_{qt})에 대해 다음을 수행한다.
(a) 역화일로부터 검색어 t 에 대한 포스팅 리스트를 읽는다.
(b) 포스팅 리스트에 포함된 각각의 포스팅 (d, w_{dt})에 대하여,
   A. 문서 d 에 대한 누산기가 존재하지 않을 경우, 누산기 A_d 를 생성하고 그

값을 0 으로 초기화한다.

B. 검색어 t 와 문서 d 사이의 부분 유사도 $w_{qt} \times w_{dt}$ 를 계산하여, 이를 문서 d 의 누산기 A_d에 합산한다. 즉,

$$A_d \leftarrow A_d + w_{qt} \times w_{dt}$$

위에서 설명된 2 가지 유사도 계산 방법들은 누산기 A_d 에 부분 유사도를 합산하기에 앞서 메모리 내에서 누산기 A_d 의 위치를 발견해야 한다. 기본적 누산기 방법에서 누산기들은 배열로서 구현되기 때문에, 배열의 첨자를 활용하여 누산기의 위치를 손쉽고 빠르게 발견할 수 있다. 그러나, 유사도 계산 과정 중에 누산기를 생성하는 방법에서 문서 번호와 유사도 필드들로 구성된 누산기는 해싱 또는 AVL 트리 등을 이용하여 관리되기 때문에, 기본적 누산기 방법과 비교하여 누산기의 위치 발견에 보다 많은 시간을 소비한다. 따라서, 메모리가 충분할 경우 누산기를 배열로서 구현하는 기본적 누산기 방법의 사용이 바람직하다.



<그림 10.1> 기본적 누산기 방법을 이용한 유사도 계산

12.2 검색 결과 생성 단계의 구현

기본적 누산기 방법의 마지막 단계는 누산기 값에 따라 문서들을 정렬한 후, 사용자가 원하는 수만큼의 상위 문서들을 검색 결과로서 반환한다. 따라서, 이 단계의 수행을 위해서는 정렬 함수의 사용이 필수적이며, 문서들의 효율적인 정렬을 위하여 다음과 같은 사항에 유의해야 한다. 많은 프로그래머들이 데이터의 정렬을 위하여 습관적으로 대부분의 운영 체제에서 지원하는 qsort 함수를 사용한다. 그러나, 이 qsort 함수는 인자로서 데이터의 비교 함수에 대한 포인터를 요구하며, 이로 인하여 정렬 속도의 감소가 초래된다. 따라서, 운영 체제에서 지원하는 qsort 함수를 사용하지 않고, 프로그래머가 정렬 함수를 직접 작성하여 사용하는 것이 바람직하다.

전통적으로 널리 사용되어 온 불리안 검색 시스템은 검색 결과로서 질의를 만족하는 문서 집합을 생성하고, 이 문서 집합을 사용자에게 제공한다. 그러나, 질의와 문서 사이의 유사도를 계산하고, 이 유사도 값에 근거하여 문서들에 순위를 부여하는 검색 시스템은 사용자에 의해 지정된 수만큼의 상위 문서들을 검색 결과로서 생성한다. 따라서, 유사도 계산이 종료된 이후에 문서 집합에 포함된 모든 문서들을 정렬하는 대신에, 사용자에 의해 지정된 수만큼의 상위 문서들을 추출하고, 추출된 문서들만을 정렬함으로써 검색 결과 생성에 소비되는 시간을 단축시킬 수 있다.

r 이 N 보다 매우 작은 숫자일 경우, N 개의 값들로부터 r 개의 상위 값들을 추출하는 문제는 알고리즘 분야에서 널리 알려져 있으며, r-선택이라고 불리는 이 문제는 힙 자료 구조를 이용하여 해결될 수 있다. 힙은 내부와 리프 노드들에 값이 저장되는 완전 이진 트리로서, 루트로부터 리프까지의 경로를 따라 값들이 내림차순으로 저장되는 자료 구조이다. 즉, 루트에는 힙에 저장되는 값들 중에서 가장 큰 값이 저장된다. 한편, 루트에는 힙에 저장되는 값들 중에서 가장 작은 값이 저장될 수도 있으며, 이때 루트로부터 리프까지의 경로를 따라 값들이 오름차순으로 저장된다. 일반적으로 루트에 가장 작은 값이 저장되는 힙 자료 구조는 최소 힙으로 지칭된다.

N 개의 값들로부터 r 개의 상위 값들을 추출하는 r-선택 문제는 최소 힙을 이용하여 다음과 같이 해결될 수 있다. 우선, N개의 값들 중에서 r개의 값들을 임의로 선택하여 r개의 노드로 구성된 최소 힙을 구축한다. 이때 최소 힙의 루트에 저장된 값은 r개의 값들 중에서 최소값이다. 그리고, 나머지 N-r개에 포함된 각각의 값 v에 대하여, v가 루트에 저장된 값보다 크면 루트에 저장된 값을 제거하고 v 를 최소 힙에 삽입한다. 이러한 과정이 종료된 이후에 최소 힙에는 N개의 값들 중에서 r개의 상위 값들만이 존재하게 된다.

제 13 장 한글 색인어 추출

한글 색인어 추출 방법들은 추출되는 색인어의 단위에 따라 어절 단위 색인법과 형태소 단위 색인법, *n*-Gram 기반 색인법으로 분류될 수 있다. 어절 단위 색인법은 문서와 질의의 각 어절들에 대해 색인어의 일부분으로서 가치가 없는 비색인 분절(non-indexable segment) 즉, 조사, 어미, 접미사 등의 음절들을 절단하여 원문에 가까운 형태로 색인어를 추출하는 것으로 색인 과정이 비교적 간단하다. 그러나, 이 방법은 비색인 분절을 절단할 때 오류가 발생할 수 있으며, 또한 복합 명사의 띄어 쓰기 문제를 적절히 처리하지 못하기 때문에, 문서들 내에 많은 복합 명사들이 포함되어 있을 경우 검색 효과가 저하되는 문제점을 지니고 있다.

형태소 단위 색인법은 일반적으로 형태소 해석을 수행함으로써 문장 중의 각 어절을 명사, 조사, 부사 등의 형태소 단위로 분리한 후, 문서나 질의의 내용 표현에 절절한 형태소들을 추출한다. 이 방법은 복합 명사를 단순 명사들로 분리할 수 있어 앞에서 언급한 어절 단위 색인법에서의 복합 명사의 띄어 쓰기 문제를 극복할 수 있다. 그러나, 형태소 단위 색인법은 형태소 해석을 위한 규칙이 복잡하고, 형태소 해석 결과의 애매성, 미등록어, 비문법적인 어절 등의 이유로 부정확한 색인어가 추출될 수 있다. 또한 형태소 사전과 같은 언어 정보들을 개발하고 유지 관리해야 하는 부담을 안고 있다.

n-Gram 기반 색인법은 어절 단위 색인법과 *n*-Gram 색인법을 결합한 방법으로, *n*-Gram 이란 인접한 *n* 개의 음절을 의미한다. *n*-Gram 기반 색인법은 문장 내의 각 어절에 대하여 어절 단위 색인법을 적용하고, 그 결과로 생성된 분절에 *n*-Gram 색인법을 적용함으로써 색인어들을 추출한다. 예를 들면, “정보검색을”이란 어절에 대하여 2-Gram 기반 색인법은 “정보”, “보검”, “검색”의 색인어들을 추출한다. 비록 2-Gram 기반 색인법이 “보검”과 같은 의미 없는 색인어를 추출할 지라도, 어절 단위 색인법에서의 복합 명사 띄어 쓰기 문제를 완화할 수 있으며, 또한 형태소 단위 색인법에서 필수적인 복잡한 문장 해석 규칙이나 언어 정보의 개발이 요구되지 않는다.

13.1 어절 단위 색인법

어절 단위 색인법은 문서나 질의로부터 어절들을 인식하고, 각 어절로부터 색인어의 부분으로서 무의미한 비색인 분절을 제거한 나머지 색인 분절(indexable segment)을 색인어 후보로 선정한 후, 이들로부터 불용어를 제거하는 방법이다(김영환 1982; 안현수 1986; 예용희 1992). 특히 한글에서 문서나 질의를 표현할 수 있는 체언이나 용언의 명사형 뒤에 조사나 접미사 등이 붙는다는 특성에 근거하여 어절로부터 조사나 접미사 등을 제거하는 데 중점을

두고 있다. 비색인 분절이란 아래와 같이 체언의 뒤에 붙여 쓰이지만 색인어에 포함시키기에는 무의미한 조사, 어미, 접미사 등의 음절들을 말한다.

는, 은, 가, 이	을, 를, 에, 에게
와, 과	부터, 로부터
들, 들도, 들의	마다, 만큼, 보다
로서, 로써	와의, 과의, 처럼
하다, 하는, 하도록	되다, 되는 되도록
당하다, 시키다	……

어절 단위 색인법의 색인 과정은 대체로 3 단계로 구성된다. 첫번째 단계에서는 빈칸과 같은 문자를 구분자로 하여 문서에서 어절들을 인식한다. 두번째 단계에서는 인식된 각 어절에 대해 비색인 분절을 제거한다. 일반적으로 비색인 분절의 검출을 위하여 최장 일치법(maximum matching mthod)이 이용된다. 최장 일치법이란 주어진 어절 내에서 검출될 수 있는 비색인 분절들 중에서 가장 긴 분절을 선택하는 방법이다. 예를 들면, “시스템으로부터”라는 어절이 있을 때 “으로부터”, “로부터”, “부터”의 비색인 분절 중에서 가장 긴 “으로부터”를 선택하여 이를 제거한다. 그리고 마지막 단계에서 어절 단위 색인법은 불용어를 제외한 나머지 색인 분절들을 색인어로 선정한다.

최장 일치법을 사용하여 비색인 분절을 제거하는 어절 단위 색인법은 구현이 간단한 반면, 비색인 분절을 검출하는 과정에서의 오류로 인하여 추출되는 색인어의 일관성이 떨어질 수 있다. 예를 들면, “벨기에로서는”과 “벨기에”의 두 어절에 대해 어절 단위 색인법은 각각 “로서는”과 “에”를 비색인 분절로 판정하고, “벨기에”와 “벨기”의 서로 다른 색인어들을 추출한다. 한글은 조사, 어미, 접미사 등과 같은 단순 비색인 분절을 2,000 개 이상 포함하고 있다. 또한, 이러한 단순 비색인 분절들을 서로 결합하여 복합 비색인 분절을 생성하는 한글의 특성은 비색인 분절의 정확한 검출을 어렵게 만드는 요인이 되고 있다.

한편, 한글은 복합 명사를 구성하는 단순 명사들 사이의 띄어 쓰기를 자유롭게 규정하고 있다. 이러한 한글 체계에서 문서들이 복합 명사를 많이 포함하고 있을 경우, 어절 단위 색인법은 검색 효과를 저하시키는 다음과 같은 문제점을 지닌다. 예를 들면, 문서 d_1과 d_2가 각각 “정보검색”과 “정보 전송”을 포함하고, 사용자의 질의 q_1은 “정보 검색”이라고 가정하자. 문서 d_1은 질의 q_1과 의미가 동일한 복합 명사를 다른 띄어 쓰기 형태로 포함하고, 문서 d_2는 문서 d_1보다 질의와 관련성이 적은 명사를 포함한다. 이들 문서 d_1, d_2와 질의 q_1에 대하여 어절 단위 색인법을 적용할 때, 문서 및 질의의 벡터 표현과 벡터 공간 모델에 의한 질의와 문서 사이의 유사도는 다음과 같다.

d_1 : { (정보검색, w_1) }
d_2 : { (정보, w_2), (전송, w_3) }
q_1 : { (정보, w_4), (검색, w_5) }

$Sim(d_1, q_1) = 0$
$Sim\ (d_2, q_1) = w_2 \times w_4$

따라서 문서 d_1은 질의 q_1과 다른 형태의 띄어 쓰기로 인하여 일치하는 색인어를 갖지 않기 때문에, 질의 q_1에 관련성이 적은 문서 d_2가 관련성이 많은 d_1보다 높은 유사도를 갖는다.

13.2 형태소 단위 색인법

형태소 단위 색인법은 <표 13.1>에서 보는 바와 같이 문장 분석의 정도에 따라 형태소 해석만을 이용하는 방법과 구문 해석을 이용하는 방법으로 구분된다. 형태소 해석만을 이용하는 방법은 문장의 모든 어절들에 대해 형태소 해석을 수행하여, 최소 의미의 명사들을 문서와 질의의 표현을 위한 색인어로 선정한다(이현아 외 3 인 1995). 이 방법은 형태소 해석, 애매성 제거, 단순 명사 추출, 불용어 제거의 네 단계를 거쳐 색인을 수행한다. 형태소 해석 단계에서는 문장을 최소의 의미 단위로 구분하여 문장 내의 각 어절을 구성하고 있는 형태소들을 파악한다(강승식 외 2 인 1995). 애매성 제거 단계에서는 단어나 어절 자체의 모호함 때문에 하나의 단어나 어절에 대해 여러 개의 해석 결과가 산출되는 형태소 해석의 애매함을 제거한다. 즉, 여러 개의 형태소 해석 결과로부터 하나를 선정한다. 마지막으로 선택된 해석 결과로부터 단순 명사와 같은 최소 의미의 형태소들을 추출하고, 불용어를 제외한 나머지를 색인어로 선정한다.

<표 13.1> 형태소 단위 색인법

	형태소 해석	구문 해석
색인과정	1. 어절들에 대한 형태소 해석 2. 형태소 해석의 애매성 제거 3. 명사 추출 4. 불용어 제거	1. 어절들에 대한 형태소 해석 2. 형태소 해석의 애매성 제거 3. 형태소 해석 결과를 이용한 구문 분석 4. 명사 추출 5. 불용어 제거

구문 해석을 통한 방법은 형태소 해석에서 한 단계 더 나아가 문장 단위의 구문 해석을 수행하여 문장에서 중요한 의미를 갖는 특정한 명사나 명사구를 색인어로 선정한다. 예를 들면, 색인어 추출을 위해 문장의 서술어와 그것과 연관된 명사구들의 의미 역할을 설정하는 격문법을 이용한 구문 해석이 시도되었다(한성현 1991 ; 최기선 1991 ; 정진성 1992). 여기에서는 서술어가 문장 중에 반드시 가져야 되는 격을 필수격이라 정의하고, 한국어 용언이 가질 수 있는 격틀(case frame)을 이용하여 각 문장의 필수격을 찾는 방식으로 구문해석을 수행한다. 그리고 필수격에 해당하는 명사나 명사구들을 색인어 후보로 채택하고 불용어를 제거한다.

어절 단위 색인법과 형태소 단위 색인법은 최종적으로 추출되는 색인어의 단위에서 구별된다. 어절 단위 색인법은 여러 개의 명사가 결합된 복합 명사를 포함한 어절의 경우에도 단순히 조사나 접미사 등의 절단만을 수행하고 나머지 분절을 색인어로 취한다. 반면에 형태소 단위 색인법은 복합 명사 어절을 최소 의미의 형태소로 분리하여 단순 명사를 색인어로 선정할 수 있다. 예를 들면, “정보검색서비스가”라는 어절에 대해, 전자의 방법은 “정보검색서비스”를 색인어로 선정하지만, 후자의 방법은 “정보”, “검색”, “서비스”의 색인어들을 선정할 수 있다.

복합 명사의 띄어 쓰기 문제는 형태소 단위 색인법을 사용하여 단순 명사들을 색인어로 추출함으로써 극복될 수 있다. 예를 들면, 문서 d_1 과 d_2 가 각각 “정보검색”과 “정보 전송”을 포함하고, 사용자의 질의 q_1 은 “정보 검색”이라고 가정하자. 문서 d_1, d_2 와 질의 q_1 에 대해 형태소 단위 색인법을 이용하여 단순 명사를 색인어로 추출할 때, 문서와 질의의 벡터 표현과 벡터 공간 모델에 의한 질의와 문서 사이의 유사도는 다음과 같다.

d_1 : { (정보, w_6), (검색, w_7) }
d_2 : { (정보, w_8), (전송, w_9) }
q_1 : { (정보, w_{10}), (검색, w_{11}) }

$Sim\ (d_1, q_1) = w_6 \times w_{10} + w_7 \times w_{11}$
$Sim\ (d_2, q_1) = w_8 \times w_{10}$

따라서 문서 d_2 의 색인어 ‘정보’의 가중치 w_8 이 아주 큰 값을 갖지 않는 한 유사도는 $Sim\ (d_1, q_1) > Sim\ (d_2, q_1)$로 계산되어 정확한 순위 결정이 이루어진다.

형태소 단위 색인법은 위의 예제에서 설명된 것처럼 복합 명사의 띄어 쓰기 문제를 잘 처리할 수 있으며, 검색 효과도 좋은 것으로 보고되고 있다. 그러나 형태소 해석이나 구문 해석과 관련하여 다음과 같은 몇 가지의 문제점을 지니고 있다. 첫째, 형태소 단위 색인법은 형태소 해석이나 구문 해석 과정에서 형태소 사전, 격틀 사전 등의 언어 정보들을 필요로 한다. 특히 형태소 사전은 많은 개발 시간과 비용을 요구하며, 형태소 해석의 대상이 되는

문서들의 성질에 크게 의존하는 경향이 있어 문서의 종류마다 서로 다르게 개발되어야 하는 부담을 안고 있다.

둘째, 형태소 단위 색인법은 형태소 해석에 의존하므로 형태소 해석 과정에서의 오류는 부정확한 색인어들의 생성을 야기한다. 형태소 해석의 가장 큰 오류는 사전 내에 등록되지 않은 미등록어로 인해 발생하며, 특히 과학 기술 분야에서는 많은 전문 용어들이 미등록어로 처리되어 검색 효과가 감소될 수 있다. 그 외에도 단어나 어절 자체의 모호함에서 오는 형태소 해석의 애매성이나, 실제 문서에서 많이 발견되는 철자 오류와 띄어 쓰기 오류 등의 비문법적인 어절들도 형태소 해석 오류의 원인이 되고 있다. <표 13.2>는 이러한 오류의 예를 보여준다.

