Journal of Korea Multimedia Society Vol. 20, No. 9, September 2017(pp. 1541-1550)
https://doi.org/10.9717/kmms.2017.20.9.1541

# NoSQL 기반 연관 콘텐츠 추천 시스템의 설계 및 구현

고은정[†], 김호준[††], 박효주[†††], 전영호[††††], 이기훈[†††††], 신사임[††††††]

# Design and Implementation of a System for Recommending Related Content Using NoSQL

Eun-Jeong Ko[†], Ho-Jun Kim[††], Hyo-Ju Park[†††], Young-Ho Jeon[††††],
Ki-Hoon Lee[†††††], Saim Shin[††††††]

## ABSTRACT

The increasing number of multimedia content offered to the user demands content recommendation. In this paper, we propose a system for recommending content related to the content that user is watching. In the proposed system, relationship information between content is generated using relationship information between representative keywords of content. Relationship information between keywords is generated by analyzing keyword collocation frequencies in Internet news corpus. In order to handle big corpus data, we design an architecture that consists of a distributed search engine and a distributed data processing engine. Furthermore, we store relationship information between keywords and relationship information between keywords and content in NoSQL to handle big relationship data. Because the query optimizer of NoSQL is not as well developed as RDBMS, we propose query optimization techniques to efficiently process complex queries for recommendation. Experimental results show that the performance is improved by up to 69 times by using the proposed techniques, especially when the number of requested related keywords is small.

**Key words:** Related Content, Related Keywords, Recommendation System, NoSQL, Query Optimization

## 1. 서 론

빅데이터 시대가 도래하면서 추천 시스템에 대한 연구가 더욱 활발해졌을 뿐만 아니라, 추천 시스템의 종류도 다양해지고 있다[1]. 추천 시스템은 소셜 네트워크 서비스(SNS)의 친구 추천, 온라인 쇼핑몰의 상품 추천, 음원 사이트의 음악 추천 등 다양한 분야에서 사용되고 있으며 점점 그 중요성이 부각되고

※ Corresponding Author : Ki-Hoon Lee, Address: (01897) Kwangwoon-ro 20, Nowon-gu, Seoul, Korea, TEL : +82-2-940-8674, E-mail : kihoonlee@kw.ac.kr
Receipt date : Jun. 26, 2017, Revision date : Jun. 29, 2017
Approval date : Jul. 19, 2017
[†] School of Computer and Information Engineering, Kwangwoon University
(E-mail : kej3535@kw.ac.kr)
[††] School of Computer and Information Engineering, Kwangwoon University
(E-mail : timetopray@naver.com)
[†††] School of Computer and Information Engineering, Kwangwoon University
(E-mail : gkwk011@naver.com)
[††††] School of Computer and Information Engineering, Kwangwoon University
(E-mail : pointnb@naver.com)
[†††††] School of Computer and Information Engineering, Kwangwoon University
(E-mail : kihoonlee@kw.ac.kr)
[††††††] Korea Electronics Technology Institute
(E-mail : sishin@keti.re.kr)
※ This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2015R 1C 1A 1A02036517).

있다. 특히 세계 최대 온라인 쇼핑몰인 아마존닷컴의 경우 매출의 35%가 추천 시스템에 의해 발생하고 있으며, 미국의 온라인 동영상 스트리밍 서비스인 넷플릭스의 경우 추천 시스템이 추천한 영화가 대여되는 영화의 65% 이상을 차지한다[2].

한편, 방송 콘텐츠 산업은 사용자가 단순히 시청만 하던 단방향 서비스 형태로 시작하여, 시청자 참여 TV 프로그램(시청자 의견, 퀴즈 등)이나 T-커머스(T-commerce) 등의 소통이 가능한 양방향 서비스 형태로 발전하였다. 최근에는 인터넷, 모바일 및 소셜 서비스 등과 결합하여 사용자가 언제 어디서나 원하는 콘텐츠를 자유롭게 선택하여 시청할 수 있게 되었으며, 방송 콘텐츠의 개인화를 위한 연구도 활발하게 진행되고 있다[3]. 하지만 수많은 콘텐츠들이 생겨나면서 사용자가 자신에게 알맞은 콘텐츠를 쉽고 빠르게 찾는 것이 어려워졌다. 사용자에게 알맞은 콘텐츠를 추천하는 연구는 주로 사용자의 프로파일과 콘텐츠의 메타데이터를 서로 매칭하는 형태로 진행되어 왔다. 하지만 이 방법은 사용자의 프로파일 정보가 없는 경우에는 추천이 불가능하다는 문제가 있으며, 콘텐츠의 메타데이터만으로 콘텐츠를 추천해주는 연구는 잘 이루어지지 않고 있는 실정이다.