셋째, 형태소 단위 색인법은 형태소 해석이나 구문 해석 과정에서 복잡한 규칙을 요구한다. 앞에서 언급된 오류들에 대처하기 위한 형태소 해석의 규칙은 자연 복잡해질 수 밖에 없으며, 문서 내의 어절이나 문장들은 다양한 형태의 구조를 갖고 있고 예외적인 상황도 많이 발생할 수 있어 복잡한 구문 해석 규칙이 요구된다. 예를 들어, 문장 구성 성분들의 순서가 도치되거나 구성 성분의 역할을 결정하는 조사나 어미 등의 생략은 구문 해석을 어렵게 한다.

<표 13.2> 형태소 해석 오류의 예

원 인	입력 어절	잘못된 형태소 해석 결과
애매한 형태소 해석	10 년 형을 **선고함**	**선**/보통명사 + **고함**/보통명사 **선고**/보통명사 + **함**/보통명사
미등록어	**가내공장**으로	**가**/보통명사 + **내**/보통명사 + **공장**/보통명사 + 으로/부사격조사
비문법적 어절	**추출하였다고하자**	**추출하였다고**/동작성보통명사 + **하**/파생접미사 + **자**/연결어미
띄어쓰기 오류	**공유할수있는**	**공유할수있**/보통명사 + **는**/주격조사

13.3 *n*-Gram 기반 색인법

n-Gram 기반 색인법은 기존의 어절 단위 색인법과 *n*-Gram 방법(Cavnar 1994; Damashek 1995)의 결합을 통하여 개발되었다. 다음은 *n*-Gram 기반 색인법을 단계별로 기술하고 있다. 단계 1 부터 단계 3 까지는 어절 단위 색인법에 대한 설명이며, 단계 4 는 어절 단위 색인법에 *n*-Gram 방법을 적용하는 과정이다. <표 13.3>은 *n*-Gram 기반 색인법의 예를 보여준다.

단계 1: 문서나 질의를 색인하기 위해 먼저 빈칸, 마침표, 쉼표, 따옴표 등을 구분자로 하여 모든 어절들을 추출한다.

단계 2: 나머지 어절들에 대해 최장 일치법을 이용하여 비색인 분절을 절단한다. 비색인 분절은 단일 조사(-가, -이, -를, -으로, -부터), 복합 조사(-으로부터, -에서부터), 조사, 어미, 접미사 등이 결합된 다양한 형태의 음절들을 포함한다. 예를 들면, 다음과 같은 어절들에서 "색인" 뒤에 오는 모든 문자열이 여기에 포함된다.

색인**을**	색인**하여**	색인**하였는데**
색인**되어**	색인**되었으니**	색인**임을**
색인**이기에**	색인**이라고**	색인**이지만**

단계 3: 불용어 리스트를 이용하여 색인어로서 무의미한 어절들을 제거한다. 한글에서는 단어에 다양한 종류의 조사나 어미 등이 붙을 수 있고 복합어와 동사의 활용이 다양하므로 불용어 선정에 신중을 기해야 한다.

단계 4: 생성된 각각의 색인 분절에 대해 *n*-Gram 방법을 적용한다. *n*-Gram 이란 인접한 *n* 개의 음절을 말한다(Cavnar 1994; Damashek 1995). 예를 들면, "프로그래밍"이란 어절에 대해 2-Gram 은 "프로", "로그", "그래", "래밍"이며, 3-Gram 은 "프로그", "로그래", "그래밍"이다. 색인 분절의 음절 수가 *n* 보다 큰 경우에는 색인 분절을 여러 개의 *n*-Gram 들로 분리하고, 작은 경우에는 색인 분절 전체를 하나의 *n*-Gram 으로 취한다.

<표 13.3> 2-Gram 기반 색인법의 예

내년 중반부터 정보검색서비스가 실시된다.	
단계 1 :	문장 내의 어절 인식
	내년, 중반부터, 정보검색서비스가, 실시된다
단계 2 :	비색인 분절의 절단
	내년, 중반, 정보검색서비스, 실시
단계 3 :	불용어 제거
	정보검색서비스, 실시
단계 4 :	2-Gram 의 적용
	정보, 보검, 검색, 색서, 서비, 비스, 실시

n-Gram 기반 색인법은 어절 단위 색인법에서의 복합 명사 띄어 쓰기 문제를 완화하며, 형태소 단위 해석에서와 같은 복잡한 문장 해석 규칙이나 언어 정보의 개발을 요구하지 않는다. 다음에서는 *n*-Gram 기반 색인법이 어절 단위 색인법과 형태소 단위 색인법의 문제점을 완화시키는 이유와 검색 효과의 측면에서 *n*-Gram 기반 색인법이 지니는 장점에 대하여 설명한다.

- *n*-Gram 기반 색인법은 어절 단위 색인법을 이용할 때의 절단 오류로 인한 파급 효과를 완화한다. 예를 들면, 어절 "벨기에로서는"과 "벨기에"는 어절 단위 색인 과정에서 "벨기에"와 "벨기"로 색인된다. 여기에 2-Gram 방법을 적용하면 모두 "벨기"의 공통된 색인어가 생성된다.

- *n*-Gram 기반 색인법은 복합 명사의 띄어 쓰기 문제를 완화한다. 예를 들면, 아래와 같은 문서 $d_1, \ldots, d_5$와 질의 q_1, q_2가 있다고 가정하자.

  d_1: 과학기술정보 유통의
  d_2: 과학기술 정보유통의
  d_3: 과학 기술 정보 유통의
  d_4: 과학기술 분야의 정보를 유통하기 위한
  d_5: 과학과 기술의 정보를 유통하기 위한
  q_1: 과학기술정보유통에 관한
  q_2: 과학 기술 정보 유통에 관한

  2-Gram 기반 색인법은 이들 문서와 질의에 대해 다음과 같은 색인어들을 생성한다.

  d_1: {과학, 학기, 기술, 술정, 정보, 유통}
  d_2: {과학, 학기, 기술, 정보, 보유, 유통}
  d_3: {과학, 기술, 정보, 유통}
  d_4: {과학, 학기, 기술, 분야, 정보, 유통}
  d_5: {과학, 기술, 정보, 유통}
  q_1: {과학, 학기, 기술, 술정, 정보, 보유, 유통}
  q_2: {과학, 기술, 정보, 유통}

  이와 같은 경우 질의 q_1과 q_2의 복합 명사 띄어 쓰기가 서로 다르지만, 유사도를 계산하는 벡터 공간 모델에서 검색을 수행할 때, 모든 문서들은 두 질의에 대하여 높은 유사도를 갖는 문서로서 검색될 가능성이 크다.

- 한글 문서들을 살펴보면 아래의 예와 같이 단순 명사의 뒤에 한 글자의 명사가 붙거나 또는 파생 접사가 붙어서 형성된 명사들을 많이 발견할 수 있다. 형태소 단위 색인법에서는 이러한 명사를 보통 단일 형태소로 취급하여 색인어로 추출한다.

가공기	가공력	가공도	가공량	가공면
가공물	가공법	가공부	가공비	가공사
가공성	가공상	가공수	가공압	가공업
가공열	가공용	가공률	가공재	가공칩
가공품	가공형	가공자	가공학	…

이러한 경우 “가공”의 질의가 입력되면, 관련된 많은 문서들이 검색되지 않을 수 있다. 제안하는 색인 방법은 이와 같은 경우에 관련 문서의 검색을 도와 준다.

- *n*-Gram 기반 색인법은 철자 오류나 일관성이 없는 외래어 표기 문제를 적절히 완화할 수 있다. 예를 들면, 문서 d_1 이 “정보검섹”으로 잘못 표기된 어절을 포함하고, 사용자는 “정보검색”으로 질의 q_1 을 입력한다고 가정하자. 2-Gram 기반의 색인법은 문서 d_1 과 질의 q_1 에 대해 각각 다음과 같은 벡터 표현을 형성한다.

  d_1 : {(정보, w_1), (보검, w_2), (검섹, w_3)}
  q_1 : {(정보, w_4), (보검, w_5), (검색, w_6)}

  따라서 문서에 “검섹”의 철자 오류가 있더라도 문서는 질의의 결과로 검색될 가능성이 높다. 서로 다른 외래어 표기의 문제도 이와 유사하다. 사용자마다 “database”를 “데이터 베이스”로 표기하기도 하고 “데이터베이스”로 표기하기도 한다. 어떤 식으로 문서에 표기되어 있든 *n*-Gram 기반 색인법을 이용하는 시스템에서는 서로 다른 표기법이 사용된 문서가 비슷한 수준의 유사도를 갖고 검색될 가능성이 높다.

n-Gram 기반 색인법은 어절 단위 색인법과 형태소 단위 색인법에 비교하여 많은 수의 색인어를 추출하기 때문에, 인덱스 저장에 보다 많은 저장 공간을 사용한다. 또한 의미없는 *n*-Gram 의 생성으로 인해 질의에 부적합한 문서들이 검색될(false match) 가능성이 있으며, 특히 가중치 기법과 관련하여 이들 부적합 문서들에 상위의 순위가 부여될 수 있다. 예를 들면, 다음과 같은 문서 d_1 과 질의 q_1 이 있다고 가정하자.

d_1 : …자방친 및 화분친에 따라 감자 반수체 **유기효율**이 컸으며…
q_1 : 배기관 형상에 따른 2 행정 기관의 **소기효율** 및 성능 예측

이때 문서 d_1 의 “유기효율”과 질의 q_1 의 “소기효율”에 대해 *n*-Gram 기반 색인법은 각각 {유기, 기효, 효율}과 {소기, 기효, 효율}의 색인어를 생성한다. 여기에서 “기효”가 일치하므로 문서 d_1 은 질의 q_1 에 관련이 없는데도 검색 결과로서 출력될 수 있으며, 만일 “기효”가 높은 가중치 값을 부여받는다면 문서 d_1 과 질의 q_1 사이의 유사도가 커져 문서 d_1 이 상위의 순위를 부여받을 수 있다. 따라서 이러한 문제를 해결할 수 있는 처리 방안이 고려되어야 한다.

제 14 장 중국어 색인어 추출

일반적으로 사용되고 있는 중국어 표준 코드는 GB2312-80 으로 7,445 개의 문자를 포함하고 있으며, 이들 중에서 6,763 개의 문자가 한자이다. 중국어에서 문자 수에 따른 단어의 비율을 살펴보면, 하나의 문자로 구성된 단어가 5%, 두 개의 문자로 구성된 단어가 75%, 세 개의 문자로 구성된 단어가 14%, 네 개의 문자 이상으로 구성된 단어가 6%를 차지한다. 한편, 문자 수에 따른 단어의 사용 비율을 살펴보면, 하나의 문자로 구성된 단어가 68%, 두 개의 문자로 구성된 단어가 30%, 세 개의 문자로 구성된 단어가 1%, 네 개의 문자 이상으로 구성된 단어가 1%의 비율로 사용되고 있다.

문서로부터 그 문서의 내용을 반영하는 색인어를 추출하는 과정은 정보 검색 시스템의 필수적인 요소이다. 중국어는 단어 사이의 구분이나 공백이 없기 때문에, 중국어 문서로부터 색인어를 추출하기 위해서는 중국어 문자열을 단어들로 분할하는 작업이 선행되어야 한다. 본 장에서는 중국어 문자열을 단어들로 분할하기 위하여 지금까지 개발된 사전 기반 분할법, 상호 정보 기반 분할법, 분할 확률 기반 분할법, 단어 통계 기반 분할법과 같은 방법들에 대하여 설명한다.

14.1 사전 기반 분할법

사전 기반 분할법은 사전을 이용하여 중국어 문자열을 단어들로 분할하는 방법으로, 최장 및 최단 일치법과 같은 간단한 휴리스틱을 사용하는 방법과 문법과 같은 복잡한 언어 지식을 사용하는 방법들이 이에 속한다. 일반적으로 복잡한 언어 지식을 사용할수록 보다 정확한 분할 결과를 얻을 수 있으며, 사전에 대한 지속적인 관리가 필요하다는 단점을 지니고 있다. 다음에서는 최장 및 최단 일치법을 사용하여 중국어 문자열을 단어들로 분할하는 방법에 대하여 자세히 기술한다.

최장 일치법은 입력 문자열의 일부가 사전에 포함된 단어와 일치하면 그 단어를 입력 문자열로부터 분할한다. 이때 둘 이상의 단어가 일치될 수 있으며, 최장 일치법은 가장 긴 단어를 입력 문자열로부터 분할한다. 한편, 입력 문자열의 분할을 진행하는 방향으로 두 가지 방향, 즉 순방향(forward)과 역방향(backward)을 사용할 수 있다. 순방향 분할은 입력 문자열의 앞에서부터 뒤쪽으로 단어의 분할을 수행하며, 역방향 분할은 입력 문자열의 뒤에서 앞쪽으로 단어의 분할을 수행한다. 예를 들어, <표 14.1>은 최장 순방향 일치법과 최장 역방향 일치법을 사용하여 입력 문자열 “中國大陸新發現的油田”을 분할하는 과정을 보여준다.

<표 14.1> 최장 일치법을 이용한 중국어 단어 분할

단계	최장 순방향 일치법		최장 역방향 일치법	
	문자열	행동	문자열	행동
1	中國大陸新發現的油田	中國 제거	中國大陸新發現的油田	油田 제거
2	大陸新發現的油田	大陸 제거	中國大陸新發現的	的 제거
3	新發現的油田	新 제거	中國大陸新發現	發現 제거
4	發現的油田	發現 제거	中國大陸新	新 제거
5	的油田	的 제거	中國大陸	大陸 제거
6	油田	油田 제거	中國	中國 제거
7	종료			

최단 일치법은 최장 일치법과 반대로 사전에 포함된 단어들 중에서 입력 문자열의 일부와 일치되는 가장 짧은 단어를 입력 문자열로부터 분할한다. 중국어 문서는 하나의 문자로 구성된 단어들을 많이 포함하고 있으며, 이러한 단어들의 분할을 위하여 사전 기반 분할법에서 사용되는 사전은 하나의 문자로 구성된 다수의 단어들을 포함하고 있다. 이처럼 사전에 짧은 길이의 단어들이 많이 포함되어 있을 경우, 최단 일치법은 분할에 있어 많은 오류를 발생시키며, 따라서 최장 일치법이 최단 일치법보다 중국어 단어 분할에 적합한 것으로 알려져 있다. 한편, 최단 일치법도 입력 문자열의 분할을 진행하는 방향에 따라 최단 순방향 일치법과 최단 역방향 일치법으로 구분될 수 있다.

14.2 상호 정보 기반 분할법

두개의 문자에 대한 상호 정보 계수(mutual information coefficient)는 두개 문자 사이의 관련 정도를 나타낸다. 즉, 상호 정보 계수는 두개의 문자가 인접해서 사용될 가능성이 높을수록 큰 값을 갖는다. 상호 정보 계수를 이용한 중국어 단어 분할 방법은 다음과 같다.

(1) 문서 집합으로부터 모든 문자에 대한 출현 빈도를 계산한다.

(2) 문서 집합으로부터 모든 2-Gram 에 대한 출현 빈도를 계산한다.

(3) 다음의 수식을 사용하여 각각의 2-Gram c_1c_2 에 대한 상호 정보 계수 $M(c_1,c_2)$를 계산한다.

$$M(c_1,c_2)=\log_2\frac{p(c_1,c_2)}{p(c_1)p(c_2)}=\log_2\frac{p(c_1)p(c_2\mid c_1)}{p(c_1)p(c_2)}=\log_2\frac{\frac{f(c_1)}{N}\frac{f(c_1c_2)}{f(c_1)}}{\frac{f(c_1)}{N}\frac{f(c_2)}{N}}=\log_2\frac{f(c_1c_2)\times N}{f(c_1)f(c_2)}$$

여기에서 $p(c_1)$과 $p(c_2)$는 각각 문자 c_1 과 c_2 가 출현할 확률이고, $p(c_1,c_2)$는 문자 c_1 과 c_2 가 인접해서 출현할 확률을 의미한다. 또한, $f(c_1)$과 $f(c_2)$는 각각 문자 c_1 과 c_2 의 출현 빈도이고, $f(c_1\,c_2)$는 2-Gram c_1c_2 의 출현 빈도이며, N 은 문서에 포함된 전체 문자 수이다.

(4) 상호 정보 계수를 이용하여 중국어 문자열 "中國大陸新發現的油田"을 분할하는 방법은 다음과 같다.

2-Grams	***M*(c_1,c_2)**	**분할 결과**
中國	4.69	
國大	0.18	
大陸	6.13	中國 大陸 新發現的 油田 (b)
陸新	-1.46	
新發	-0.30	
發現	4.07	中國 大陸 新 發現 的 油田 (c)
現的	0	
的油	-0.30	
油田	7.87	中國大陸新發現的 油田 (a)

(a) 가장 큰 상호 정보 계수를 갖는 2-Gram 油田을 문자열로부터 분리한다. (b) 2 번째로 큰 상호 정보 계수를 갖는 2-Gram 大陸을 문자열로부터 분리한다. 이로 인하여 2-Gram 中國도 문자열로부터 분리된다. (c) 마지막으로 문자열 新發現的이 분할되어야 하며, 이 문자열에 포함된 2-Gram 들 중에서 가장 큰 상호 정보 계수를 갖는 發現을 문자열로부터 분리한다.

위의 과정을 수행한 후 중국어 문자열 "中國大陸新發現的油田"은 "中國 大陸 新 發現 的 油田"로 분할된다. 즉, 상호 정보 기반 분할법은 입력된 문자열을 하나 또는 두 문자로 이루어진 단어들로 분할한다. 중국어는 자주 사용되는 단어들이 하나 또는 두 글자로 구성된다는 특성을 지니고 있기 때문에, 상호 정보 기반 분할법은 비교적 좋은 결과를 제공한다. 그러나 세 문자 이상으로 이루어진 단어들을 분할하지 못하는 단점을 지니고 있다.

14.3 분할 확률 기반 분할법

분할 확률 기반 분할법은 상호 정보 기반 분할법과는 반대로 2 개의 인접된 문자가 분할될 확률 $p_{seg}(c_1c_2)$를 계산하고, 분할 확률 $p_{seg}(c_1c_2)$를 이용하여 문자열을 분할하는 방법이다. 분할 확률 기반 분할법은 일본어의 한자 문자열을 분할하기 위해서 개발되었으나, 언어의 유사성으로 인하여 중국어 문자열의 분할에도 적용될 수 있다. 분할 확률 $p_{seg}(c_1c_2)$는 다음과 같이 계산된다.

$$p_{seg}(c_1c_2) = p_{tail}(c_1) \times p_{head}(c_2)$$

$$p_{head}(c) = \frac{\#(c \text{ appeared at the head of words})}{\#(c \text{ appeared at any place})}$$

$$p_{tail}(c) = \frac{\#(c \text{ appeared at the tail of words})}{\#(c \text{ appeared at any place})}$$

여기에서 #(x)는 x 의 출현 빈도이고, $p_{head}(c)$와 $p_{tail}(c)$는 각각 문자 c 가 단어의 처음에 출현할 확률과 끝에 출현할 확률을 의미한다. 분할 확률 $p_{seg}(c_1c_2)$는 $p_{head}(c)$와 $p_{tail}(c)$의 곱으로 계산되기 때문에, $p_{seg}(c_1c_2)$의 값이 클수록 문자 c_1 과 c_2 가 분할되는 것이 바람직하다.

<표 14.2>는 중국어 문자열 "熱帶雨林"에 포함된 각 문자들의 분할 확률을 보여주고 있다. 이때 분할 확률을 이용하여 중국어 문자열 "熱帶雨林"을 분할하는 방법은 다음과 같다. 첫째, 문자열 분할에 선행하여 분할을 위한 기준값(threshold)을 설정한다. 둘째, 설정된 기준값보다 큰 분할 확률을 갖는 문자들 사이를 분리한다. 예를 들면, 기준값이 0.2 인 경우 "熱帶", "雨", "林"으로 분할되며, 0.3 인 경우 "熱帶", "雨林"으로 분할된다. 즉, 기준값이 작을수록 입력 문자열은 많은 수의 단어들로 분할된다. 따라서 기준값의 설정에 따라 분할 확률 기반 분할법의 성능이 변화하며, 일반적으로 기준값은 실험을 통하여 결정된다.

<표 14.2> 분할 확률을 이용한 중국어 단어 분할

	p_{head}	p_{tail}	p_{seg}
熱	0.7541	0.3599	
			0.0916
帶	0.2546	0.8573	
			0.5886
雨	0.6866	0.7377	
			0.2677
林	0.3629	0.8512	

14.4 단어 확률 기반 분할법

단어 확률 기반 분할법은 단어 분할에 선행하여 문자열과 단어열 쌍들로 구성된 훈련 집합을 이용하여 단어 확률 사전을 구축한다. 즉, 훈련 집합은 다음과 같이 문자열과 이 문자열에 대한 단어열로 구성되며, 일반적으로 이러한 단어열은 사람이 문자열을 분할함으로써 생성된다.

中國大陸新發現的油田 → 中國/大陸/新/發現/的/油田

그리고, 훈련 집합으로부터 생성된 통계 정보를 이용하여 훈련 집합의 단어열에 포함된 각 단어 W에 대하여 다음과 같이 단어 확률 $p(W)$를 산출한다.

$$p(W) = \frac{\text{훈련 집합에서 단어 } W\text{가 출현하는 단어열의 수}}{\text{훈련 집합에서 단어 } W\text{가 출현하는 문자열의 수}}$$

이러한 방법으로 계산된 단어 확률은 대응하는 단어와 함께 단어 확률 사전에 등록된다.

한편, 단어 확률 기반 분할법은 문자열을 분할하기 위해 문자열을 구성하는 모든 단어들에 대한 단어 확률을 요구하며, 일반적으로 단어 확률 사전에 포함되는 단어들은 사람에 의해 생성된 훈련 집합으로부터 생성된다. 그러나, 많은 경우에 훈련 집합에 포함되지 않은 단어들이 문자열에 존재할 수 있으며, 이러한 문제점을 보완하기 위해 임의의 단어들을 단어 확률 사전에 등록시키고, 이 단어들에 대한 단어 확률로서 사람에 의해 지정된 일정한 값이 사용된다.

단어 확률 기반 분할법은 단어 확률 사전을 이용하여 다음과 같이 임의의 문자열을 단어들로 분할한다.

(1) 문자열로부터 단어로 인식될 수 있는 모든 부분 문자열을 추출한다. 예를 들면, 문자열 "大會決議"로부터 다음과 같은 6 개의 부분 문자열들이 생성될 수 있다.

大, 會, 決, 議, 大會, 決議

(2) 문자열로부터 생성될 수 있는 모든 단어열들을 생성한다. 즉, 단계 1 에서 생성된 부분 문자열을 조합하여 단어열들을 생성한다. 예를 들면, 문자열 "大會決議"의 부분 문자열들을 조합함으로써 생성될 수 있는 단어열들은 다음과 같다.

大/會/決/議, 大會/決/議, 大/會/決議, 大會/決議

(3) 단계 2 에서 생성된 각 단어열에 대하여 단어열 확률을 계산하고, 가장 높은 단어열 확률을 갖는 단어열을 문자열에 대한 분할 결과로서 채택한다. 이때 단어열 확률은 구성 단어들의 단어 확률들을 곱한 값으로 정의된다. 예를 들면, 단계 1 에서 생성된 부분 문자열들의 단어 확률을 大(0.016), 會(0.029), 決(0.00108), 議(0.0005), 大會(1.0), 決議(0.956)로 가정할 경우, 단계 2 에서 생성된 단어열들의 단어열 확률은 다음과 같이 계

산되며, 따라서 단어열 “大會/決議”이 분할 결과로서 채택된다.

大/會/決/議 ：0.016×0.029×0.00108×0.0005 = 0.00000000025956
大會/決/議 ：1.0×0.00108×0.0005 = 0.00000054
大/會/決議 ：0.016×0.029×0.956 = 0.000443584
大會/決議 ：1.0×0.956 = 0.956

제 15 장 확장 불리안 모델

퍼지 집합, Waller-Kraft, Paice, P-Norm, Infinite-One 과 같은 확장 불리안 모델들은 기존의 불리안 검색 시스템에 순위 결정 기능을 부여하기 위하여 개발되어 왔다(Sachs, 1976; Smith 1990; Fox et al. 1992). 이들은 문서 내에서 색인어의 중요성을 반영하는 색인어 가중치를 이용하는 공통된 특성을 지니고 있다. 확장 불리안 모델을 기반으로 하는 정보 검색 시스템은 다음의 <*T, Q, D, F*>에 의해 정의된다.

- T 는 질의와 문서를 표현하기 위해 사용되는 색인어들의 집합이다.
- Q 는 시스템이 인식할 수 있는 질의들의 집합이다. Q 에 속하는 각각의 질의 q 는 색인어들과 논리 연산자 *AND, OR, NOT* 으로 구성된 불리안 수식이다.
- D 는 문서들의 집합이다. D 에 속하는 각각의 문서 d 는 w_i 가 색인어 t_i 의 가중치일때, $\{(t_1, w_1), \ldots, (t_n, w_n)\}$와 같이 표현된다. 색인어 가중치 w_i 는 0 부터 1 사이의 값을 갖는다.
- F 는 문서값을 계산하는 순위 결정 함수(Ranking Function)로서 다음과 같이 정의된다.

  $$F : D \times Q \rightarrow [0, 1]$$

  함수 F 는 각 쌍의 (d, q)에 0 부터 1 사이의 값을 부여한다. 이 값은 문서 d 와 질의 q 사이의 유사도이며, 질의 q 에 대한 문서 d 의 문서값이라고 일컬어진다. *AND* 와 *OR* 에 대한 연산자 계산식은 순위 결정 함수의 성능을 결정하는 가장 중요한 요소이다. <그림 15.1>은 확장 불리안 모델에서 *AND* 와 *OR* 연산을 위해 사용된 연산자들을 보여준다.

퍼지 집합 모델의 연산자 계산식은 단지 2 개의 피연산자를 갖는 이항 연산자이고, Waller-Kraft, Paice, P-Norm, Infinite-One 모델의 연산자 계산식은 2 개 이상의 연산자를 갖는 다항 연산자이다. 이것은 *MIN* 과 *MAX* 연산자가 결합법칙을 만족하는데 비하여, Waller-Kraft, Paice, P-Norm, Infinite-One 모델의 연산자는 결합법칙을 만족하지 못하기 때문이다. 결합법칙을 만족하지 못하는 이항 연산자의 사용은 2 개의 논리적으로 동등한 질의 $(t_1 \; AND \; t_2) \; AND \; t_3$ 와 $t_1 \; AND \; (t_2 \; AND \; t_3)$에 대해 서로 다른 문서값을 생성한다.

$$F(d, t_1\ AND\ t_2) = MIN(w_1, w_2)$$
$$F(d, t_1\ OR\ t_2) = MAX(w_1, w_2)$$

(a) 퍼지 집합 모델

$$F(d, t_1\ AND \cdots AND\ t_n) = (1-r)\cdot MIN(w_1, \cdots, w_n) + r \cdot MAX(w_1, \cdots, w_n),\ 0 \le r \le 0.5$$
$$F(d, t_1\ OR \cdots OR\ t_n) = (1-r)\cdot MIN(w_1, \cdots, w_n) + r \cdot MAX(w_1, \cdots, w_n),\ 0.5 \le r \le 1$$

(b) Waller-Kraft 모델

$$F(d, t_1\ AND \cdots AND\ t_n) = \frac{\sum_{i=1}^{n}(r^{i-1} \cdot w_i)}{\sum_{i=1}^{n} r^{i-1}},\ 0 \le r \le 1\ and\ w_i's\ are\ considered\ in\ ascending\ order$$

$$F(d, t_1\ OR \cdots OR\ t_n) = \frac{\sum_{i=1}^{n}(r^{i-1} \cdot w_i)}{\sum_{i=1}^{n} r^{i-1}},\ 0 \le r \le 1\ and\ w_i's\ are\ considered\ in\ descending\ order$$

(c) Paice 모델

$$F(d, t_1\ AND\ \cdots\ AND\ t_n) = 1 - \left(\frac{\sum_{i=1}^{n}(1-w_i)^p}{n} \right)^{1/p},\ 0 \le p \le \infty$$

$$F(d, t_1\ OR\ \cdots\ OR\ t_n) = \left(\frac{\sum_{i=1}^{n} {w_i}^p}{n} \right)^{1/p},\ 0 \le p \le \infty$$

(d) P-Norm 모델

$$F(d, t_1\ AND \cdots AND\ t_n) = r \cdot (1 - MAX(1-w_1, \cdots, 1-w_n)) + (1-r)\frac{\sum_{i=1}^{n} w_i}{n},\ 0 \le r \le 1$$

$$F(d, t_1\ OR \cdots OR\ t_n) = r \cdot MAX(1-w_1, \cdots, 1-w_n) + (1-r)\frac{\sum_{i=1}^{n} w_i}{n},\ 0 \le r \le 1$$

(e) Infinite-One 모델

<그림 15.1> AND 와 OR 연산에 대한 연산식

15.1 긍정적 보상 연산자

퍼지 집합 모델은 문서들의 순위를 결정하는 문서값을 계산함으로써 불리안 검색 시스템의 단점을 극복하였을 지라도, 부정확한 문서값을 생성하는 요인을 지니고 있기 때문에 정보 검색 모델로서 부적합하다고 비판되어 왔다(Bookstein 1980; Lee et al. 1994). 이것은 퍼지 집합 모델이 *AND* 와 *OR* 연산을 위하여 사용하는 *MIN* 과 *MAX* 연산자가 검색효과를 저하시키는 특성을 지니고 있기 때문이다.

퍼지 집합 이론이 개발된 이후로 *MIN* 과 *MAX* 연산자를 대신할 수 있는 다양한 퍼지 연산자들이 제안되어 왔다. 이들 퍼지 연산자들이 확장 불리안 모델에서 *AND* 와 *OR* 연산을 위해 사용될 때, 단일 피연산자 의존 특성(Single Operand Dependency Property)과 부정적 보상 특성(Negative Compensation Property)을 지니는 연산자는 검색 효과를 저하 시킴이 입증되었다 (Kim et al. 1993; Lee et al. 1992; Lee et al. 1993).

단일 피연산자의 의존 문제: 임의의 연산자 θ가 단일 피연산 의존 특성을 갖는다면, $\theta(x, y) = x$ 또는 y (x, $y \in [0,1]$, $x \neq y$) 이다. *AND* 와 *OR* 연산을 위하여 단일 피연산자 의존 특성을 갖는 연산자를 사용하는 확장 불리안 모델은 단일 피연산자 의존 문제를 발생시킨다. 예를 들면, 두개의 문서 d_1, d_2 와 질의 q_1 이 다음과 같이 주어졌다고 가정하자.

d_1 = {(Information, 0), (Retrieval, 0)}
d_2 = {(Information, 1), (Retrieval, 0)}
q_1 = Information *AND* Retrieval

곱하기 연산자가 *AND* 연산을 위해 사용될 때, 질의 q_1 에 대한 문서 d_1, d_2 의 문서값은 모두 0 으로 동일하다. 그러나 대부분의 사람들은 d_1 보다 d_2 가 질의 q_1 에 유사하다고 결정할 것이다. 이러한 부정적 결과는 곱하기 연산자가 단일 피연산자 의존 특성을 지니고 있기 때문이다. 즉, $x \cdot 0 = 0$ ($x \in (0, 1)$).

부정적 보상 문제: 임의의 연산자 θ 가 부정적 보상 특성을 갖는다면, $\theta(x, y) < MIN(x, y)$ 또는 $\theta(x, y) > MAX(x, y)$ (x, $y \in [0, 1]$)이다. *AND* 와 *OR* 연산을 위하여 부정적 보상 특성을 갖는 연산자를 사용하는 확장 불리안 모델은 부정적 보상 문제를 발생시킨다. 예를 들면, 문서 d_3와 두 개의 질의 q_1, q_2가 다음과 같이 주어졌다고 가정하자.

d_3 = {(Information, 0.70), (Retrieval, 0.70)}
q_2 = Information AND Retrieval
q_3 = Information

곱하기 연산자가 *AND* 연산을 위해 사용될 때, 질의 q_2 와 q_3 에 대한 d_3 의 문서값은 각각 0.49 와 0.70 이다. 즉, q_2 와 d_3 사이의 유사도가 q_3 와 d_3 사이의 유사도보다 작다. 그러나

이것은 사람들의 순위 결정에 대한 행동 방식과 상반되는 결과로서, 곱하기 연산자가 단일 피연산자 의존 특성뿐 아니라 부정적 보상 특성도 지니고 있기 때문이다. 즉, $x \cdot y < MIN(x, y)$ $(x \in (0, 1))$.

퍼지 연산자들이 검색 효과에 미치는 영향을 분석함으로써, 높은 검색 효과를 제공할 수 있는 긍정적 보상 연산자라 불리는 이항 연산자 집합이 정의되었다.(Kim et al. 1993; Lee et al, 1992; Lee et al. 1993). 높은 검색 효과를 제공하는 긍정적 보상 연산자는 다음과 같이 정의된다.

$$p: [0, 1] \times [0, 1] \rightarrow [0, 1]$$

임의의 긍정적 보상 연산자 p 는 다음과 같은 특성을 갖는다.

특성 $\boldsymbol{p_1}$: $p(x, x) = x$; 즉, p 는 idempotent 이다.

특성 $\boldsymbol{p_2}$: $MIN(x, y) < p(x, y) < MAX(x,y), x \neq y$

긍정적 보상 연산자에 대한 위의 정의로부터 긍정적 보상 연산자는 단일 피연산자 의존 특성과 부정적 보상 특성 모두를 지니고 있지 않음을 알 수 있다. 따라서 긍정적 보상 연산자를 사용하는 확장 불리안 모델은 단일 피연산자 의존 문제와 부정적 보상 문제를 발생시키지 않는다. 퍼지 집합 이론에서 개발된 퍼지 연산자들로부터 다음과 같은 2 개의 긍정적 보상 연산자를 발견하였다.

$$(A_2) \quad (1-r) \cdot MIN(x, y) + r \cdot MIN(x, y), \quad 0 \le r \le 1$$
$$(A_{4,AND}) \quad r \cdot MIN(x, y) + (1-r) \cdot \frac{x+y}{2}, \quad 0 \le r \le 1$$
$$(A_{4,OR}) \quad r \cdot MAX(x, y) + (1-r) \cdot \frac{x+y}{2}, \quad 0 \le r \le 1$$

A_2 와 A_4 연산자는 각각 다른 사람에 의해 다른 시점에 개발되었을 지라도, 수학적으로 동일한 식임이 증명되었다(Lee et al. 1993).