본 논문에서는 사용자의 프로파일 정보 없이, 콘텐츠 간의 연관 정보를 이용하여 콘텐츠 시청 시에 사용자에게 해당 콘텐츠와 연관된 콘텐츠들을 추천해주는 시스템을 제안한다. 제안하는 시스템에서 콘텐츠 간 연관 정보는 키워드 간의 연관 정보를 바탕으로 생성한다. 연관 키워드를 이용한 콘텐츠 추천 시스템은 기존에 연구된 바가 없는 것으로, 키워드 간 연관 정보는 인터넷 뉴스 문서에 함께 등장하는 빈도가 높은 키워드들을 분석하여 생성하며, 콘텐츠 간 연관 정보는 키워드와 콘텐츠들을 매핑하여 생성한다. 예를 들어, 콘텐츠 $A$, $B$를 대표하는 키워드를 각각 $Key_A$, $Key_B$라고 할 때, $Key_A$와 $Key_B$가 함께 등장하는 빈도가 높으면 콘텐츠 $A$, $B$가 연관되어 있다고 판단한다. 제안하는 시스템의 성능을 높이기 위하여 인터넷 뉴스 문서의 전처리 과정에서 분산 및 병렬 처리를 적용하며, 전처리를 마친 데이터를 분석하는 과정에서도 분산 검색 엔진과 분산 데이터 처리 엔진을 사용해 키워드 간의 연관 정보를 생성하는 전 과정을 분산화한다. 또한 생성된 키워드 간의 연관 정보 및 콘텐츠 정보는 NoSQL에 저장하여 분산 처리한다. NoSQL은 대용량 데이터일 경우, 관계형 데이터베이스 관리 시스템(Relational Database Management System; RDBMS)에 비하여 데이터 처리 성능이 더 좋을 수 있다[4]. 하지만 NoSQL의 질의 최적화기는 RDBMS처럼 발달이 되어 있지 않아 복잡한 질의에 대해서는 최적의 실행 계획을 찾지 못하는 경우가 많다. 따라서 본 논문에서는 콘텐츠 메타데이터만을 이용한 연관 콘텐츠 추천 시스템을 제안할 뿐만 아니라, 콘텐츠 간의 연관 정보를 생성하는 복잡한 질의를 NoSQL에 맞게 최적화하는 방법을 제안한다. 실험을 통해 질의 최적화 유무에 따른 실행 시간을 비교한 결과, 질의를 최적화한 경우, 최적화하지 않았을 때와 비교하여 성능이 최대 69배 향상된 것을 확인할 수 있었으며, 하나의 주요 키워드 당 추출할 연관 키워드의 개수가 적을수록 성능이 좋았다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 소개하며, 3장에서는 제안하는 시스템에 대하여 자세히 설명한다. RDBMS에 기반한 구현을 먼저 설명한 뒤, NoSQL에 맞게 최적화한 구현을 비교하여 설명한다. 4장에서는 제안한 시스템의 성능 평가를 다룬다. 마지막으로 5장에서 본 논문의 결론을 맺는다.

## 2. 관련 연구

[5]는 사용자 로그를 분석하여 사용자 별로 선호할 것으로 예상되는 콘텐츠들을 추천해 주는 시스템을 제안하였다. [6]은 국가표준 디지털 식별자인 UCI의 메타데이터를 확장하여, 대규모 콘텐츠 추천에 효과적으로 활용할 수 있도록 하였다. [7]은 하둡 클러스터를 이용한 트렌드 분석 서비스를 제안하였다. 트렌드 분석을 위하여 인터넷 뉴스 사이트에서 뉴스 문서를 수집한다. 전처리 과정을 거쳐, 수집한 문서에서 단어만을 추출한 후에 하둡 클러스터를 이용해 단어의 빈도를 계산한다. 계산한 단어 빈도를 데이터베이스에 저장하여 사용자가 트렌드를 검색하고 확인할 수 있도록 하였다. [8]은 대량의 데이터를 사용하는 추천 시스템의 성능을 효과적으로 높이기 위하여 분산 그래프 데이터 모델을 제안하였다. 대량의 사용자 데이터를 처리하는 경우에도 추천 정확도를 유지하고 검색 시간을 단축하기 위하여, 새로운 형태의 유사 사용자 인덱스가 적용된 그래프 데이터 저장 시스

템을 구현하였다. 성능 평가를 위하여 RDBMS 기반의 추천 시스템과 [8]에서 제안한 시스템의 추천 항목 구성 시간을 각각 측정하였고, 대규모 데이터를 분석하여 추천하는 시스템인 경우에 [8]에서 제안한 시스템이 RDBMS를 사용한 시스템보다 효과적으로 성능을 높일 수 있음을 보였다. 본 논문에서 제안하는 연관 콘텐츠 추천 시스템은 뉴스 문서에 함께 등장하는 키워드들의 빈도를 분석하여 키워드들 간의 연관 정보를 생성하고, 이를 이용하여 연관 콘텐츠를 추천한다. 따라서 [5] 및 [6]과 달리, 사용자 로그가 없어도 추천이 가능하다. 본 논문에서는 대량의 데이터를 효율적으로 처리하기 위하여 분산 처리가 가능한 NoSQL을 사용하며, 콘텐츠 추천을 위하여 NoSQL에 적합하게 질의를 최적화하는 방법을 제안한다.

## 3. 연관 콘텐츠 추천 시스템

본 논문에서는 빅데이터 처리가 가능한 연관 콘텐츠 추천 시스템과 그 시스템에서 콘텐츠 간의 연관 정보를 생성하는 복잡한 질의를 NoSQL에 맞게 최적화하는 방법을 제안한다. 전체적인 시스템의 아키텍처는 Fig. 1과 같다. 제안하는 시스템은 크게 연관 키워드 추출 시스템과 연관 콘텐츠 검색 시스템으로 나뉜다.

### 3.1 연관 키워드 추출 시스템

연관 키워드 추출 시스템은 콘텐츠 간 연관 정보의 바탕이 되는 키워드 간 연관 정보를 생성하는 시스템으로, 특정 기간 동안 많이 언급된 주요 키워드 및 특정 키워드의 연관 키워드를 제공한다. 인터넷 뉴스 문서에는 시대의 흐름이 반영되며, 분야별로 분류가 잘 되어있기 때문에 인터넷 뉴스 문서의 분석을 통하여 특정 분야의 주요 키워드나 연관 키워드를 얻을 수 있다. 따라서 연관 키워드 추출 시스템에서는 대량의 인터넷 뉴스 문서를 수집하고, 형태소 분석기와 개체명 인식기 등을 이용해 수집한 문서에서 개체명이나 명사 등의 의미 있는 키워드를 추출하는 전처리를 진행한다. 전처리 과정에서 대용량 데이터의 처리 성능 및 확장성을 높이기 위하여 하나의 노드에서 멀티 코어를 모두 활용하는 병렬 처리와 이를 여러 대의 노드로 확장하는 분산 처리를 적용한다.

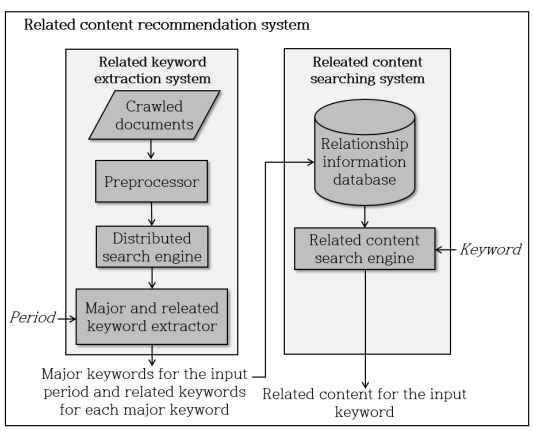



Fig. 1. The architecture of the system for recommending related content.

전처리를 마친 데이터는 분산 검색 엔진인 Elasticsearch[9]에 저장한다. Elasticsearch는 Apache Lucene[10]을 기반으로 개발된 오픈소스 실시간 분산 검색 엔진으로, 뛰어난 전문검색을 지원하며 확장성도 좋다. 본 시스템에서는 Elasticsearch에 전처리가 완료된 인터넷 뉴스 문서를 월 단위로 색인하여 저장한다. 다음으로 분산 데이터 처리 엔진인 Spark[11]를 이용하여 키워드의 빈도수를 구한다. Spark는 대용량 데이터를 처리하고 분석할 수 있는 인메모리 클러스터 컴퓨팅 프레임워크로서, 분산 및 병렬 처리가 가능하며 메모리를 이용해 데이터를 빠른 속도로 처리할 수 있다. 본 시스템은 Spark와 Elasticsearch를 연동하여 특정 기간의 문서들을 대상으로 키워드의 빈도수를 구한다. 이때 출현 빈도가 높은 키워드들을 해당 기간의 주요 키워드로 본다. 또한 특정 키워드가 포함된 문서 집합을 대상으로 해당 문서 집합에 나타난 모든 키워드들의 빈도수를 구하여, 빈도가 높은 키워드들을 특정 키워드와 관련이 높은 연관 키워드로 본다. 사용자는 특정 기간의 주요 키워드나 특정 키워드와 관련된 연관 키워드들을 추출할 수 있다.

연관 키워드 추출 시스템의 인터페이스는 Fig. 2와 같다. 화면의 왼쪽에서는 주요 키워드들을 추출할 수 있으며, 오른쪽에서는 사용자가 입력한 키워드의 연관 키워드들을 추출할 수 있다. 주요 키워드 및 연관 키워드를 추출할 기간, 추출할 키워드 개수(all, 10, 20, 50, 100, 1000, 직접입력), 추출 대상 문서의 출처(네이버뉴스, 일간스포츠), 추출 키워드의 카테고리 종류를 지정하여 추출할 수 있다. 연관 키워드


Fig. 2. The web interface for extracting major and related keywords.

추출 시스템은 또한 주요 키워드 추출 결과를 바탕으로 각 주요 키워드의 연관 키워드들을 추출할 수 있는 기능을 제공한다.

### 3.2 연관 콘텐츠 검색 시스템

연관 콘텐츠 검색 시스템은 연관 키워드 추출 시스템에서 얻은 주요 키워드에 대한 연관 키워드 정보를 이용하여 현재 시청하고 있는 콘텐츠를 대표하는 키워드와 연관된 콘텐츠를 검색해준다. 콘텐츠를 대표하는 키워드는 콘텐츠의 제목, 태그, 설명 등을 바탕으로 구할 수 있다. 연관 콘텐츠 검색을 위해 연관 키워드 추출 시스템에서 얻은 키워드 간 연관 정보를 데이터베이스에 저장한다. 키워드 간 연관 정보는 대량의 데이터이므로 RDBMS보다는 NoSQL에 저장하여 분산 처리하는 것이 더 빠를 수 있다. 하지만 NoSQL은 질의 최적화기가 RDBMS처럼 발달이 되어 있지 않기 때문에 복잡한 질의를 처리하는 경우, NoSQL에 맞게 질의를 최적화해 주어야한다. 먼저 RDBMS 기반 연관 콘텐츠 검색 시스템의 구현을 다루고, NoSQL에 맞게 최적화한 연관 콘텐츠 검색 시스템의 구현을 제안한다.

#### 3.2.1 RDBMS 기반 연관 콘텐츠 검색 시스템

연관 키워드 추출 시스템에서 얻은 키워드 간 연관 정보를 연관 정보 데이터베이스에 저장한다. 연관 정보 데이터베이스에는 KEYWORD 테이블, COLLO_PAIR 테이블, YOUTUBE_URL 테이블이 있으며, 각 테이블의 스키마는 Fig. 3과 같다. KEYWORD 테이블에는 주요 키워드(KEYWORD) 및 키워드의 ID, 등장한 빈도수(FREQ), 개체명 인식 결과(TAG)를 저장한다. COLLO_PAIR 테이블에는 주요 키워드와 같은 문서에 등장한 빈도가 높은 연관 키워드들을 저장하며, 하나의 주요 키워드 당 N개의 연관 키워드를 추출하여 저장한다. COLLO_PAIR 테이블의 연관 키워드의 ID(COLLO_ID)는 KEYWORD 테이블의 ID(KEYWORD_ID)를 참조하며, 연관 키워드가 주요 키워드와 함께 나타나는 빈도수(COLLO_FREQ)와 연관 키워드의 개체명 인식 결과(COLLO_TAG)를 저장한다. COLLO_PAIR 테이블의 COLLO_RATIO는 주요 키워드와 연관 키워드 간의 연관도를 나타내며, 주요 키워드의 빈도수로 COLLO_FREQ를 나눈 값 이다. 주요 키워드는 아니지만, 연관 키워드에 등장하는 키워드도 KEYWORD 테이블에 저장해 COLLO_ID 정보를 KEYWORD 테이블에서 찾을 수 있도록 하며, 주요 키워드와 구분하기 위하여 FREQ를 0으로 설정한다. YOUTUBE_URL 테이블에는 각 키워드와 관련된 콘텐츠들의 정보를 저장한다. 콘텐츠는 전 세계 최대 무료 동영상 공유 사이트인 유튜브[12]에 업로드된 콘텐츠를 활용하며, 대표 키워드의 ID(KEYWORD_ID), 콘텐츠 주소(URL), 제목(TITLE), 업로드 날짜(TIME), 설명(DESCRIP-