15.2 이항 유연한 불리안 연산자

<그림 15.2>는 Waller-kraft, Paice, P-Norm, Infinite-One 모델로부터 유도된 이항 연산자를 보여준다. 이들은 다음과 같은 형태의 함수이다.

$$b: [0, 1] \times [0, 1] \rightarrow [0 , 1]$$

이항 연산자 b는 다음과 같은 특성을 갖는다.

$$F(d, t_1\ AND\ t_2) = (1-r)\cdot MIN(w_1, w_2) + r\cdot MAX(w_1, w_2), 0 \le r \le 0.5$$
$$F(d, t_1\ OR\ t_2) = (1-r)\cdot MIN(w_1, w_2) + r\cdot MAX(w_1, w_2), 0.5 \le r \le 1$$

(a) Waller-Kraft 모델

$$F(d, t_1\ AND\ t_2) = \frac{1}{1+r}\cdot MIN(w_1, w_2) + \frac{r}{1+r}\cdot MAX(w_1, w_2),$$
$$0 \le r \le 1 \text{ and } w_i\text{'s are considered in ascending order}$$
$$F(d, t_1\ OR\ t_n) = \frac{1}{1+r}\cdot MIN(w_1, w_2) + \frac{r}{1+r}\cdot MAX(w_1, w_2),$$
$$0 \le r \le 1 \text{ and } w_i\text{'s are considered in descending order}$$

(b) Paice 모델

$$F(d, t_1\ AND\ t_n) = 1 - \left(\frac{(1-w_1)^p + (1-w_2)^p}{2}\right)^{1/p}, \quad 1 \le p \le \infty$$
$$F(d, t_1\ OR\ t_n) = \left(\frac{{w_1}^p + {w_2}^p}{2}\right)^{1/p}, \quad 1 \le p \le \infty$$

(c) P-Norm 모델

$$F(d, t_1\ AND\ t_2) = r\cdot(1 - MAX(1-w_1, \cdots, 1-w_2)) + (1-r)\cdot\frac{w_1 + w_2}{2}, \quad 0 \le r \le 1$$
$$F(d, t_1\ OR\ t_n) = r\cdot MAX(1-w_1, \cdots, 1-w_2) + (1-r)\cdot\frac{w_1 + w_2}{2}, \quad 0 \le r \le 1$$

(d) Infinite-One 모델

<그림 15.2> AND와 OR 연산에 대한 이항 연산식

특성 b_1: $b(x, x) = x$; 즉, b는 idempotent이다.

특성 b_2: $x < x'$ 이면 $b(x, y) < b(x', y)$이고, $y < y'$ 이면 $b(x, y) < b(x, y')$이다. 즉, b는 각각의 피연산자에 대하여 절대 단조성(Strict Monotonicity)을 갖는다.

특성 b_3: b는 연속 함수이다.

특성 b_4: $b(x, y) = b(y, x)$; 즉, b는 교환 법칙을 만족한다.

특성 b_1은 3개의 질의 $q_1 = t_1\ AND\ t_1$, $q_2 = t_1\ OR\ t_1$, $q_3 = t_1$이 임의의 문서에 대해 동일한 문서값을 지정함을 의미한다. 특성 b_2에 의해 색인어 가중치의 증가로써 문서값의 증가를 보장할 수 있다. 특성 b_3의 연속성의 특성은 색인어 가중치의 미세한 증가가 문서값의 큰

변화를 발생시키는 상황을 막는다. 교환 법칙의 특성에 의해 질의 t_1 *AND* t_2 와 t_2 *AND* t_1 은 임의의 문서에 대해 동일한 문서값을 지정한다. 특성 b_1 부터 b_4 를 만족하는 연산자는 이항 유연한 불리안 연산자(Binary Soft Boolean Operator)로 정의된다.

정리 1: 임의의 연산자 θ 가 특성 b_1 과 b_2 를 만족한다면, 서로 다른 $x, y \in [0, 1]$에 대해 $MIN(x,y) < \theta(x,y) < MAX(x, y)$이다.

증명: 절대 단조성에 대해 다음의 식이 성립한다.

$\theta(MIN(x,y), MIN(x,y)) < \theta(x,y) < \theta(MAX(x,y), MAX(x,y))$

또한 idempotency 에 의한 다음의 두 식이 성립한다.

$\theta(MIN(x,y), MIN(x,y)) = MIN(x,y)$

$\theta(MAX(x,y), MAX(x,y)) = MAX(x,y)$

따라서, $MIN(x,y) < \theta(x,y) < MAX(x, y)$가 성립한다.

정리 1 로부터 이항 유연한 불리안 연산자가 긍정적 보상 연산자에 속함을 알 수 있다. 따라서, Waller-Kraft, Paice, P-Norm, Infinite-One 모델의 이항 연산자들은 높은 검색 효과를 제공하는 긍정적 보상 연산자이다. A_2 와 A_4 연산자도 특성 b_1 부터 b_4 를 만족하는 이항 유연한 불리안 연산자이다. A_2, A_4 와 Waller-Kraft, Paice, Infinite-One 모델의 이항 연산자는 수학적으로 동일한 식임을 쉽게 입증할 수 있다.

15.3 다항 연산자의 필요성

AND 연산은 단어들을 연결하여 구를 생성하고, *OR* 연산은 두개의 색인어들을 동의어로 간주한다. 따라서, fuzzy *AND* set *AND* theory 와 같이 연속적인 *AND* 연산이나, human *OR* people *OR* man 과 같이 연속적인 *OR* 연산이 자주 발생한다. 그러나, 이항 유연한 불리안 연산자는 결합 법칙을 만족하지 못한다.

정리 2: 임의의 연산자 θ가 특성 b_1 과 b_2 를 만족한다면, θ 는 결합 법칙을 만족하지 못한다.

증명: $x, y\,(x < y)$ 가 정리 2 를 만족한다고 가정하자. 정리 1 에 의해 다음의 식이 성립한다.

$\theta(x,y) > x$

첫번째 파라미터에 절대 단조성을 적용한다면 다음의 식을 얻을 수 있다.

$\theta(\theta(x,y), y) > \theta(x,y),$

그러나, θ 가 결합 법칙을 만족한다면 다음의 식이 성립한다.

$\theta(\theta(x,y), y) = \theta(x, \theta(y,y)) = \theta(x,y),$

따라서 θ가 결합 법칙을 만족한다는 가정은 모순을 발생시킨다.

결합 법칙을 만족하지 않은 이항 연산자는 의미적으로 동일한 질의에 대하여 서로 다른 문서값을 생성한다. 예를 들면, 문서 d가 $\{(t_1, 1), (t_2, 0.7), (t_3, 0.5)\}$로 표현되어 있고, 두 개의 질의 $q_1 = t_1\ AND\ (t_2\ AND\ t_3)$와 $q_2 = (t_1\ AND\ t_2)\ AND\ t_3$가 주어졌다고 가정하자. t_{4AND} (r=0.3) 연산자를 사용하는 확장 불리안 모델은 q_1에 대한 d의 문서값으로 0.721 을 생성하고, q_2에 대한 문서값으로 0.607 을 생성한다.

질의 $t_1\ OR\ t_2\ OR\ t_3$에 나타난 색인어 t_1, t_2, t_3는 같은 중요도로 사용자가 요구하는 정보를 표현하고 있다. 그러나 이항 유연한 불리안 연산자를 사용한다면, 질의 q 를 연산하는데 이러한 기본적인 가정을 유지할 수 없다. 질의 q를 연산하는 2 가지 방법 $(t_1\ OR\ t_2)\ OR\ t_3$과 $t_1\ OR\ (t_2\ OR\ t_3)$를 고려하자. $(t_1\ OR\ t_2)\ OR\ t_3$의 문서값은 색인어 t_1, t_2보다 t_3에 의존적이고, $t_1\ OR\ (t_2\ OR\ t_3)$의 문서값은 색인어 t_2, t_3보다 t_1에 의존적이다.

문서 $d = \{(t_1, w_1), (t_2, w_2), \cdots, (t_n, w_n)\}$와 질의 $q = t_1\ OR\ t_2\ OR\ \cdots\ OR\ t_n$를 가정하자. 질의를 왼쪽에서 오른쪽으로 Infinite-One 모델의 이항연산자(r=0)를 적용하면, 다음과 같은 문서 d의 문서값을 얻을 수 있다.

$$F(d,q) = \frac{w_1}{2^{n-1}} + \frac{w_2}{2^{n-1}} + \frac{w_3}{2^{n-2}} + \frac{w_4}{2^{n-3}} + \cdots + \frac{w_n}{2}$$
$$= \frac{1}{2}\left(\frac{w_1}{2^{n-2}} + \frac{w_2}{2^{n-2}} + \frac{w_3}{2^{n-3}} + \frac{w_4}{2^{n-4}} + \cdots + w_n \right)$$

위의 결과로부터 먼저 연산에 참여한 색인어와 나중에 연산에 참여한 색인어의 중요도가 다르게 취급되고 있음을 알 수 있다. 색인어 t_1과 t_n을 살펴보면, 질의에서는 같은 중요도로 질의를 표현하고 있다. 그러나 연산 결과에 있어서 색인어 t_1의 중요도를 1 이라고 할 때 t_n의 중요도는 2^{n-2} ($n \geq 2$)이다. 결론적으로 먼저 연산에 참여한 색인어는 나중에 연산에 참여한 색인어보다 훨씬 작은 중요도로 취급되며, n 의 값이 크면 클수록 이러한 왜곡의 정도는 더욱 커진다.

15.4 다항 유연한 불리안 연산자

Waller-Kraft, Paice, P-Norm, Infinite-One 모델은 다항 연산을 가능하게 함으로써 결합 법칙을 만족하지 못하는 문제를 회피하였다. 그러나, Waller-Kraft, Paice, P-Norm, Infinite-One 모델은 어떤 경우에 사람이 생각하는 것과 다르게 문서의 순위를 결정한다. 이 모델들은 질의에 주어진 모든 색인어들이 문서값의 계산에 있어서 동등하게 고려되어야 한다는 일반적 가정을 위반하며, 이것은 다음과 같은 불균등 중요성 문제(Unequal Importance Problem)를 발생시킨다.

불균등 중요성 문제 (유형 1): 문서값 계산에 있어서 Waller-Kraft 모델은 단지 2 개의 피연산자, 최소값과 최대값만을 고려한다. 예를 들면, 두개의 문서 d_1, d_2 와 질의 q_1 이 다음과 같이 주어졌다고 가정하자.

$$d_1 = \{(t_1, 0), (t_2, 0.9), (t_3, 0.9), \cdots, (t_{99}, 0.9), (t_{100}, 1)\}$$
$$d_2 = \{(t_1, 0), (t_2, 0.1), (t_3, 0.1), \cdots, (t_{99}, 0.1), (t_{100}, 1)\}$$
$$q_1 = t_1 \; AND \; t_2 \; AND \; \cdots \; AND \; t_{100}$$

Waller-Kraft 모델은 질의 q_1 에 대한 d_1, d_2 의 문서값으로 동일한 값을 지정한다. 그러나 대부분의 사람들은 질의 q_1 에 대하여 문서 d_1 의 만족도가 문서 d_2 의 만족도보다 큰 것으로 결정할 것이다.

불균등 중요성 문제 (유형 2): Paice 모델은 모든 피연산자들을 문서값에 반영함으로써, 유형 1 의 불균등 중요성 문제를 회피한다. 그러나 Paice 모델은 질의에 주어진 색인어들에 서로 다른 중요도를 부여한다. 예를 들면, 두개의 문서 d_3, d_4 와 질의 q_2 가 다음과 같이 주어졌다고 가정하자.

$$d_3 = \{(t_1, 0.1), (t_2, 0.3), (t_3, 0.3), (t_4, 0.3), (t_5, 0.3), (t_6, 0.3)\}$$
$$d_4 = \{(t_1, 0.1), (t_2, 0.7), (t_3, 0.3), (t_4, 0.3), (t_5, 0.3) , (t_6, 0.1)\}$$
$$q_2 = t_1 \; AND \; t_2 \; AND \; t_3 \; AND \; t_4 \; AND \; t_5 \; AND \; t_6$$

Paice 모델은(r=0.7)은 질의 q_2 에 대한 문서 d_3 의 값으로 0.2119 를 지정하고, 문서 d_4 의 문서값으로 0.2310 을 지정한다. 즉, t_6 의 색인어 가중치의 적은 감소가 t_2 의 색인어 가중치의 많은 증가에 대한 효과를 상쇄시키며, 이러한 특성은 문서값 계산에 대한 사람들의 행동 방식과 일치하지 않는다.

불균등 중요성 문제 (유형 3): Infinite-One 모델도 모든 피연산자들을 문서값에 반영함으로써, 유형 1 의 불균등 중요성 문제를 회피한다. 그러나 Infinite-One 모델이 생성한 문서값은 *AND* 연산에 대하여 최소값의 피연사자, *OR* 연산자에 대하여 최대값의 피연산자에 의해 보다 많은 영향을 받는다. 예를 들면, 두개의 문서 d_5, d_6 와 질의 q_3 가 다음과 같이 주어졌다고 가정하자.

$$d_5 = \{(t_1, 0), (t_2, 1), (t_3, 1), \ldots , (t_{100}, 1)\}$$
$$d_6 = \{(t_1, 0.4), (t_2, 0.6), (t_3, 0.6), \ldots, (t_{100}, 0.6)\}$$
$$q_3 = t_1 \; AND \; t_2 \; AND \; \ldots \; AND \; t_{100}$$

Infinite-One 모델(r=0.5)은 질의 q_3 에 대한 문서 d_5 와 d_6 의 문서값이 동일한 것으로 결정한다. 즉, t_1 의 색인어 가중치에 대한 0.4 의 증가가 t_2 부터 t_{100} 까지의 99 개의 가중치에 대한 0.4 의 감소와 동일한 효과를 갖는다. 그러나 사람들은 d_5 의 문서값이 보다 큰 것으로 결정할 것이다.

P-Norm 모델은 문서값 계산에 있어서 모든 피연산자들을 동일한 중요도로 고려하기 때문에, 불균등 중요성 문제를 발생시키지 않는다. P-Norm 모델의 다항 연산자는 다음과 같은 형태의 함수이다.

$$n : [0, 1] \times \cdots \times [0, 1] \rightarrow [0, 1]$$

다항 연산자 n 은 다음과 같은 특성을 지니며, 특성 n_1 부터 n_5 를 만족하는 연산자는 다항 유연한 연산자(N-ary Soft Boolean Operator)로 정의된다.

특성 n_1: $n(x, x, \cdots, x) = x$; 즉, n 은 idempotent 이다.

특성 n_2: n 은 각각의 피연산자에 대하여 절대 단조성(Strict Monotonicity)을 갖는다.

특성 n_3: n 은 연속 함수이다.

특성 n_4: $y_1, y_2, \cdots, y_n$이 $x_1, x_2, \cdots, x_n$에 대한 임의의 조합일 때, $n(x_1, x_2, \cdots, x_n) = n(y_1, y_2, \cdots, y_n)$; 즉, n은 symmetric 함수이다.

특성 n_5: n 은 모든 피연산자를 동일한 중요도로 고려한다.

긍정적 보상 연산자 A_2 와 A_4 는 이항 유연한 불리안 연산자에 속하기 때문에, 결합 법칙을 만족하지 않는다. 이들 연산자의 다항 연산식이 퍼지 집합 이론에 대한 연구 분야에서 다음과 같이 제시되었다(Zimmermann, 1987).

$$(A_{2.n}) \qquad r \cdot MAX(w_1, \cdots, w_n) + (1-r) \cdot MIN(w_1, \cdots, w_n), \quad 0 \le r \le 1$$

$$(A_{4.N.AND}) \quad r \cdot MIN(w_1, \cdots, w_n) + (1-r) \cdot \frac{w + \cdots + w_n}{n}, \quad 0 \le r \le 1$$

$$(A_{4.N.OR}) \quad r \cdot MIN(w_1, \cdots, w_n) + (1-r) \cdot \frac{w + \cdots + w_n}{n}, \quad 0 \le r \le 1$$

이항 연산자 A_2 와 A_4 는 수학적으로 동일한 식일 지라도, 다항 연산자 $A_{2.N}$ 과 $A_{4.N}$ 은 서로 다른 식이다. $A_{2.N}$은 Waller-Kraft 모델의 연산자 계산식과 동일하며, $A_{4.N}$는 Infinite-One 모델의 연산자 계산식과 동일하다.

제 16 장 확률 모델

검색어의 출현 빈도에 따른 확률 분포를 이용한 확률 검색의 개념은 Maron & Kuhns의 논문을 효시로 하여[Maro60], 1960년대 초기에 논의되기 시작하였다. 오늘날 주로 사용되고 있는 확률 검색 모델은 Robertson & Sparck Jones에 의해 정립되었으며[Robe76, Spar79a, Spar79b], 이진 독립 모델이라고 불리고 있다. 이외의 확률 검색 모델에 대한 많은 연구들이 수행되었다[Crof79, Crof81, Crof83, Fuhr89a, Fuhr89b].