**KEYWORD** (ID, KEYWORD, FREQ, TAG)
**COLLO_PAIR** (COLLO_ID, COLLO_FREQ, COLLO_TAG, COLLO_RATIO)
**YOUTUBE_URL** (KEYWORD_ID, URL, TITLE, TIME, DESCRIPTION)

Fig. 3. The schema for the relationship information database.

TION)을 저장한다.

RDBMS 기반 연관 콘텐츠 검색 시스템의 질의를 관계 대수로 나타내면 Fig. 4와 같다. 이 질의는 현재 시청 중인 콘텐츠를 대표하는 키워드인 입력 키워드(:INPUT)와의 연관도가 높은 연관 키워드들에 콘텐츠를 매핑하는 질의이다. 먼저 키워드 정보와 콘텐츠 정보를 결합한다($T_0$). 이를 위하여 KEYWORD 테이블과 YOUTUBE_URL 테이블을 KEYWORD_ID에 대하여 조인하는데 키워드에 해당하는 콘텐츠가 YOUTUBE_URL 테이블에 존재하지 않아도 키워드의 정보를 포함하도록 왼쪽 외부 조인(LEFT OUTER JOIN) 연산을 적용한다. 다음으로, 입력 키워드 및 입력 키워드와 연관된 키워드들의 정보를 얻는다($T_1$). 이를 위하여 KEYWORD 테이블에서 입력 키워드의 정보를 얻는 데 입력 키워드 정보가 결과에 가장 먼저 나오도록 입력 키워드의 COLLO_RATIO에 최댓값(MAX_COLLO_RATIO)을 부여한다. 또한, 입력 키워드의 ID를 이용하여 COLLO_PAIR 테이블에서 입력 키워드에 대한 연관 키워드들의 정보를 가져온다. 이후, 입력 키워드와 연관 키워드 간의 연관도를 나타내는 COLLO_RATIO에 대해 내림차순으로 정렬하여 입력 키워드와 관련이 높은 순으로 정렬한다. $T_1$이 입력 키워드 정보와 연관 키워드 정보를 합쳐 최대 M개의 키워드 정보를 가지도록, COLLO_RATIO에 대해 정렬한 연관 키워드들의 정보에 LIMIT 연산을 적용하여 상위 M-1개의 행을 얻는다. 마지막으로, M개의 입력 및 연관 키워드($T_1$)와 콘텐츠 정보($T_0$)를 합친다. 이를 위하여 KEYWORD_ID에 대하여 $T_0$와 $T_1$에 자연 조인(NATURAL JOIN) 연산을 적용한다. 최종적으로 COLLO_RATIO에 대하여 내림차순, TIME에 대해 내림차순으로 정렬한다.

### 3.2.2 NoSQL 기반 연관 콘텐츠 검색 시스템

본 절에서는 제 3.2-1)절의 구현에서 RDBMS를 NoSQL로 변경한 연관 콘텐츠 검색 시스템을 제안한다. 제안하는 시스템은 대량의 데이터를 분산하여 빠르게 처리할 수 있다. 본 시스템은 다양한 NoSQL 중에서 RDBMS와 유사한 데이터 모델을 가지는 HBase를 사용한다. HBase[13,14]는 컬럼 기반 시스템으로, 동적으로 컬럼을 추가할 수 있는 동적 스키마를 가진다. HBase는 초당 수십만 건의 데이터 입출력이 가능하고 최대 페타바이트(PB)급까지 저장 가능하다. HBase는 하둡 분산 파일 시스템(Hadoop Distributed File System; HDFS)[15] 위에서 동작한다. ZooKeeper[16]는 분산 시스템에 꼭 필요한 분산

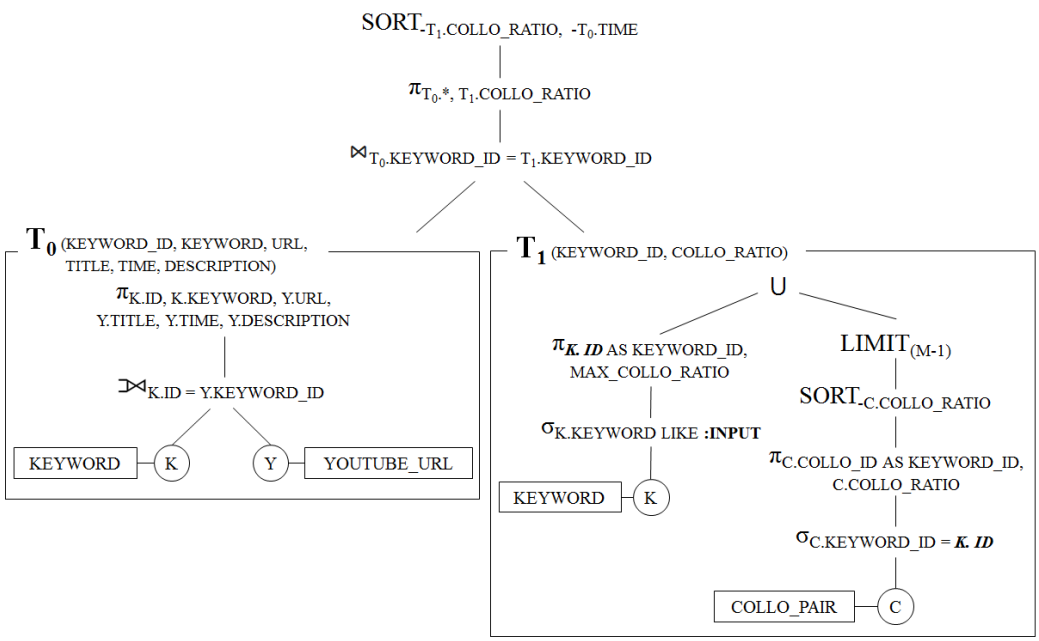



Fig. 4. The algebraic tree for searching related content stored in RDBMS.