이진 독립 모델은 문서 검색을 위한 준비로서 "적합성 정보"를 요구한다. 즉, 전체 문서 집합을 대상으로 질의에 대한 적합 문서 또는 부적합 문서 판정이 검색 이전에 수행되어 있음을 가정하고 있다. 이진 독립 모델에서 문서는 다음과 같은 벡터 X로 표현될 수 있으며, x_i는 i번째 색인어의 출현 유무를 나타내는 것으로 0 또는 1 값을 갖는다.

$$X = (x_1, x_2, x_3, \ldots, x_n)$$

벡터 X로 표현되는 문서가 특정한 질의에 대해 적합 문서로 검색되기 위해서는 이 문서의 적합 확률이 부적합 확률보다 높아야 한다. 질의에 대한 적합 문서들과 부적합 문서들을 각각 *rel*과 *nonrel*이라 한다면, 벡터 X로 표현된 문서가 주어진 질의에 대해 적합할 확률과 부적합할 확률은 $P(rel|X)$과 $P(nonrel|X)$로 정의된다. 따라서 벡터 X로 표현된 문서가 검색되기 위한 조건은 다음과 같이 표현될 수 있다.

$$P(rel|X) > P(nonrel|X)$$

$P(rel|X)$와 $P(nonrel|X)$의 값은 직접 산출하기 어려우므로 베이즈 정리를 이용하여 다음과 같이 변환한다.

$$P(rel \mid X) = \frac{P(X \mid rel) \bullet P(rel)}{P(X)}$$

$$P(nonrel \mid X) = \frac{P(X \mid nonrel) \bullet P(nonrel)}{P(X)}$$

위의 식을 $P(rel|X) > P(nonrel|X)$에 대입한다.

$$P(X \mid rel) \bullet P(rel) > P(X \mid nonrel) \bullet P(nonrel)$$

$$\frac{P(X \mid rel)}{P(X \mid nonrel)} > \frac{P(nonrel)}{P(rel)}$$

일반적으로 검색의 결과로서 단순히 적합 문서들의 집합을 사용자에게 제공하는 것보다

질의와 문서들 사이의 유사도를 기준으로 문서들을 내림차순으로 정렬하여 사용자에게 제공하는 것이 바람직하다. 위의 식을 이용하여 작성된 유사도 함수 $g(X)$는 다음과 같다.

$$g(X) = \frac{P(X \mid rel)}{P(X \mid nonrel)} - \frac{P(nonrel)}{P(rel)}$$

위 식에서 $P(X|rel)$은 적합 문서 집합에 벡터 X로 표현된 문서가 포함될 확률을 의미하고, $P(X|nonrel)$은 부적합 문서 집합에 벡터 X로 표현된 문서가 포함될 확률을 의미한다. 즉, 벡터 X로 표현된 문서가 적합 문서 집합에 많이 포함될수록, 그리고 부적합 문서 집합에 적게 포함될수록 유사도 함수 $g(X)$의 값도 증가한다. 한편, 위의 식에서 $P(rel)$과 $P(nonrel)$는 특정한 질의에 대해 일정한 값을 가지므로 문서들 사이의 순위에는 영향을 미치지 않는다. 따라서 유사도 함수 $g(X)$에서 두 번째 항목 $P(rel)/P(nonrel)$를 제거함으로써 다음과 같은 유사도 함수 $g'(X)$를 생성할 수 있다.

$$g'(X) = \frac{P(X \mid rel)}{P(X \mid nonrel)}$$

유사도 함수 $g'(X)$를 계산하기 위해서는 $P(X|rel)$과 $P(X|nonrel)$의 값을 추정해야 한다. 벡터 X의 변수 x_i가 통계학적으로 볼 때 상호 독립적이고, 즉 벡터 X에 포함된 색인어들이 독립적으로 출현한다고 가정하면, $P(X|rel)$과 $P(X|nonrel)$는 다음과 같은 식으로 정의될 수 있다.

$$P(X \mid rel) = P(x_1 \mid rel) \bullet P(x_2 \mid rel) \bullet \ldots \bullet P(x_n \mid rel)$$
$$P(X \mid nonrel) = P(x_1 \mid nonrel) \bullet P(x_2 \mid nonrel) \bullet \ldots \bullet P(x_n \mid nonrel)$$

한편, 색인어 x_i가 적합 문서에 존재할 확률을 p_i, 부적합 문서에 존재할 확률을 q_i라 하면, 이들은 다음과 같이 표현할 수 있다.

$$p_i = P(x_i = 1 \mid rel)$$
$$q_i = P(x_i = 1 \mid nonrel)$$

반면, 색인어 x_i가 적합 문서에 존재하지 않을 확률은 $1-p_i$이고, 부적합 문서에 존재하지 않을 확률은 $1-q_i$이며, 다음과 같은 식으로 표현될 수 있다.

$$1 - p_i = P(x_i = 0 \mid rel)$$
$$1 - q_i = P(x_i = 0 \mid nonrel)$$

따라서 $P(X|rel)$과 $P(X|nonrel)$는 다음과 같은 식으로 변환된다.

$$P(X \mid rel) = \prod_{i=1}^{n} {p_i}^{x_i} (1-p_i)^{1-x_i}$$

$$P(X \mid nonrel) = \prod_{i=1}^{n} {q_i}^{x_i} (1-q_i)^{1-x_i}$$

예를 들어, 다음과 같은 벡터 X가 있다고 가정하자.

$$X=\{(x_1, 0), (x_2, 1), (x_3, 1), (x_4, 0), (x_5, 0), (x_6, 1)\}$$

이때 적합 문서들에서 벡터 X로 표현된 문서가 발견될 확률 $P(X|rel)$과 부적합 문서들에서 벡터 X로 표현된 문서가 발견될 확률 $P(X|nonrel)$은 다음과 같이 계산된다.

$$P(X|rel) = (1\text{-}p_1) \bullet p_2 \bullet p_3 \bullet (1\text{-}p_4) \bullet (1\text{-}p_5) \bullet p_6$$
$$P(X|nonrel) = (1\text{-}q_1) \bullet q_2 \bullet q_3 \bullet (1\text{-}q_4) \bullet (1\text{-}q_5) \bullet q_6$$

위에서 정의된 식을 다음과 같이 유사도 함수 $g'(X)$에 대입한다.

$$g'(X) = \frac{\prod_{i=1}^{n} {p_i}^{x_i} \bullet (1-p_i)^{1-x_i}}{\prod_{i=1}^{n} {q_i}^{x_i} \bullet (1-q_i)^{1-x_i}} = \prod_{i=1}^{n} \frac{{p_i}^{x_i} \bullet \frac{1-p_i}{(1-p_i)^{x_i}}}{{q_i}^{x_i} \bullet \frac{1-q_i}{(1-q_i)^{x_i}}} = \prod_{i=1}^{n} \frac{{p_i}^{x_i} \bullet (1-q_i)^{x_i} \bullet (1-p_i)}{{q_i}^{x_i} \bullet (1-p_i)^{x_i} \bullet (1-q_i)}$$

유사도 함수 $g'(X)$는 다음과 같이 로그 함수를 적용함으로써 간략화되어 $g''(X)$으로 변환될 수 있으며, 로그 함수의 적용은 문서들 사이의 순위에는 영향을 미치지 않는다.

$$g''(X) = \log g'(X) = \log \prod_{i=1}^{n} \frac{{p_i}^{x_i} \bullet (1-q_i)^{x_i} \bullet (1-p_i)}{{q_i}^{x_i} \bullet (1-p_i)^{x_i} \bullet (1-q_i)}$$

$$= \sum_{i=1}^{n} \log \left\{ \frac{p_i \bullet (1-q_i)}{q_i \bullet (1-p_i)} \right\}^{x_i} + \sum_{i=1}^{n} \log \frac{1-p_i}{1-q_i}$$

$$= \sum_{i=1}^{n} x_i \log \frac{p_i \bullet (1-q_i)}{q_i \bullet (1-p_i)} + \sum_{i=1}^{n} \log \frac{1-p_i}{1-q_i}$$

위의 마지막 식에서 두 번째 항목은 특정한 질의에 대해 일정한 값을 가지므로 문서들 사이의 순위에는 영향을 미치지 않는다. 따라서 유사도 함수 $g''(X)$에서 두 번째 항목을 제거함으로써 최종적인 유사도 함수 $g'''(X)$를 생성할 수 있다.

$$g'''(X) = \sum_{i=1}^{n} x_i \log \frac{p_i \bullet (1-q_i)}{q_i \bullet (1-p_i)}$$

<표 16.1> N개의 문서로 구성된 문서 집합에서 색인어 x_i의 출현 빈도

	적합 문서	부적합 문서	
$x_i = 1$	r_i	$n_i - r_i$	n_i
$x_i = 0$	$R - r_i$	$N - R - n_i + r_i$	$N - n_i$
	R	$N - R$	N

유사도 함수 $g'''(X)$의 값을 계산하기 위해서는 각 색인어 x_i에 대한 p_i와 q_i를 추정해야 한다. 그러나 사용자가 처음으로 질의를 입력하는 최초 검색 시에는 p_i와 q_i의 값을 추정하는데 어려움이 있다. Croft & Harper는 p_i와 q_i의 값이 알려져 있지 않은 경우, p_i의 추정 값으로 0.5, q_i의 추정 값으로 n_i/N를 사용할 것을 제안하였다(Croft, 1979). 이때 N은 전체 문서 집합에 포함되어 있는 문서들의 수이며, n_i는 i번째 용어가 출현하는 문서들의 수이다. 따라서 적합성 정보가 없는 최초 검색 시에는 다음과 같은 유사도 함수가 사용될 수 있다.

$$\text{(최초 검색)} \qquad \sum_{i=1}^{n} x_i \log \frac{N - n_i}{n_i}$$

적합성 정보가 준비되어 있는 상태에서는 p_i와 q_i의 값은 실제 검색 대상 문서 집합에 포함된 적합 문서 및 부적합 문서들로부터 추정될 수 있다. 즉, 각 색인어 x_i를 포함하는 적합 문서와 부적합 문서의 수를 계산하여 <표 16.1>를 생성하면, 이 표로부터 p_i와 q_i의 값을 다음과 같이 추정할 수 있다. <표 16.1>에서 r_i는 색인어 x_i를 포함하는 적합 문서의 수를 나타내며, R은 전체 문서 집합에서 적합 문서들의 수, n_i는 색인어 x_i를 포함하는 문서들의 수, 그리고 N은 전체 문서 집합에 포함되어 있는 문서들의 수이다.

$$p_i = \frac{r_i}{R}, \quad q_i = \frac{n_i - r_i}{N - R}$$

그러나, R, r_i가 매우 적은 값을 부여받을 경우, 예를 들어 R=1, r_i=0일 경우, 유사도 함수에 포함된 로그 함수가 계산될 수 없다. 이러한 문제점을 해결하기 위해 일반적으로 p_i, q_i의 값은 다음과 같은 식에 의해 추정된다.

$$p_i' = \frac{r_i + 0.5}{R + 1}, \quad q_i' = \frac{n_i - r_i + 0.5}{N - R + 1}$$

따라서 유사도 함수 $g'''(X)$는 다음과 같은 식으로 변환된다.

$$g'''(X) = \sum_{i=1}^{n} x_i \log \frac{\dfrac{r_i + 0.5}{R - r_i + 0.5}}{\dfrac{n_i - r_i + 0.5}{N - R - n_i + r_i + 0.5}}$$

한편, p_i', q_i'의 계산식에 포함된 교정 상수 0.5는 특정 경우에 잘못된 결과를 생성할 수 있다고 비판되었으며, 이에 따라 p_i, q_i의 값을 추정하는 다음과 같은 식이 제안되었다[Robe86].

$$p_i'' = \frac{r_i + n_i / N}{R+1}, \quad q_i'' = \frac{n_i - r_i + n_i / N}{N-R+1}$$

제 17 장 질의 확장

오늘날 과학 기술 문명이 발전함에 따라 생산되는 정보의 양이 기하급수적으로 증가하고 있으며, 이러한 정보의 적절한 활용은 현대인의 성공의 여부를 결정하는 중요한 요소가 되고 있다. 이에 따라 방대한 양의 정보로부터 원하는 정보를 얻기 위해 다양한 정보 검색 시스템들이 활용되고 있다. 그러나, 많은 경우에 주어진 질의에 대한 정보 검색 시스템의 검색 결과가 부정확하기 때문에, 사용자들은 원하는 정보를 얻기 위해 많은 시간과 노력을 소비하고 있는 실정이다.

정보 검색 시스템들의 검색 결과가 부정확한 요인들 중의 하나는 사용자가 입력하는 질의가 불완전하기 때문이다. 즉, 일반적으로 사용자들은 광범위한 정보 요구를 지니고 정보 검색 시스템에 접근하여, 그 요구들을 질의로서 표현한다. 이러한 초기의 질의는 매우 임의적이며, 때때로 사용자들은 그들이 지니고 있는 문제점조차도 정확하게 표현할 줄 모른다. 예를 들어, 현재 인터넷 정보 검색 시스템에 입력되는 질의를 살펴보면, 많은 경우에 질의에 포함된 단어들의 수가 2 개를 넘지 못하고 있다. 이처럼 질의가 불완전할 경우, 정보 검색 시스템으로부터 양질의 검색 결과를 얻는 것은 매우 어려운 일이다.

지금까지 보다 높은 검색 효과를 제공하기 위하여 불완전한 초기 질의를 보완하는 다양한 방법들이 연구되어 왔다. “질의 확장(query expansion)”이라는 용어에 의해 지칭되는 이러한 방법들은 초기 질의와 관련이 높은 단어들을 선정한다. 정보 검색 시스템은 선정된 단어들을 모두 질의에 추가하거나, 또는 선정된 검색어들 중에서 사용자에 의해 선택된 일부 단어만을 질의에 추가함으로써 초기 질의를 보완할 수 있다. 본 절에서는 지금까지 연구된 다양한 질의 확장 방법들에 대하여 기술한다.

17.1 수작업 시소러스

시소러스는 분류 구조(classification structure), 제한된 어휘 사전(controlled vocabulary), 순서화 체계(ordering system)라로도 불리며, 개념을 표시하는 노드와 개념들 사이의 관계를 표시하는 링크로 구성된다. 개념들 사이의 관계로는 광의어(broader term), 협의어(narrower term)와 같은 ‘ is-a’ 관계와 동의어(synonym), 관련어(related term) 등이 있다. 일반적으로 개념은 단일 단어로서 표현되며, 단일 단어로서 개념을 표현할 수 없을 경우에는 구가 대신 사용된다.

1960, 1970 년대에 개발된 많은 정보 검색 시스템들은 시소러스를 기반으로 문서들을 색인하고, 검색을 수행하였다. 즉, 시소러스에 포함된 개념들만으로 문서들을 색인하였으며,

또한 시소러스의 개념들을 이용하여 질의를 작성하였다. 이러한 색인 작업은 컴퓨터를 이용하여 자동으로 수행하기 매우 어려운 작업이기 때문에, 대부분의 경우 사람에 의해 수작업으로 수행되었다. 또한, 시소러스를 이용한 색인은 문서에 포함되지 않은 단어 또는 구를 색인어로 선정할 수 있기 때문에, 문서에서 사용된 단어에 관계없이 문서의 주제를 기술하는 수단을 제공한다.

지금까지 다양한 시소러스들이 문서들의 주제를 기술하기 위해 사용되어 왔다. Medical Subject Headings(MeSH)는 MEDLINE 시스템이 사용하는 시소러스로서, 9 단계의 계층적 트리 형태를 갖는 15,000 여개의 개념들로 구성되어 있다. 동의어까지 고려한다면, MeSH 는 전체적으로 100,000 개 이상의 색인어들을 포함하고 있다. 또 다른 시소러스로서 Association of Computing Machinary(ACM)의 간행물들을 색인하기 위한 Computing Reviews Classification Scheme(CRCS)이 있다. CRCS 는 5 단계의 계층적 트리 형태를 갖는 1,000 여개의 개념들로 구성되어 있으며, 개념들 사이의 관계로서 'is-a' 관계만을 사용한다.

시소러스의 개념들을 사용하여 문서들이 색인되고 질의가 작성되었을 경우, 시소러스에 포함된 개념들 사이의 관계를 이용하여 질의 확장을 수행할 수 있다. 그러나, Salton & Lesk 가 수행한 시소러스를 이용한 질의 확장 실험은 미미한 정도의 검색 효과의 향상을 보여주었으며, 또한 대규모의 TREC 문서 집단에 대하여 WordNet 을 이용한 Voorhees 의 연구는 오히려 부정적인 결과를 보여주었다. 즉, 많은 언어학자 및 분야별 전문가들에 의해 수작업으로 작성되는 시소러스는 개발에 많은 비용이 소비됨에도 불구하고, 시소러스를 이용한 질의 확장은 아직까지 일관된 검색 효과의 향상을 보여주지 못 하고 있다.

17.2 색인어 클러스터링

색인어들이 동일한 문서에 출현하는 빈도, 즉 동시 출현 빈도(co-occurrence frequency)를 이용하여 색인어-색인어 유사도 행렬을 구성하고, 이러한 행렬에 클러스터링 알고리즘을 적용하여 유사성이 높은 색인어들로 구성된 클러스터들을 생성한다. 하나의 클러스터에 포함된 다수의 색인어들은 동의어로 간주되며, 질의에 포함된 검색어의 동의어들로서 질의를 확장한다.

17.3 색인어-색인어 유사도 행렬

위에서 설명된 질의 확장 방법들에서처럼 질의에 포함된 각각의 검색어에 대한 동의어들로서 질의를 확장할 경우, 초기 질의 “ power plant(발전소)”에 대하여 “ tree(나무)”나 “ crop(농작물)”과 같은 단어들이 확장될 검색어로서 선정될 수 있다. 이처럼 초기 질의와 관계가 없는 “ tree"와 ” crop"가 확장될 용어들로서 선정될 수 있는 이유는 “ plant(공장, 식물)”라는 용어의 의미가 모호하기 때문이다. 이러한 문제를 해결하기 위하여 Qiu & Frei 는 질의에 포함된 각각의 용어에 대하여 확장될 용어들을 선정하는 대신에 질의 전체와 유사도가 높은 용어들을 선정하여 질의를 확장하였다. 즉, 용어-용어 유사도 행렬을 기반으로 질의에 포함된 모든 용어들과 유사도가 높은 용어들을 확장될 용어로서 선정하였다.

17.4 연관 시소러스

문서들의 수가 많을 경우 문서들에 포함된 단어들의 수도 증가하기 때문에 색인어-색인어 유사도 행렬의 생성이 매우 어려워진다. 이러한 문제점을 해결하기 위해 Jing & Croft 는 문서들을 분석하여 개념와 연관어 2 개의 필드들로 구성된 가상 문서들을 생성하였다. 그리고, 초기 질의에 대하여 연관어 필드들을 검색하여, 상위에 검색된 가상 문서들의 개념 필드에 포함된 개념을 초기 질의의 확장을 위해 사용하였다.

제 18 장 데이터 퓨젼

질의와 문서를 위한 다양한 표현 방법들이 정보 검색 분야에서 제안되어 왔고, 이에 대응하는 많은 검색 기법들이 검색 효과의 향상을 위해 개발되어 왔다. 최근 보다 높은 검색 효과를 얻기 위하여 질의와 문서에 대해 다양한 표현 방법을 사용하거나 여러 가지의 검색 기법을 사용하여 서로 다른 집합의 문서들을 검색하고, 그 결과들을 결합하는 연구가 진행되어 왔다. 이와 관련된 연구들은 “데이타 퓨젼(data fusion)”이라는 용어에 의해 지칭되고 있다.

McGill, et al.(1979)은 서로 다른 사용자가 문서를 검색할 때, 또는 한명의 사용자가 문서를 검색할 경우에도 통제어의 사용여부에 따라, 동일한 정보 요구에 대해 검색되는 문서 집합들 사이에 중복이 적다는 것을 발견하였다. 이와 같은 현상은 동일한 정보 요구로부터 생성된 서로 다른 질의 표현들이 서로 다른 문서들을 검색함을 의미한다. Katzer, et al.(1982)은 서로 다른 질의 표현보다는 서로 다른 문서 표현이 검색되는 문서에 미치는 영향을 조사하였고, 서로 다른 문서 표현들이 서로 다른 문서 집합을 검색한다는 동일한 현상을 발견하였다. 이러한 결과는 결합된 검색 실행이 각각의 실행보다 보다 많은 적합 문서를 검색할 수 있음을 암시한다.

Saracevic & Kantor(1988)는 여러 전문가에게 동일한 정보 요구에 대하여 불리안 질의를 생성하도록 요청하고, 이들이 생성한 다양한 질의에 대한 검색을 수행하였다. 이들은 이 실험을 통하여 서로 다른 질의 표현이 서로 다른 문서들을 검색함을 재확인하였다. 또한, 검색된 문서가 보다 많은 검색 실행에 의해 검색될수록 적합 문서일 확률이 증가함을 발견하였다. 따라서, 검색 실행들을 결합하는 결합 방법이 보다 많은 검색 실행들에 의해 검색된 문서를 선호하도록 설계된다면, 결합된 검색 실행은 질의와 문서 사이의 유사도 계산을 보다 정확하게 수행함으로써 보다 높은 정확률을 제공할 것이다.

Turtle & Croft(1991)는 서로 다른 문서 또는 질의 표현을 확률적 관점에서 결합할 수 있는 추론 네트워크 모델을 개발하였다. Turtle & Croft 는 이 모델을 기반으로 하여 정보 검색 시스템 INQUERY 를 개발하였고, 다중 증거의 결합이 검색 효과를 향상시킬 수 있음을 보였다. Fox & Shaw(1994)는 다양한 다중 증거를 결합하기 위한 방법에 대한 연구를 수행하였으며, 다중 증거를 사용한 검색 실행이 단일 증거만을 사용한 각각의 검색 실행보다 높은 검색 효과를 제공함을 발견하였다. Belkin, et al.(1993)은 서로 다른 부울 질의 표현의 연속적인 결합이 검색 효과의 연속적인 향상으로 나타남을 보였다.

본 장에서는 1 절과 2 절에서 서로 다른 특성의 가중치 기법들의 결합과 서로 다른 특성의 적합성 피드백 방법들의 결합이 보다 높은 검색 효과를 제공함에 대하여 설명한다. 3 절에서는 데이터 퓨젼이 보다 높은 검색 효과를 제공하는 이유에 대하여 살펴보고, 4 절에서는

결합 방법을 고찰하며, 5 절에서는 유사도 결합과 순위 결합의 차이점을 분석한다.

18.1 다중 가중치 기법의 결합

10.3 절에서 설명된 바와 같이 서로 다른 부류에 속하는 가중치 기법들은 서로 다른 형태의 문서들을 검색할 수 있다. 본 연구에서는 WSJ.D2 자료 집합을 대상으로 서로 다른 가중치 기법을 사용하는 다양한 검색 실행을 수행하여, 그 결과를 결합한다. 검색 실행과 결합된 검색 실행의 결과로서 상위 순위 200 개 문서들이 검색되었으며, 사용된 검색 실행의 결합 방법은 다음과 같다(Lee 1995). 첫째, 각각의 검색 실행이 생성한 문서값을 그 검색 실행이 생성한 최대 문서값으로 나눈다. 둘째, 결합된 검색 실행에서 각각의 문서에 대한 문서값은 결합에 참가하는 검색 실행들이 그 문서에 대해 생성한 문서값들의 합이다.

서로 다른 가중치 기법을 사용하는 6 개의 검색 실행을 수행하고, 이들의 쌍들을 결합하였으며, <표 18.1>은 그 결과를 보여준다. % 변화는 각 쌍에서 높은 검색 효과를 보이는 검색 실행에 대하여 계산되었다. <표 10.2>와 <표 18.1>의 비교로부터 결합되는 검색 실행이 유사한 검색 효과를 제공하고 상이한 문서들을 검색할수록, 그들의 결합으로부터 얻을 수 있는 검색 효과의 향상이 큼을 알 수 있다. 또한 <표 18.1>은 부류 C 에 속하는 가중치 기법과 다른 부류에 속하는 가중치 기법 사이의 결합에 대해서만 의미 있는 검색 효과의 향상을 얻을 수 있음을 보여준다. 즉, 서로 다른 2 개의 검색 실행의 결합 시에, 다음과 같은 조건 하에서 의미 있는 검색 효과의 향상을 얻을 수 있다.

- 결합되는 검색 실행은 유사한 수준의 검색 효과를 제공한다.
- 하나의 검색 실행의 가중치 기법은 코사인 정규화를 수행하고, 다른 하나의 검색 실행의 가중치 기법은 코사인 정규화를 수행하지 않는다.

<표 18.1> 서로 다른 가중치 기법을 사용하는 검색 실행의 결합

	lnc•ltc (C) 0.3284	*anc•ltc* (C) 0.3034	*anc•ltc* (C) 0.2661	*ltn•ntc* (N) 0.3228	*ann•ntc* (M) 0.3002
anc•ltc (C) 0.3034	0.3205 (-2.4%)	–	–	–	–
lnn•ntc (N) 0.2661	0.3378 (+2.9%)	0.3433 (+13.1%)	–	–	–
ltn•ntc (N) 0.3028	0.3464 (+5.5%)	0.3517 (+15.9%)	0.2887 (-4.7%)	–	–
ann•ntc (M) 0.3002	0.3575 (+8.9%)	0.3473 (+14.5%)	0.2982 (-0.7%)	0.3162 (+4.4%)	–
atn•ntc (M) 0.3198	0.3627 (+10.4%)	0.3451 (+7.9%)	0.3130 (-2.2%)	0.3217 (+0.6%)	0.3148 (-1.6%)

18.2 다중 적합성 피드백 방법의 결합

본 절에서는 주어진 정보 요구에 대한 다중 질의 벡터를 생성하기 위해 정보 검색 분야에서 개발된 5 개의 다양한 적합성 피드백 방법들을 이용하였다. 이들 중 2 개는 벡터 수정에 의한 방법이고 3 개는 확률 검색 모델에 근거한 방법이며, 다음과 같이 요약될 수 있다.

Rocchio 피드백 질의 벡터 Q_{new}는 초기 질의 벡터와 적합 및 부적합 문서 벡터들의 벡터합에 의해 생성된다(Rocchio 1971).

$$Q_{new} = \alpha \bullet Q_{old} + \beta \bullet \frac{1}{n_{rel}} \sum_{r=1}^{n_{rel}} D_r - \gamma \bullet \frac{1}{n_{nonrel}} \sum_{n=1}^{n_{nonrel}} D_n$$

α ,β , γ 상수

D_r 적합 문서 d_r에 대한 벡터

D_n 부적합 문서 d_n에 대한 벡터

n_{rel} 적합 문서들의 수

n_{nonrel} 부적합 문서들의 수

Ide Ide 공식은 Rocchio 공식을 수정함으로써 생성되었다. 즉, 적합 문서들의 수에 의한 정규화를 수행하지 않고, 최상위 순위 부적합 문서만을 피드백에 사용하였다(Ide 1971).

$$Q_{new} = \alpha \bullet Q_{old} + \beta \bullet \sum_{r=1}^{n_{rel}} D_r - \gamma \bullet T_{nonrel}$$

T_{nonrel} 최상위 순위 부적합 문서에 대한 벡터

Pr_cl Pr_cl 공식은 전형적인 확률 피드백 공식으로 확률 검색 모델에 근거하여 개발되었다 (Croft & Harper 1979).

$$w_{qi}' = \log \frac{p_i \bullet (1-q_i)}{q_i \bullet (1-p_i)}$$

$$p_i = \frac{r_i + 0.5}{R+1}, \quad q_i = \frac{n_i - r_i + 0.5}{N-R+1}$$

r_i 색인어 t_i를 포함하는 적합 문서들의 수

n_i 컬렉션내 색인어 t_i를 포함하는 문서들의 수

R 적합 문서들의 수

N 컬렉션내 문서들의 수

Pr_adj Pr_adj 공식은 Pr_cl 공식을 수정함으로써 개발되었다. 즉, 상수 0.5 를 n_i/N 로 대치하였다 (Robertson 1986).

$$w_{qi}' = \log \frac{p_i \bullet (1-q_i)}{q_i \bullet (1-p_i)}$$

$$p_i = \frac{r_i + n_i / N}{R+1}, \quad q_i = \frac{n_i - r_i + n_i / N}{N-R+1}$$

S_rpi S_rpi 공식은 Furh 의 RPI 공식을 비선형 함수로 단순화시킨 식이다(Fuhr & Buckely 1991).

$$w_{qi}' = \log \frac{p_i \bullet (1-q_i)}{q_i \bullet (1-p_i)}$$

$$p_i = \sum_{r=1}^{n_{rel}} \frac{w_{ri}}{n_{rel}}, \quad q_i = \sum_{n=1}^{n_{nonrel}} \frac{w_{ni}}{n_{nonrel}}$$

w_{ri} 적합 문서 d_r 내에서 색인어 t_i의 가중치

w_{ni} 부적합 문서 d_n 내에서 색인어 t_i의 가중치

n_{rel} 적합 문서들의 수

n_{nonrel} 부적합 문서들의 수

본 연구에서는 하나의 사용자 질의에 대하여 다중의 질의 표현을 자동으로 생성하는 방법을 제안한다. 이에 사용되는 기본적인 생각은 서로 다른 적합성 피드백 방법들은 서로 다른 특성을 지니기 때문에, 동일한 초기 질의와 동일한 적합성 정보에 대하여 서로 다른 질

의 벡터를 생성하며, 이러한 질의 벡터들은 서로 다른 문서들을 검색한다는 것이다. 이러한 생각에 대한 진위를 확인하기 위하여 앞 절에서 서술한 5 개의 피드백 방법들을 이용하여 다음과 같은 실험을 수행하였다.

<표 18.2> 초기 질의와 피드백 질의에 대한 11-포인트 평균 정확률 (TREC D1 & D2; 50 개 질의에 대한 평균)

Initial	Ide	Rocchio	Pr_cl	Pr_adj	S_rpi
0.2893	0.3523 (+ 21.8%)	0.3482 (+ 20.4%)	0.3361 (+ 16.2%)	0.3378 (+ 16.8%)	0.3301 (+ 14.1%)

<표 18.3> 초기 질의 벡터와 피드백 질의 벡터 사이의 유사도 (TREC D1 & D2; 50 개 질의에 대한 평균)

	Ide	Rocchio	Pr_cl	Pr_adj	S_rpi
Initial	0.5803	0.9566	0.2742	0.2522	0.2387

<표 18.4> 피드백 질의 벡터들 사이의 유사도 (TREC D1 & D2; 50 개 질의에 대한 평균)

	Ide	Rocchio	Pr_cl	Pr_adj
Rocchio	0.7900 (4)			
Pr_cl	0.6746 (6)	0.4456 (8)		
Pr_adj	0.6595 (5)	0.4239 (9)	0.9856 (1)	
S_rpi	0.6470 (7)	0.4091 (10)	0.9725 (2)	0.9403 (3)

1. 주어진 질의에 대한 초기 질의 벡터를 생성한다.
2. 초기 검색을 수행하고 상위 30 개 문서를 적합문서로 가정한다.
3. 다양한 적합성 피드백 방법들을 사용하여 피드백 질의 벡터들을 생성한다.
4. 피드백 질의 벡터들에 의한 피드백 검색을 수행한다.

문서 집합으로 TREC D1 & D2, 질의 집합으로 TREC Q151-Q200 50 개 질의를 사용하였고, 각각의 질의에 대해 상위 1000 개의 문서들을 검색하였으며, 11-포인트 평균 정확률을 사용하여 검색 성능을 평가하였다. <표 18.2>는 그 결과를 보여주며, 피드백 검색이 초기 검색보다 높은 검색 효과를 제공함을 알 수 있다.

한편, 질의 벡터들 사이의 유사도를 계산함으로써 초기 질의와 피드백 질의들이 어느 정도 상이한가를 알 수 있다. 질의 벡터들 사이의 유사도는 질의와 문서 사이의 유사도를 계산할 때와 동일하게 질의 벡터들 사이의 내적으로 계산하였으며, 질의 벡터를 정규화하기 위해 코사인 정규화 방법을 사용하였다. 유사도는 0 과 1 사이의 값을 지니며, 유사도 1 은 두 개의 질의 벡터가 동일함을 의미한다. <표 18.3>은 초기 질의 벡터와 확장된 질의 벡터들 사이의 유사도를 보여주며, <표 18.4>는 확장된 질의 벡터들 사이의 유사도를 보여준다. <표 18.3>과 <표 18.4>로부터 서로 다른 적합성 피드백 방법들이 초기 질의를 다르게 수정함으로 서로 다른 질의 벡터들을 생성함을 알 수 있다.

서로 다른 질의 표현들이 서로 다른 문서 집합들을 검색함은 정보 검색 분야에서 널리 알려져 있다. 지금까지 본 연구에서는 하나의 사용자 질의에 대하여 서로 다른 다중 질의 벡터를 생성하는 방법을 제안하였다. 다음에서는 생성된 다중 질의 벡터에 의해 검색된 결과를 분석하여, 이 질의 벡터들이 서로 다른 문서들을 검색함을 입증한다. 즉, 검색 결과가 어느 정도 상이한가를 분석하기 위하여 Spearman 연관 계수와 공통으로 검색된 문서 수를 계산한다.

Spearman 연관 계수는 두 개의 순위 리스트가 얼마나 유사한가를 나타낸다. k 개의 객체 $e_1, ..., e_k$가 있을 때, 이에 대한 서로 다른 순위 $r_1, ..., r_k$와 $r_1', ..., r_k'$ 사이의 Spearman 연관 계수 ρ는 다음과 같이 정의된다 (Lee et al., 1991).

$$\rho = 1 - 6 \times \frac{\sum_{i=1}^{k} (r_i' - r_i)^2}{k(k^2 - 1)}$$

Spearman 연관 계수는 두 순위가 일치할 때 1 이고, 서로 관련이 없는 순위일 때 0 이며, 서로 정반대의 순위일 때 -1 의 값을 갖는다.

TREC D1 & D2 는 740,000 건 이상의 문서를 포함하고 있기 때문에, 모든 문서들의 문서값을 계산하여 문서들에 순위를 부여하고 Spearman 연관 계수를 계산하는 것은 많은 노력이 요구한다. 따라서 본 연구에서는 적합 문서만을 대상으로 문서값을 계산하여 Spearman 연관 계수를 계산하였다. <표 18.5>는 5 개 피드백 질의에 의해 생성된 문서들의 순위에 대한 Spearman 연관 계수를 보여준다. 이 표로부터 서로 다른 적합성 피드백 방법들에 의해 확장된 질의 벡터들이 서로 다른 검색 결과를 생성함을 알 수 있다.

<표 18.5> 피드백 질의에 의해 생성된 순위출력 사이의 Spearman 연관 계수 (TREC D1 & D2; 적합 문서만을 대상으로 순위 계산)

	Ide	Rocchio	Pr_cl	Pr_adj
Rocchio	0.9355 (4)			
Pr_cl	0.8947 (6)	0.8029 (9)		
Pr_adj	0.9019 (5)	0.8114 (8)	0.9974 (1)	
S_rpi	0.8890 (7)	0.7951 (10)	0.9898 (2)	0.9891 (3)

<표 18.6> 피드백 질의 벡터들에 의해 검색된 공통 문서의 수 (TREC D1 & D2; 50 개 질의에 대한 평균; 상위 1000 개의 문서를 검색)

	Ide	Rocchio	Pr_cl	Pr_adj
Rocchio	737.58 (4)			
Pr_cl	655.32 (6)	530.64 (9)		
Pr_adj	668.54 (5)	539.54 (8)	985.50 (1)	
S_rpi	642.36 (7)	516.72 (10)	881.06 (2)	880.82 (3)

피드백 질의 벡터들이 서로 상이할수록 이들에 의해 공통적으로 검색되는 문서들의 수가 적어진다. 즉, 피드백 질의 사이의 유사도가 크면 클수록, 이들에 의해 생성된 검색 결과 사이의 연관 계수가 커진다. <표 18.4>와 <표 18.5>에서 괄호 안의 숫자는 각각 질의 사이의 유사도와 검색 결과 사이의 Spearman 연관 계수를 내림차순으로 순위를 부여한 것이다. 이들로부터 <표 18.4>에 부여된 순위와 <표 18.5>에 부여된 순위들이 밀접하게 연관되어 있음을 알 수 있다.

본 연구에서는 피드백 질의 벡터에 의한 검색 결과가 어느 정도 상이한가를 분석하기 위하여 서로 다른 피드백 질의 벡터에 의해 공통적으로 검색된 문서들의 수를 조사하였다. 이러한 방법은 Spearman 연관 계수의 계산에 비하여 비교적 간단하게 검색 결과들이 상이한 정도를 알 수 있다. <표 18.6>은 피드백 질의 벡터에 의해 공통적으로 검색된 문서들의 수를 보여준다. <표 18.6>에서 괄호 안의 숫자는 공통적으로 검색되는 문서들의 수를 내림차순으로 순위를 부여한 것이다. 이로부터 공통적으로 검색된 문서의 수가 Spearman 연관 계수의 계산에 의한 검색 결과 상이도를 대치할 수 있음을 알 수 있다.

앞에서 서로 다른 적합성 피드백 방법들에 의해 확장된 질의 벡터들이 서로 다른 문서

들을 검색함을 입증하였다. 두 개의 검색 실행들이 서로 다른 문서 집합들을 검색할 경우, 이들의 검색 결과들을 결합함으로써 보다 높은 검색 효과를 얻을 수 있음이 정보 검색 분야에서 알려져 왔다. 다음에서는 피드백 질의 벡터들에 의해 검색된 결과들을 결합함으로써 보다 높은 검색 효과를 얻을 수 있는가를 살펴본다. 검색 실행의 결합을 위하여 16.1 절의 다중 가중치 기법의 결합에서와 동일한 방법을 사용하였다.