**SELECT** …
**FROM**
( SUBQUERY$_0$ ) $T_0$,
( SUBQUERY$_1$ ) $T_1$
**WHERE** $T_0.col = T_1.col$

Fig. 5. The form of a complex query.

조정 시스템으로, 분산 시스템의 구성 정보를 유지 관리하며 처리 결과를 동기화하고 시스템 결함에 대한 대응을 제공해주는 중앙 집중식 서비스이다. 본 시스템은 하둡 분산 파일 시스템 위에서 동작하는 HBase를 ZooKeeper가 관리하는 형태이다. HBase는 SQL을 직접 지원하지 않으므로 SQL 질의를 HBase API로 변환하여 처리해주는 Phoenix[17]를 이용한다. Phoenix는 내부 조인(inner join)과 외부 조인(outer join) 연산을 모두 지원한다. 조인 알고리즘으로는 해시 조인(hash join)과 정렬-합병 조인(sort-merge join)을 지원하는데, 테이블 통계에 의해 제공되는 권고에 따라 자동으로 두 알고리즘 중 더 나은 알고리즘이 선택한다.

RDBMS에서는 질의 최적화기가 최적의 실행 계획을 찾기 때문에 Fig. 4의 질의를 빠르게 처리할 수 있지만, 질의 최적화기가 발달되어 있지 않은 NoSQL은 복잡한 질의를 효율적으로 처리하지 못하는 경우가 많다. 본 논문에서는 FROM절에 중첩된 질의에서 생성된 테이블들을 조인하는 질의를 복잡한 질의로 정의한다. RDBMS 기반 시스템에서 사용한 복잡한 질의를 제안하는 NoSQL 기반 시스템에 적합하게 최적화하기 위하여 분산 데이터베이스를 위한 대표적인 조인 전략 중 하나인 세미조인(semi-join)[18]을 이용한다. Fig. 5는 중첩 질의인 SUBQUERY$_0$와 SUBQUERY$_1$의 결과인 $T_0$ 테이블과 $T_1$ 테이블을 *col*에 대하여 조인하는 복잡한 질의이다. 이때 SUBQUERY$_1$의 결과가 SUBQUERY$_0$의 결과에 비해 상당히 적다면, SUBQUERY$_1$에서 SUBQUERY$_0$로 세미조인을 적용하여 질의를 최적화한다. 먼저 SUBQUERY$_1$를 계산해 $T_1$을 얻는다. 이후 $T_1$의 결과에서 조인되는 컬럼만 프로젝션(projection)하고, 이를 이용하여 SUBQUERY$_0$를 계산하면 $T_0$의 크기를 줄일 수 있다. 따라서 $T_0$와 $T_1$의 조인 비용을 줄일 수 있다.

NoSQL 기반 연관 콘텐츠 검색 시스템의 질의를 관계 대수로 나타내면 Fig. 6과 같다. Fig. 4는 중첩 질의의 결과인 $T_0$ 테이블과 $T_1$ 테이블을 KEYWORD_ID에 대하여 조인하는 복잡한 질의이다. Fig. 4에서는 $T_0$를 생성하는 과정에서 KEYWORD 및 YOUTUBE_URL 테이블 전체를 조인한다. 하지만 연관도가 높은

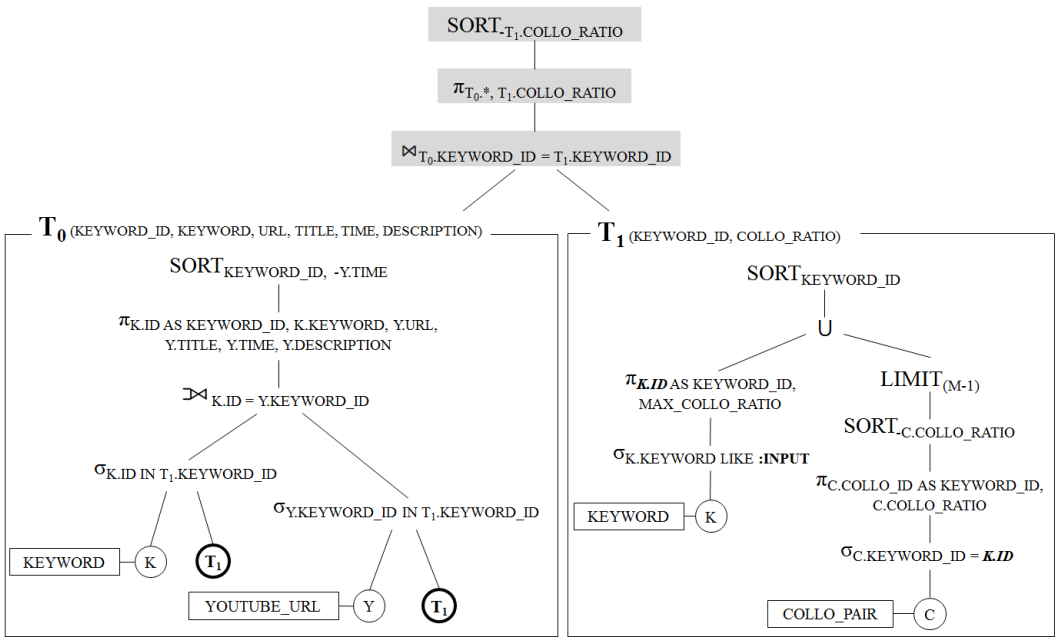



Fig. 6. The algebraic tree for searching related content stored in NoSQL.

소수의 키워드에 대한 연관 콘텐츠들만 필요하므로, 전체를 조인할 필요가 없다. 따라서 제안하는 시스템에서는 결과가 적은 $T_1$을 먼저 생성하여 연관도가 높은 M개의 키워드를 얻고, 이 M개의 키워드에 대해서만 키워드 정보와 콘텐츠 정보를 구하여 불필요한 계산을 줄인다. 이를 위해 Fig. 6에서 $T_0$ 생성 시에 $T_1$의 결과를 이용하여 KEYWORD 및 YOUTUBE_URL 테이블에서 M개의 키워드에 대한 정보만을 얻는다.

또한 성능 향상을 위하여 $T_0$와 $T_1$을 메모리상에 저장한다. Fig. 6의 질의를 처리하기 위해 $T_1$을 데이터베이스에 저장하면 $T_0$ 생성 시에 다시 읽어야하며, $T_0$와 $T_1$의 조인 연산 시에도 $T_0$와 $T_1$을 다시 읽어야 하는데, 데이터베이스에 저장된 데이터를 읽고 쓰면 많은 비용이 발생한다. 따라서 $T_0$와 $T_1$을 메모리상에 저장하여 데이터를 읽고 쓰는 비용을 줄인다.

최종적으로 생성된 $T_0$와 $T_1$에 조인 연산을 적용하는데, Phoenix는 메모리상에 저장된 $T_0$와 $T_1$을 조인하는 연산은 지원하지 않는다. 따라서 $T_0$와 $T_1$을 조인하는 연산부터는 애플리케이션 상에서 처리한다. Fig. 6에서 음영이 있는 부분은 애플리케이션 상에서 구현한 것이다. KEYWORD_ID에 대하여 정렬된 $T_0$와 $T_1$에 정렬-합병 조인 연산을 적용한다. $T_0$와 $T_1$은 각각 KEYWORD_ID에 대하여 오름차순 정렬된 상태이다. 조인을 마친 후 COLLO_RATIO에 대해 내림차순 정렬하는 데 이때 안정 정렬(stable sort) 알고리즘을 사용한다. 안정 정렬이란 서로 같은 키(key)에 대하여 원래의 순서가 그대로 유지되는 정렬 방식이다. 즉, COLLO_RATIO가 같은 경우, 이전에 KEYWORD_ID 및 TIME으로 정렬하였던 결과를 유지하기 위하여 사용한다.

### 3.2.3 연관 콘텐츠 검색 시스템 인터페이스

제안하는 연관 콘텐츠 검색 시스템의 인터페이스는 Fig. 7 및 8 같다. Fig. 7은 현재 시청 중인 콘텐츠를 대표하는 키워드가 '이승엽'일 때의 연관 콘텐츠 검색 결과이다. 입력 키워드로 '이승엽'이 입력되며, '이승엽'에 대한 연관 키워드들이 '이승엽'과 연관도가 높은 순으로 '이승엽'에 가까이 배치된다. 각 키워드를 더블클릭하면 Fig. 8과 같이 연관 콘텐츠가 재생된다.

## 4. 성능 평가

### 4.1 실험 방법

연관 키워드 추출을 위하여 2014년 1월부터 2015년 7월까지 약 60만 개(약 600백만 문장)의 프로야구 관련 인터넷 뉴스 문서[19]를 수집하였으며, 이를 통해 연관 콘텐츠를 검색하는 데 활용되는 키워드 간 연관 정보를 생성하였다. 성능 평가를 위하여 제안한 NoSQL 기반 연관 콘텐츠 검색 시스템에서 최적화하지 않은 Fig. 4의 질의와 최적화한 Fig. 6의 질의의 실행 시간을 측정하였다. 연관 콘텐츠 검색 시스템의

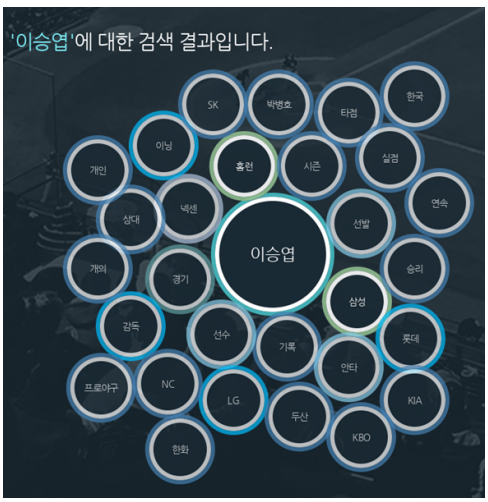



Fig. 7. The result screen of searching related content for '이승엽'.

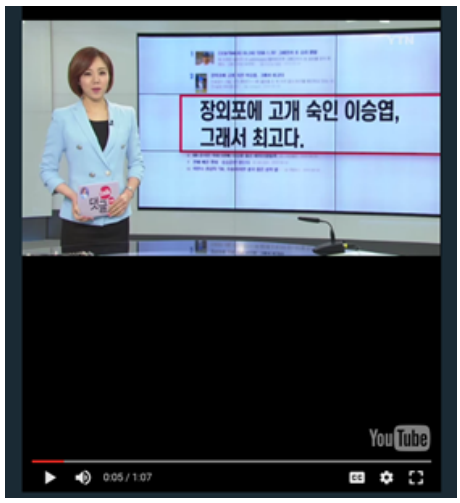



Fig. 8. The screen of playing content.