<표 18.7> 결합된 검색 결과들의 11-포인트 평균 정확률 (TREC D1 & D2; 50 개 질의에 대한 평균; 11-포인트 평균 정확률 0.2893 을 제공하는 초기 검색 결과를 기준으로 검색 효과 향상의 % 변화를 계산)

	Ide 0.3523 (+ 21.8%)	Rocchio 0.3482 (+ 20.4%)	Pr_cl 0.3361 (+ 16.2%)	Pr_arj 0.3378 (+ 16.8%)
Rocchio 0.3482 (+ 20.4%)	0.3529 (+ 22.0%)			
Pr_cl 0.3361 (+ 16.2%)	0.3602 (+ 24.5%)	0.3659 (+ 26.5%)		
Pr_arj 0.3378 (+ 16.8%)	0.3604 (+ 24.6%)	0.3666 (+ 26.7%)	0.3376 (+ 16.7%)	
S_rpi 0.3301 (+ 14.1%)	0.3578 (+ 23.7%)	0.3655 (+ 26.3%)	0.3335 (+ 15.3%)	0.3342 (+ 15.5%)

<표 18.7>은 앞에서 설명된 5 개의 피드백 질의에 의해 생성된 검색 결과들의 쌍을 결합한 결과에 대한 검색 효과를 보여주며, 여기에서 % 변화는 초기 검색 결과의 검색 효과 0.2893 에 대한 변화율이다. 이 표로부터 결합된 검색 결과들이 결합 이전의 피드백 질의의 검색 결과보다 높은 검색 효과를 제공함을 알 수 있다. 이러한 결과는 서로 다른 적합성 피드백 방법들에 의해 생성된 질의 벡터의 검색 결과를 결합함으로써 보다 높은 검색 효과를 얻을 수 있음을 입증한다.

두 개의 다른 검색 실행이 서로 다른 문서들을 많이 검색할수록, 이들의 검색 결과를 결합함으로써 보다 높은 검색 효과의 향상을 얻을 수 있다. 이러한 사실을 확인하기 위하여, 검색 결과들의 결합에 의해 얻을 수 있는 검색 효과 향상의 정도를 계산할 필요가 있다. 이에 따라 결합에 참여한 검색 결과 중에서 높은 검색 효과를 제공하는 검색 결과를 기준으로, 검색 효과 향상의 % 변화를 계산하였으며, <표 18.8>은 그 결과를 보여준다. <표 18.8>에 나타난 결과는 결합되는 검색 결과의 상이도가 검색 효과의 상승도와 반드시 비례하지 않는다는 것을 보여준다. 예를 들면, Ide 와 Pr_adj 사이의 Spearman 연관 계수는 0.9019 이고,

검색 효과의 향상은 +2.3%이다. 그러나 Ide 와 S_rpi 사이의 Spearman 연관 계수는 0.8890 으로 0.9019 보다 다소 적으나, 검색 결과 결합시 +1.6%의 검색 효과 향상을 얻을 수 있다.

결합된 검색 결과의 검색 효과는 결합에 참여한 검색 결과의 연관 관계뿐 아니라, 결합에 참여한 검색 결과의 검색 효과에 의해 영향을 받는다. 따라서 Ide 와 S_rpi 결합이 Ide 와 Pr_adj 결합보다 검색 효과의 향상이 적은 이유는 S_rpi 의 검색 효과 0.3301 가 Pr_adj 의 검색 효과 0.3378 보다 작기 때문일 것이다. 이를 확인하기 위하여 <표 18.9>에서 결합에 참여한 검색 결과들이 제공하는 검색 효과의 평균을 기준으로, 검색 효과 향상의 % 변화를 계산하였다. <표 18.9>의 결과와 <표 18.5>의 결과는 거의 일치하는 것을 알 수 있으며, 두 개의 검색 결과들이 적은 연관 계수를 가질수록 그들의 결합으로부터 보다 많은 검색 효과의 향상을 얻을 수 있음을 알 수 있다.

마지막으로 본 연구에서는 3 개 이상의 검색 결과를 결합했을 때 검색 효과의 향상에 대하여 살펴보았다. 즉, 5 개의 검색 결과들을 2 단계에서는 2 개씩 결합하여 10 개, 3 단계에서는 3 개씩 결합하여 10 개, 4 단계에서는 4 개씩 결합하여 5 개, 5 단계에서는 5 개 모두를 결합하여 1 개의 결합된 검색 결과를 생성한 후, 각 단계에서 최대 검색 효과와 검색 효과들의 평균을 조사하였다. <표 18.10>은 그 결과를 보여주며, 이 표로부터 검색 효과들의 평균은 단계가 증가함에 따라 단순하게 증가함을 알 수 있다. 그러나 최대 검색 효과는 2 단계와 3 단계가 나머지 단계보다 우수하다. 이러한 결과는 Belkin et al.(1995)이 다중 불리안 질의의 결합에서 얻은 결과와 일치한다.

<표 18.8> 결합된 검색 결과들의 11-포인트 평균 정확률 (TREC D1 & D2; 50 개 질의에 대한 평균; 결합에 참여한 검색 결과 중에서 높은 검색 효과를 제공하는 검색 결과를 기준으로 검색 효과 향상의 % 변화를 계산)

	Ide 0.3523	Rocchio 0.3482	Pr_cl 0.3361	Pr_arj 0.3378
Rocchio 0.3482	0.3529 (+0.1%)			
Pr_cl 0.3361	0.3602 (+2.2%)	0.3659 (+5.1%)		
Pr_arj 0.3378	0.3604 (+2.3%)	0.3666 (+5.3%)	0.3376 (-0.1%)	
S_rpi 0.3301	0.3578 (+1.6%)	0.3655 (+5.0%)	0.3335 (-0.7%)	0.3342 (-1.1%)

<표 18.9> 결합된 검색 결과들의 11-포인트 평균 정확률 (TREC D1 & D2; 50 개 질의에 대한 평균; 결합에 참여한 검색 결과들이 제공하는 검색 효과의 평균을 기준으로 검색 효과 향상의 % 변화를 계산)

	Ide 0.3523	Rocchio 0.3482	Pr_cl 0.3361	Pr_arj 0.3378
Rocchio 0.3482	0.3529 (+0.8%)			
Pr_cl 0.3361	0.3602 (+4.6%)	0.3659 (+6.9%)		
Pr_arj 0.3378	0.3604 (+4.4%)	0.3666 (+6.9%)	0.3376 (+0.2%)	
S_rpi 0.3301	0.3578 (+4.9%)	0.3655 (+7.8%)	0.3335 (+0.1%)	0.3342 (+0.1%)

<표 18.10> 3 개 이상의 검색 결과 결합시의 검색 효과 (TREC D1 & D2; 50 개 질의에 대한 평균)

	Initial	1-way	2-way	3-way	4-way	5-way
Average	0.2893	0.3409 (+17.8%)	0.3535 (+22.2%)	0.3565 (+23.2%)	0.3587 (+24.0%)	0.3582 (+23.8)
best	0.2893	0.3523 (+21.8%)	0.3666 (+26.7%)	0.3682 (+27.3%)	0.3653 (+26.3%)	0.3582 (+23.8)

18.3 이론적 근거

Belkin, et al.(1993)은 동일한 정보 요구에 대한 서로 다른 부울 질의 표현들의 결합이 검색 효과의 향상으로 나타남을 보였으며, 또한 이러한 검색 효과 향상에 대한 이유를 다음과 같이 설명하였다.

> *질의에 대한 서로 다른 표현, 데이터베이스의 문서들에 대한 서로 다른 표현 또는 서로 다른 검색 기법은 서로 다른 집합의 적합 문서들과 서로 다른 집합의 부적합 문서들을 검색한다.*

위에서 제시된 다중 증거 결합에 대한 이론적 근거는 다음과 같은 데이터 퓨젼에 대한 초기 연구 결과로부터 도출되었다. 첫째, McGill, Koll & Norreault(1979)는 동일한 정보 요구에 대한 표현 방법에 따라, 즉 서로 다른 사용자가 문서를 검색하거나 또는 통제어 사용 여부에 따라 상이한 문서들이 검색됨을 발견하였다. 둘째, Katzer, et al.(1982)은 질의 표현 방

법 대신에 문서 표현 방법이 검색 효과에 미치는 영향을 조사하였으며, 서로 다른 문서 표현을 사용한 검색들이 매우 상이한 문서들을 검색함을 발견하였다.

한편, 데이터 퓨젼에 대한 다른 연구 결과를 살펴보면, 위에서 언급된 다중 증거 결합에 대한 이론적 근거가 수정되어야 함을 알 수 있다. Turtle & Croft(1991)는 동일한 정보 요구를 확률과 부울 질의로 표현하고, 그 검색 결과들을 결합하여 검색 효과의 향상을 얻었다. 또한 이들은 다음과 같은 매우 흥미있는 분석 결과를 제시하였다.

> *우리는 연구 초기에 확률 질의와 부울 질의의 검색 결과 결합이 검색 효과를 개선시키는 이유들 중의 하나로서 다음과 같은 생각을 지니고 있었다: 서로 다른 형태의 2 가지 질의는 서로 다른 적합 문서들을 검색하기 때문에, 결합된 검색 결과는 각각의 검색 결과에 포함된 적합 문서들의 수에 비하여 보다 많은 적합 문서들을 포함한다. 그러나 실험을 수행한 결과, 부울 질의에 의해 검색된 문서들은 확률 질의에 의해 검색된 문서들의 부분 집합 이었다.*

상기의 분석 결과는 유사한 문서들을 검색하는 두 개의 서로 다른 검색 실행을 결합할 지라도 검색 효과의 향상을 얻을 수 있음을 제시하고 있으며, 따라서 Belkin et al.이 제시한 다중 증거 결합의 이론적 근거에 대한 수정이 요구된다.

Saracevic & Kantor(1988)는 동일한 정보 요구에 대하여 여러 사용자에게 부울 질의를 생성하도록 요청하고, 생성된 다양한 질의를 검색에 이용하였다. 그 결과 서로 다른 질의 표현이 서로 다른 문서들을 검색함을 재확인하였다. 그러나 또한 어떠한 문서가 보다 많은 질의에 의해 검색될수록 적합 문서로 판단될 가능성이 증가함을 추가로 발견하였다. 이러한 연구 결과로부터 다음과 같은 다중 증거 결합에 대한 새로운 이론적 근거를 유도할 수 있다: *서로 다른 검색 실행은 유사한 집합의 적합 문서들을 검색하나, 서로 다른 집합의 부적합 문서들을 검색한다.*

다중 증거 결합에 대한 새로운 이론적 근거를 입증하기 위하여 두 개의 검색 결과에 있어서 적합 문서들과 비적합 문서들 사이의 중복 정도를 나타내는 중복 계수 $R_{overlap}$ 과 $N_{overlap}$ 을 계산하였다. 두 개의 검색 결과 $run1$ 과 $run2$ 에 대한 중복 계수 $R_{overlap}$ 과 $N_{overlap}$은 다음과 같이 정의된다.

$$R_{overlap} = \frac{common_{rel} \times 2}{run1_{rel} + run2_{rel}}$$

$$N_{overlap} = \frac{common_{nonrel} \times 2}{run1_{nonrel} + run2_{nonrel}}$$

$common_{rel}$ $run1$, $run2$ 가 공통적으로 검색한 적합 문서 수

$common_{nonrel}$ $run1$, $run2$ 가 공통적으로 검색한 부적합 문서 수

$run1_{rel}$	*run*1 이 검색한 적합 문서 수
$run1_{nonrel}$	*run*1 이 검색한 부적합 문서 수
$run2_{rel}$	*run*2 가 검색한 적합 문서 수
$run2_{nonrel}$	*run*2 가 검색한 부적합 문서 수

위의 식에서 $R_{overlap}$ 은 *run1* 과 *run*2 가 동일한 집합의 적합 문서들을 검색하면 1 의 값을 가지며, 어떠한 적합 문서도 공통으로 검색하지 않으면 0 의 값을 갖는다. 또한, $N_{overlap}$ 은 *run*1 과 *run*2 가 동일한 집합의 부적합 문서들을 검색하면 1 의 값을 가지며, 어떠한 부적합 문서도 공통으로 검색하지 않으면 0 의 값을 갖는다.

일반적으로 데이터 퓨젼은 하나의 검색 시스템 내에서 수행된다. 즉, 하나의 검색 시스템이 검색 효과의 향상을 위하여 다양한 검색 결과를 생성한 후, 이들을 결합하여 하나의 검색 결과를 생성한다. 그러나, 본 절에서는 다중 증거 결합에 대한 이론적 근거를 분석하기 위하여 전혀 다른 검색 시스템들, 즉 학술 대회 TREC 에 참가한 다양한 검색 시스템들이 생성한 검색 결과들을 이용하였다. TREC 에 참가한 시스템들은 동일한 문서와 질의 집합에 대한 검색을 수행하며, 따라서 이러한 시스템들이 생성한 검색 결과들의 결합은 일종의 데이터 퓨젼으로 간주될 수 있다.

본 절에서는 TREC3 ad-hoc 트랙에 참가한 시스템들의 검색 결과들을 다중 증거 결합에 대한 이론적 근거의 분석을 위해 활용하였다. 즉, TREC3 ad-hoc 트랙에 제출된 검색 결과들 중에서 *westp*1, *pircs*1, *vtc*5*s*2, *brkly*6, *eth*001, *nyuir*1 이라고 불리는 6 개의 검색 결과를 선택한 후, 이들의 쌍들에 대한 중복 계수를 계산하였다. <표 18.11>은 각 쌍들에 대한 중복 계수 $R_{overlap}$ 과 $N_{overlap}$ 을 보여준다. 이 표로부터 적합 문서들 사이의 중복 계수 $R_{overlap}$ 이 부적합 문서들 사이의 중복 계수 $N_{overlap}$ 보다 훨씬 큼을 알 수 있다. 이러한 특성은 서로 다른 검색 실행들이 유사한 적합 문서들을 검색하나, 서로 다른 부적합 문서들을 검색함을 의미하며, 또한 이러한 특성은 어떠한 문서가 보다 많은 검색 실행에 의해 검색될수록 그 문서에 보다 높은 순위가 부여되어야 한다는 Saracevic & Kantor 의 연구 결과와 일치한다.

18.4 결합 방법

일반적으로 서로 다른 검색 실행들은 상이한 범위의 유사도 값들을 생성할 수 있다. 예를 들어, <표 18.12>는 TREC3 ad-hoc 트랙에 참가한 6 개 검색 시스템이 TREC 질의 151 에 대해 생성한 최소 및 최대 유사도를 보여준다. 이 표로부터 서로 다른 검색 시스템들이 매우 다른 범위의 유사도를 문서들에 할당함을 알 수 있다. 따라서 검색 실행들의 결

과를 결합하기에 앞서, 유사도의 최소 및 최대값을 일치시키기 위한 유사도 정규화가 수행되어야 한다. 본 연구에서는 다음과 같이 식을 사용하여 유사도에 대한 정규화를 수행하였다.

$$normalized\_similarity = \frac{unnormalized\_similarity - minimum\_similarity}{maximum\_similarity - minimum\_similarity}$$

<표 18.11> 적합 및 부적합 문서들 사이의 중복 계수

		*westp*1	*pircs*1	*vtc*5*s*2	*brkly*6	*eth*001
*pircs*1	$R_{overlap}$	0.7970				
	$N_{overlap}$	0.3620				
*vtc*5*s*2	$R_{overlap}$	0.7712	0.7562			
	$N_{overlap}$	0.3009	0.3035			
*brkly*6	$R_{overlap}$	0.7846	0.7813	0.7846		
	$N_{overlap}$	0.3522	0.3649	0.3272		
*eth*001	$R_{overlap}$	0.7706	0.7927	0.7686	0.8253	
	$N_{overlap}$	0.3260	0.3869	0.2936	0.4179	
*nyuir*1	$R_{overlap}$	0.7902	0.8210	0.7457	0.7562	0.7882
	$N_{overlap}$	0.3517	0.4360	0.3303	0.3238	0.4009

<표 18.12> TREC3 ad-hoc 트랙에 참가한 6개 검색 시스템이 TREC질의 151에 대해 생성한 최소 및 최대 유사도

	최대 유사도	최소 유사도
*westp*1	0.7533	0.6567
*pircs*1	6.1525	2.0258
*vtc*5*s*2	1.8289	0.6860
*Brkly*6	0.4682	0.1415
*eth*001	0.3780	0.0903
*nyuir*1	28643	6326

Fox & Shaw(1994)는 동일한 정보 요구에 대하여 3 개의 확장된 부울 질의 *Pn*1.0, *Pn*2.0, *Pn*3.0 과 2 개의 벡터 질의 *LV*, *SV*를 생성하고, *AP*-1, *AP*-2, *WSJ*-1, *WSJ*-2 라 불리는 다양한 문서 집합에 대해 검색을 수행하였다. 그리고, 그 검색 결과들을 <표 18.13>에서 기술된 5 개의 함수 *CombMIN*, *CombMAX*, *CombSUM*, *CombANZ*, *CombMNZ*를 이용하여 결합하였다. <표 18.14>는 Fox & Shaw 가 제시한 실험 결과를 인용한 것으로, *CombSUM* 이 다른 결합 함수들보다 높은 검색 효과를 제공하고 있다. 이러한 결과는 *CombSUM*이 CombMAX, *CombMIN*, *CombANZ*보다 높은 검색 효과를 제공한다는 점에서,

보다 많은 검색 실행에 의해 검색되는 문서에게 보다 높은 순위를 부여해야 한다는 본 연구의 분석 결과와 일치한다. 그러나, *CombMNZ* 이 *CombSUM* 보다 낮은 검색 효과를 제공한다는 점에서 본 연구의 분석 결과와 일치하지 않는다.

<표 18.13> Fox & Shaw 가 제안한 다중 증거 결합 함수

CombMIN	서로 다른 검색 실행들이 산출한 유사도들 중 최소값
CombMAX	서로 다른 검색 실행들이 산출한 유사도들 중 최대값
CombSUM	서로 다른 검색 실행들이 산출한 유사도들의 총합
CombANZ	*CombSUM* ÷ (0이 아닌 유사도의 수)
CombMNZ	*CombSUM* × (0이 아닌 유사도의 수)

<표 18.14> 다중 증거 결합 함수들의 성능(유사도 정규화를 수행하지 않는 경우)

검색 실행	*AP*-1	*WSJ*-1	AP-2	WSJ-2	평균
*Pn*1.0	0.2810	0.2491	0.3004	0.2206	0.2740
*Pn*1.5	0.3122	0.3199	0.3332	0.2327	0.2995
*Pn*2.0	0.3027	0.3217	0.3300	0.2325	0.2967
SV	0.2387	0.2203	0.2543	0.1503	0.2159
LV	0.2435	0.2414	0.2664	0.1633	0.2287
CombMIN	0.2863	0.1924	0.3047	0.1308	0.2286
CombMAX	0.2856	0.3205	0.3337	0.2343	0.2935
CombSUM	0.3493	0.3605	0.3748	0.2752	0.3340
CombANZ	0.3493	0.3367	0.3748	0.2465	0.3268
CombMNZ	0.3059	0.3368	0.3516	0.2467	0.3103

Fox & Shaw 가 수행한 실험에 있어서 유의할 사항은 검색 결과에 대한 정규화를 수행하지 않았다는 것이다. 앞에서 언급된 것처럼 서로 다른 검색 실행은 매우 상이한 범위의 유사도 값들을 생성할 수 있으며, 유사도에 대한 정규화를 수행하지 않았을 경우 다중 증거의 결합시 예상치 못한 결과를 생성할 수 있다. 따라서, 본 연구에서는 TREC3 ad-hoc 트랙에 참가한 6 개 검색 시스템의 결과들을 정규화한 후, Fox & Shaw 가 제안한 5 개의 결합 함수들을 사용하여 결합함으로써, Fox & Shaw 가 제안한 5 개의 결합 함수들을 재평가하였다. <표 18.15>는 그 결과를 보여주며, *CombMNZ* 이 *CombSUM* 보다 다소 높은 검색 효과를 제공함을 알 수 있다.

또한, 본 연구에서는 3 개 이상의 검색 결과들을 결합했을 때, <표 18.13>의 결합 함수들이 검색 효과에 미치는 영향을 살펴보았다. 즉, TREC3 ad-hoc 트랙에 참가한 6 개 검색 시스템의 결과들을 정규화한 후, 2 단계에서는 2 개씩 결합하여 15 개, 3 단계에서는 3 개씩

결합하여 20 개, 4 단계에서는 4 개씩 결합하여 15 개, 5 단계에서는 5 개씩 결합하여 6 개, 6 단계에서는 6 개 모두를 결합하여 1 개의 결합된 검색 결과를 생성하였다. <표 18.16>은 각 단계별로 결합 결과의 검색 효과 평균을 보여주며, 이로부터 *CombMNZ* 이 *CombSUM* 보다 여전히 다소 높은 검색 효과를 제공함을 알 수 있다.

결합 함수 *CombSUM*과 *CombMNZ*는 다음에서 정의되는 함수 *CombGMNZ*로 일반화 될 수 있다.

$$CombGMNZ = CombSUM \times number\_of\_nonzero\_similarities^{\gamma}\ ,\ \gamma \geq 0$$

CombGMNZ 은 γ =0 일 때 *CombSUM* 과 동등한 식이 되며, γ =1 일 때 *CombMNZ* 과 동등한 식이 된다. 파라미터 γ 는 보다 많은 검색 실행에 의해 검색된 문서에 얼마나 높은 가중치를 부여하는가와 관계가 있다. 본 연구에서는 파라미터 γ 의 값을 변화시키면서, TREC3 ad-hoc 트랙에 참가한 6 개 검색 시스템의 결과들의 쌍을 결합하였다. <표 18.17>은 파라미터 γ 의 값이 1 일 때, 가장 높은 검색 효과를 제공하는 결합 결과를 얻을 수 있음을 보여준다.

<표 18.15> 다중 증거 결합 함수들의 성능(유사도 정규화를 수행한 경우)

	CombMIN	*CombMAX*	*CombSUM*	*CombANZ*	*CombMNZ*
*westp*1 & *pircs*1	0.2780	0.3377	0.3586	0.3280	0.3604
*westp*1 & *vtc5s*2	0.2524	0.3355	0.3690	0.3100	0.3737
*westp*1 & *brkly*6	0.2641	0.3238	0.3490	0.3111	0.3525
*westp*1 & *eth*001	0.2560	0.3258	0.3502	0.3112	0.3522
*westp*1 & *nyuir*1	0.2566	0.3205	0.3505	0.3054	0.3559
*pircs*1 & *vtc*5*s*2	0.2469	0.3221	0.3463	0.2994	0.3520
*pircs*1 & *brkly*6	0.2595	0.3083	0.3275	0.2979	0.3305
*pircs*1 & *eth*001	0.2637	0.3115	0.3287	0.3045	0.3293
*pircs*1 & *nyuir*1	0.2663	0.3062	0.3259	0.3014	0.3273
*vtc*5*s*2 & *brkly*6	0.2456	0.3106	0.3390	0.2911	0.3431
*vtc*5*s*2 & *eth*001	0.2295	0.3108	0.3468	0.2876	0.3543
*vtc*5*s*2 & *nyuir*1	0.2532	0.3150	0.3341	0.3027	0.3334
*brkly*6 & *eth*001	0.2554	0.2900	0.3142	0.2876	0.3170
*brkly*6 & *nyuir*1	0.2378	0.3011	0.3354	0.2834	0.3402
*eth*001 & *nyuir*1	0.2564	0.2877	0.3214	0.2963	0.3230
평 균	0.2548	0.3144	0.3398	0.3012	0.3430

<표 18.16> 3 개 이상의 검색 결과들을 결합할 경우 다중 증거 결합 함수들의 성능

	1단계	2단계	3단계	4단계	5단계	6단계
CombMIN	0.2884	0.2548	0.2237	0.2012	0.1849	0.1720
CombMAX	0.2884	0.3144	0.3273	0.3350	0.3404	0.3460
CombSUM	0.2884	0.3398	0.3646	0.3797	0.3899	0.3972
CombANZ	0.2884	0.3012	0.3058	0.3088	0.3115	0.3134
CombMNZ	0.2884	0.3430	0.3685	0.3835	0.3927	0.3991

<표 18.17> 다중 증거 결합 함수 *CombGMNZ*의 성능

	γ = 0 *CombSUM*	γ = 0.5	γ = 1 *CombMNZ*	γ = 2	γ = 5	γ = 10
*westp*1 & *pircs*1	0.3585	0.3603	0.3604	0.3599	0.3596	0.3596
*westp*1 & *vtc5s2*	0.3690	0.3731	0.3737	0.3728	0.3718	0.3718
*westp*1 & *brkly*6	0.3490	0.3519	0.3525	0.3520	0.3512	0.3511
*westp*1 & *eth*001	0.3502	0.3523	0.3522	0.3510	0.3501	0.3501
*westp*1 & *nyuir*1	0.3505	0.3548	0.3559	0.3557	0.3551	0.3550
*pircs*1 & *vtc5s2*	0.3463	0.3509	0.3520	0.3520	0.3515	0.3515
*pircs*1 & *brkly*6	0.3275	0.3302	0.3305	0.3300	0.3297	0.3297
*pircs*1 & *eth*001	0.3287	0.3295	0.3293	0.3285	0.3278	0.3278
*pircs*1 & *nyuir*1	0.3259	0.3272	0.3273	0.3264	0.3257	0.3257
vtc5s2 & *brkly*6	0.3390	0.3427	0.3431	0.3423	0.3418	0.3418
vtc5s2 & *eth*001	0.3468	0.3526	0.3543	0.3543	0.3537	0.3537
vtc5s2 & *nyuir*1	0.3341	0.3345	0.3334	0.3318	0.3308	0.3307
*brkly*6 & *eth*001	0.3142	0.3164	0.3170	0.3172	0.3173	0.3173
*brkly*6 & *nyuir*1	0.3354	0.3393	0.3402	0.3401	0.3395	0.3395
*eth*001 & *nyuir*1	0.3214	0.3230	0.3230	0.3224	0.3217	0.3217
평 균	0.3398	0.3426	0.3430	0.3424	0.3418	0.3418

18.5 순위 결합

일반적으로 데이터 퓨젼 분야에서 순위(rank)보다 유사도가 다중 증거를 결합하는데 많이 이용된다. 이는 연구자들이 순위보다 유사도를 사용한 결합이 보다 높은 검색 효과를 제공한다고 믿고 있기 때문이나, 유사도를 사용한 다중 증거 결합이 순위를 사용한 다중 증거 결합보다 높은 검색 효과를 제공한다는 사실은 입증되지 않은 상태이다. 따라서, 본 절에서는 다중 증거의 결합에 순위의 사용이 검색 효과에 미치는 영향을 분석하고자 한다.

본 연구에서 다중 증거 결합에 순위의 사용을 조사한 이유는 순위를 사용한 다중 증거 결합이 유사도를 사용한 다중 증거 결합보다 높은 검색 효과를 제공하는 경우가 있다고 믿기 때문이다. 즉, 다중 증거 결합에 유사도를 사용할 경우, 검색 실행이 제공하는 검색 효과

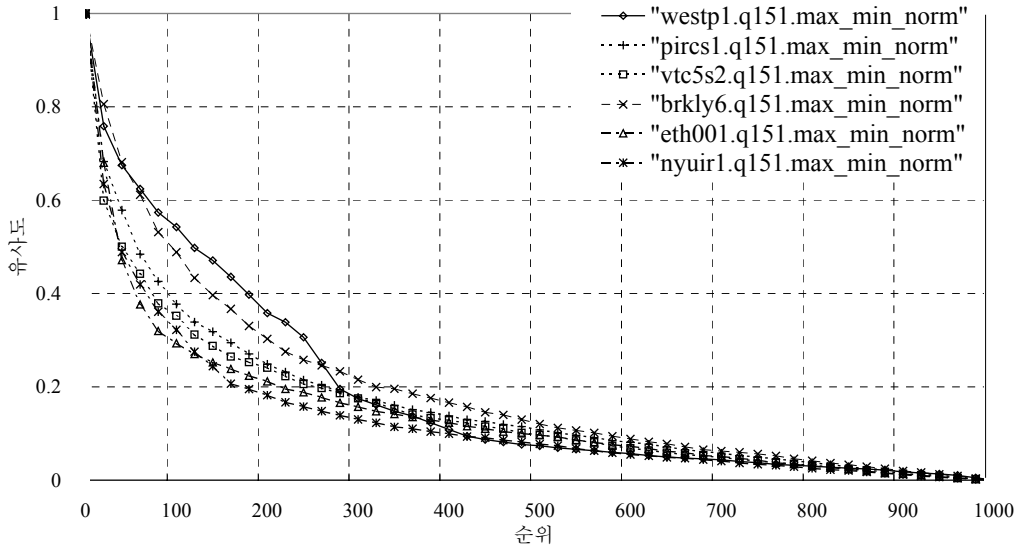
와 관계없이 특정 검색 실행의 결과를 우선적으로 결합에 반영하는 경우가 발생하며, 이로 인하여 검색 효과의 저하가 초래된다. 예를 들면, <그림 18.1>은 TREC3 ad-hoc 트랙에 참가한 6 개 시스템의 TREC 질의 151 번에 대한 검색 결과를 보여준다. 각각의 시스템들이 검색한 상위 1000 개의 문서들의 유사도를 0 부터 1 사이의 값으로 정규화한 후, 그 결과를 도식화하였다. <그림 18.1>은 *pircs*1 의 정규화된 유사도 값이 각 순위에 대해 *brkly*6 의 유사도 값보다 낮다는 것을 보여준다. 따라서, *pircs*1 과 *brkly*6 를 결합할 경우, 결합 결과에 *brkly*6 의 결과가 *pircs*1 의 결과보다 많이 반영된다. 이러한 특성은 *pircs*1 이 *brkly*6 보다 높은 검색 효과를 제공한다면 결합된 결과의 검색 효과를 감소시킬 수도 있다.

다음에서는 실험을 통하여 순위를 사용한 다중 증거 결합의 검색 효과를 살펴본다. 이를 위하여 다음에서 정의되는 함수 *Rank_Sim* 을 각 문서들의 순위에 적용하고, 그 결과 값을 그 문서에 대한 유사도로 활용하였다.

$$Rank\_Sim(rank) = 1 - \frac{rank - 1}{number\_of\_retrieved\_documents}$$

예를 들어, 임의의 검색 실행이 유사도가 높은 상위 1000 개의 문서들을 검색한다고 가정하자. 이때 어떠한 문서가 10 위의 순위를 갖는다면, 그 문서의 유사도는 함수 *Rank_Sim* 에 의해 0.991 이 된다.

다음의 실험에서는 TREC3 ad-hoc 트랙에 참가한 6 개 시스템의 검색 결과들에 함수 *Rank_Sim* 을 적용하여 문서들의 순위를 유사도로 변환하였다. 그리고, 검색 결과들의 쌍들을 *CombMNZ* 함수를 사용하여 결합한 후, 그 결과의 검색 효과를 평가하였다. <표 18.18>은 미세하지만 유사도의 사용이 순위의 사용보다 높은 검색 효과를 제공함을 보여준다. <그림 18.1>로부터 실험에 사용된 6 개 검색 결과들은 유사한 순위-유사도 곡선을 제공함을 알 수 있다. 따라서, <그림 18.1>과 <표 18.1>로부터 유사한 순위-유사도 곡선을 제공하는 검색 결과들의 결합에는 유사도의 사용이 순위의 사용보다 높은 검색 효과를 제공함을 알 수 있다.



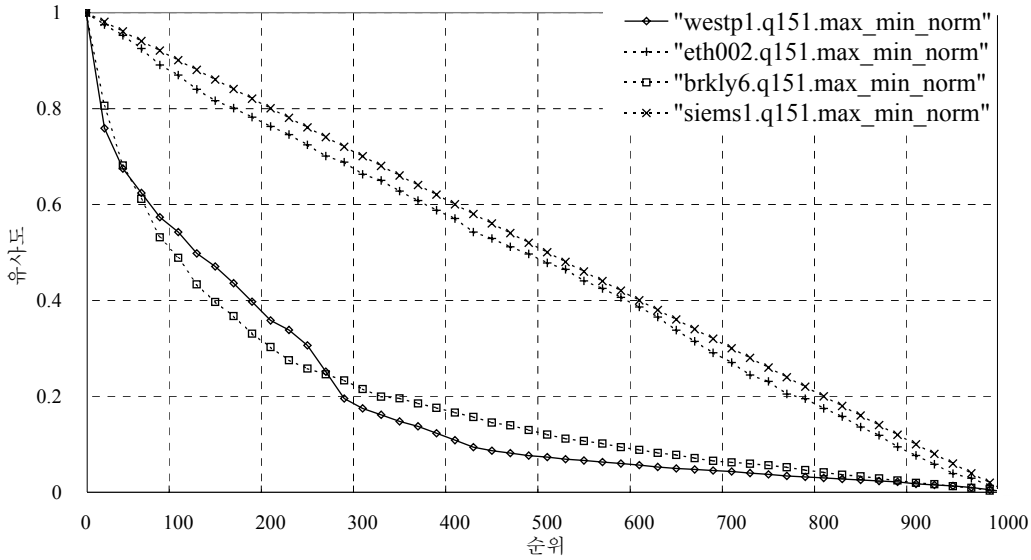
<그림 18.1> 순위-유사도 곡선

<표 18.18> 유사도 결합과 순위 결합의 검색 효과 (유사한 순위 - 유사도 곡선을 갖는 검색 결과들을 결합할 경우)

		*westp*1	*pircs*1	*vtc*5*s*2	*brkly*6	*eth*001
*pircs*1	rank similarity	0.3525 0.3604				
*vtc*5*s*2	rank similarity	0.3654 0.3737	0.3476 0.3520			
*brkly*6	rank similarity	0.3460 0.3525	0.3276 0.3305	0.3381 0.3431		
*eth*001	rank similarity	0.3442 0.3522	0.3260 0.3293	0.3519 0.3543	0.3173 0.3170	
*nyuir*1	rank similarity	0.3504 0.3559	0.3250 0.3273	0.3282 0.3334	0.3373 0.3402	0.3205 0.3230

위의 결과는 앞에서 언급된 유사도의 사용이 순위의 사용보다 낮은 검색 효과를 제공할 것이라는 추측에 위배된다. 그러나, TREC3 ad-hoc 트랙에 참가한 일부 시스템들은 매우 상이한 순위-유사도 곡선을 제공한다. 예를 들어, <그림 18.2>는 *westp*1, *brkly*6 의 순위-유사도 곡선이 *eth*002, *siems*1 의 것과는 매우 상이함을 보여준다. 다음의 실험에서는 4 개의 검색 결과들 *westp*1, *eth*002, *brkly*6, *siems*1 의 쌍을 결합하였으며, <표 18.19>는 그 결과를 보여준다. 이 표로부터 유사한 순위-유사도 곡선을 제공하는 검색 결과들 *westp*1 & *brkly*6, *eth*002 & *siems*1 을 결합할 경우에는 유사도의 사용이 보다 높은 검색 효과를 제

공하나, 매우 상이한 순위-유사도 곡선을 제공하는 검색 결과들 *westp*1 & *eth*002, *westp*1 & *siems*1, *eth*002 & *brkly*6, *brkly*6 & *siems*1 을 결합할 경우에는 순위의 사용이 보다 높은 검색 효과를 제공함을 알 수 있다.



<그림 18.2> 순위-유사도 곡선

<표 18.19> 유사도 결합과 순위 결합의 검색 효과 (매우 상이한 순위 - 유사도 곡선을 갖는 검색 결과들을 결합할 경우)

	*Westp*1 *eth002*	*westp*1 *brkly6*	*westp*1 *siems1*	*eth*002 *brkly6*	*eth*002 *siems1*	*brkly*6 *siems1*
rank similarity	0.3495 0.3460	0.3460 0.3525	0.3255 0.3166	0.3210 0.3164	0.2847 0.2859	0.2929 0.2842