입력 키워드는 연관 키워드 추출 시스템에서 추출한 주요 키워드 중, 빈도가 높은 1,000개의 키워드를 사용하였으며, 1,000개의 키워드에 대해 최적화하기 전과 후의 질의를 각각 실행하고 실행 시간의 평균을 계산하였다.

**실험 1: 키워드 간 연관 정보의 개수를 변화시키면서 실행 시간 측정**

키워드 간 연관 정보의 개수가 작은 경우와 많은 경우의 실행 시간을 비교하기 위하여 818,258개와 4,994,092개의 키워드 간 연관 정보 데이터 집합을 생성하였다. 이때 하나의 주요 키워드 당 추출할 연관 키워드의 개수 $k$를 30개로 설정하여 실행 시간의 평균을 계산하였다.

**실험 2: 하나의 주요 키워드 당 추출할 연관 키워드의 개수 $k$를 증가시키면서 실행 시간 측정**

Fig. 4 및 6에서 $T_1$의 크기를 증가시키면서 성능 향상 비율을 측정하였다. $T_1$의 크기는 $k$에 의해 결정되므로 $k$를 초기 값 30에서 30씩 늘려가면서 성능 향상 비율을 측정하였다. 성능 향상 비율은 최적화하지 않은 질의의 실행 시간을 최적화한 질의의 실행 시간으로 나눈 값으로 계산하였으며, 4,994,092개의 키워드 간 연관 정보 데이터 집합을 사용하였다.

실험은 3개의 노드로 분산하여 진행하였으며, 1대를 네임노드(NameNode)로 설정하고, 2대를 데이터노드(DataNode)로 설정하였다. 본 실험은 Intel Core i5-4690 CPU, 8GB RAM, Samsung 850PRO 256GB SSD, CentOS 7.2 버전 OS인 컴퓨터에서 수행하였다. NoSQL 기반의 연관 콘텐츠 검색 시스템은 Hadoop 2.5.2, Zookeeper 3.4.8, HBase 0.98.22, Apache Phoenix 4.8.1을 사용하여 구현하였다.

## 4.2 실험 결과

**실험 결과 1: 키워드 간 연관 정보의 개수를 변화시키면서 실행 시간을 측정한 결과**

키워드 간 연관 정보의 개수를 변화시키면서 질의 최적화 유무에 따른 실행 시간을 측정한 결과는 Fig. 9와 같다. 키워드 간 연관 정보가 적은 경우, 최적화하지 않은 질의를 실행하면 약 1.11초가 걸리는 반면 최적화한 질의는 약 0.06초가 걸려 약 19배가 빨라졌다. 키워드 간 연관 정보가 많은 경우, 최적화하지 않은 질의는 약 5.51초, 최적화한 질의는 약 0.08초가 소요되어, 약 69배가 빨라졌다. 이를 통해 NoSQL 기반 시스템에서 질의를 최적화한 경우, 최적화하지 않았을 때와 비교하여 성능이 최대 69배까지 향상된 것을 확인할 수 있었다.

**실험 결과 2: 하나의 주요 키워드 당 추출할 연관 키워드의 개수 $k$를 증가시키면서 실행 시간을 측정한 결과**

$k$를 증가시키면서 질의 최적화 유무에 따른 실행 시간을 측정한 결과, 성능 향상 비율은 Fig. 10과 같다. 최적화하지 않은 질의에서는 $k$가 30에서 300까지 증가하는 동안 평균 실행 시간이 거의 변화하지 않은 반면, 최적화한 질의의 평균 실행 시간은 점점 증가하였다. 이에 따라 $k$가 커질수록 성능 향상 비율이 감소하였다. 이는 $k$값이 커질수록 $T_1$이 커지기 때문

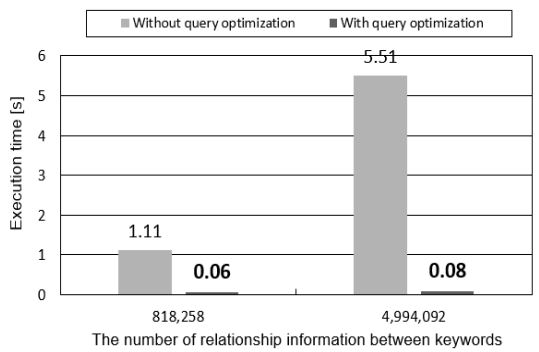



Fig. 9. Comparison of execution time with and without query optimization.

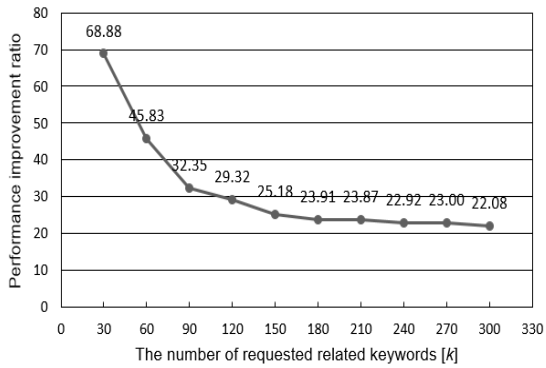



Fig. 10. The ratio of the execution time of the non-optimized query to the optimized query

에 세미조인을 이용한 최적화의 이점이 감소하기 때문이다. 일반적으로 추천 시스템에서 $k$는 작기 때문에 $k$가 클 때 성능 향상이 적은 것은 큰 문제가 되지 않는다.

## 5. 결 론

본 논문에서는 키워드 간의 연관 정보를 이용하여 시청 중인 콘텐츠와 연관된 콘텐츠를 사용자에게 추천해주는 시스템을 제안하였다. 키워드 간 연관 정보는 대량의 인터넷 뉴스 문서를 수집하여 함께 등장하는 빈도가 높은 키워드들을 분석하여 생성하였다. 제안하는 시스템의 성능을 높이기 위하여 인터넷 뉴스 문서의 전처리 과정에서 분산 및 병렬 처리를 적용하였으며, 전처리를 마친 데이터를 분석하는 과정에서도 분산 검색 엔진과 분산 데이터 처리 엔진을 사용해 키워드 간 연관 정보를 생성하는 전 과정을 분산화하였다. 또한 생성된 키워드 간 연관 정보 및 콘텐츠 정보를 NoSQL에 저장하여 분산 처리하였으며 NoSQL에 맞게 복잡한 질의를 최적화하였다. 실험을 통해 질의 최적화 유무에 따른 실행 시간을 비교한 결과, 질의를 최적화한 경우, 최적화하지 않았을 때와 비교하여 성능이 최대 69배 향상된 것을 확인할 수 있었으며, 하나의 주요 키워드 당 추출할 연관 키워드의 개수가 적을수록 성능이 좋았다.

## REFERENCE


[1] J. Son, S.B. Kim, H. Kim, and S. Cho, "Review and Analysis of Recommender Systems," *Journal of the Korean Institute of Industrial Engineers*, Vol. 41, No. 2, pp. 185-208, 2015.

[2] M. Kumar, D.K. Yadav, A. Singh, and V.K. Gupta, "A Movie Recommender System: MOVREC," *International Journal of Computer Applications*, Vol. 124, No. 3, pp. 7-11, 2015.

[3] K. Matsumura, M.J. Evans, Y. Shishikui, and A. McParland, "Personalization of Broadcast Programs Using Synchronized Internet Content," *Proceeding of International Conference on Consumer Electronics*, Vol. 4, pp. 1-5, 2010.

[4] R. Cattell, "Scalable SQL and NoSQL Data Stores," *Association for Computing Machinery Special Interest Group Management of Data Record*, Vol. 39, No. 4, pp. 12-27, 2010.

[5] S. Lee, "Personalized Contents Recommendation System Based On Social Network," *Journal of Broadcast Engineering*, Vol. 18, No. 1, pp. 98-105, 2013.

[6] M. Na and J. Lee, "Improvement of UCI Metadata and Resolution Service for Massive Contents Recommendation," *Journal of Korea Multimedia Society*, Vol. 13, No. 3, pp. 475-486, 2010.

[7] Y. Jeon, E. Kim, H. Park, and K. Lee, "A Trend Analysis Service Using a Hadoop Cluster of Mini PCs," *Proceeding of the Korea Information Processing Society Spring Conference*, Vol. 22, No. 1, pp. 710-711, 2015.

[8] H. Lee and J. Kwon, "A New Distributed Graph Data Storage System for Large-scale Recommender Engines," *Journal of Korean Institute of Information Technology*, Vol. 11, No. 7, pp. 139-149, 2013.

[9] O. Kononenko, O. Baysal, R. Holmes, and M.W. Godfrey, "Mining Modern Repositories with Elasticsearch," *Proceeding of the 11th Working Conference on Mining Software Repositories*, pp. 328-331, 2014.

[10] Apache Lucene, https://lucene.apache.org (June 29, 2017)

[11] Apache Spark, http://spark.apache.org. (June 29, 2017)

[12] YouTube, https://www.youtube.com. (June 29, 2017)

[13] Apache HBase, https://hbase.apache.org. (June 29, 2017)

[14] M.N. Vora, "Hadoop-HBase for Large-scale Data," *Proceeding of the International Conference on Computer Science and Network Technology*, pp. 601-605, 2011.

[15] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File Sys-

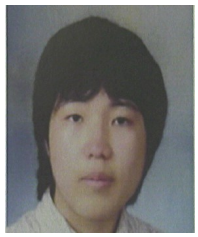
tem," *Proceeding of the Symposium on Mass Storage Systems and Technologies*, pp. 1-10, 2010.
[16] Apache ZooKeeper, https://zookeeper.apache.org. (June 29, 2017).
[17] Apache Phoenix, https://phoenix.apache.org. (June 29, 2017).
[18] P.A. Bernstein and D.W. Chiu, "Using Semi-joins to Solve Relational Queries," *Journal of the Association for Computing Machinery*, Vol. 28, No. 1, pp. 25-40, 1981.
[19] Naver news. http://news.naver.com. (June 29, 2017).

**고 은 정**

2016년 광운대학교 컴퓨터공학과 학사
2016년~현재 광운대학교 컴퓨터공학과 석사과정

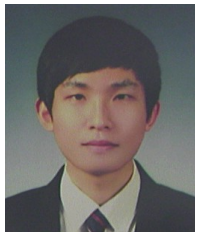
**김 호 준**

2016년 광운대학교 전자통신공학과 학사
2017년~현재 광운대학교 컴퓨터공학과 석사과정



**박 효 주**

2016년 광운대학교 컴퓨터공학과 학사
2016년~현재 광운대학교 컴퓨터공학과 석사과정



**전 영 호**

2015년 광운대학교 컴퓨터공학과 학사
2015년~2017년 8월 광운대학교 컴퓨터공학과 석사과정
2017년 8월~현재 SK 하이닉스 선임연구원

**이 기 훈**

2000년 KAIST 전산학과 학사
2002년 KAIST 전산학과 석사
2009년 KAIST 전산학과 박사
2009년~2012년 KT 종합기술원 매니저
2012년~2013년 SAP Labs Korea, Senior Developer
2013년~현재 광운대학교 컴퓨터공학과 부교수

**신 사 임**

2000년 숙명여자대학교 전산학과 학사
2002년 KAIST 전산학과 석사
2004년 KAIST 전산학과 박사 수료
2006년~현재 전자부품연구원 인공지능연구센터 책임연구원