https://doi.org/10.5392/JKCA.2017.17.11.209

시맨틱 RDF 데이터에 대한 효과적인 키워드 검색

Effective Keyword Search on Semantic RDF Data

박창섭
동덕여자대학교 컴퓨터학과

Chang-Sup Park(cspark@dongduk.ac.kr)

요약

최근 지식 베이스, 시맨틱 웹 등 여러 응용 분야에서 시맨틱 데이터의 활용이 증가함에 따라 대규모 RDF 데이터에 대한 효과적인 검색 방법의 필요성이 커지고 있다. 기존의 개별 루트 시맨틱에 기반한 키워드 검색 방법들은 서로 다른 루트 노드를 갖는 결과 트리들의 집합만을 검색함에 따라, 의미적으로 유사하거나 연관성이 낮은 결과 트리들이 함께 검색되고, 동일한 루트 노드를 공유하되 의미적으로 다르고 질의 연관도가 높은 결과들은 함께 검색될 수 없는 문제점이 있다. 이를 개선하기 위해 본 논문에서는 결과 트리들의 루트 노드의 중복을 제한적으로 허용하여 질의 연관도가 높으면서 다양한 결과들을 함께 검색하는 방법을 제안한다. 이를 위해 결과 트리 집합의 루트 중복도 척도를 정의하고, 주어진 키워드 질의와 최대 루트 중복도에 따라 제한적인 루트 중복성을 가지면서 연관도가 높은 top-k 결과 트리들을 효율적으로 구하기 위한 검색 알고리즘을 제시한다. 실 데이터를 이용한 성능 실험 결과, 제안한 방법이 기존 방법보다 콘텐트 노드들의 중복이 적은 다양한 결과 트리들을 검색할 뿐만 아니라 결과 트리들의 루트 노드의 중복을 허용함으로써 질의 연관도가 더 높은 결과들을 생성할 수 있음을 보였다.

■ 중심어 : | 시맨틱 데이터 | RDF | 키워드 검색 | top-k 질의 |

Abstract

As a semantic data is widely used in various applications such as Knowledge Bases and Semantic Web, needs for effective search over a large amount of RDF data have been increasing. Previous keyword search methods based on distinct root semantics only retrieve a set of answer trees having different root nodes. Thus, they often find answer trees with similar meanings or low query relevance together while those with the same root node cannot be retrieved together even if they have different meanings and high query relevance. We propose a new method to find diverse and relevant answers to the query by permitting duplication of root nodes among them. We present an efficient query processing algorithm using path indexes to find top-k answers given a maximum amount of root duplication a set of answer trees can have. We show by experiments using a real dataset that the proposed approach can produce effective answer trees which are less redundant in their content nodes and more relevant to the query than the previous method.

■ keyword : | Semantic Data | RDF | Keyword Search | Top-k Query |

* 본 연구는 2016년도 동덕여자대학교 학술연구비 지원에 의하여 수행된 것임
접수일자 : 2017년 07월 27일
수정일자 : 2017년 08월 22일
심사완료일 : 2017년 08월 23일
교신저자 : 박창섭, e-mail : cspark@dongduk.ac.kr

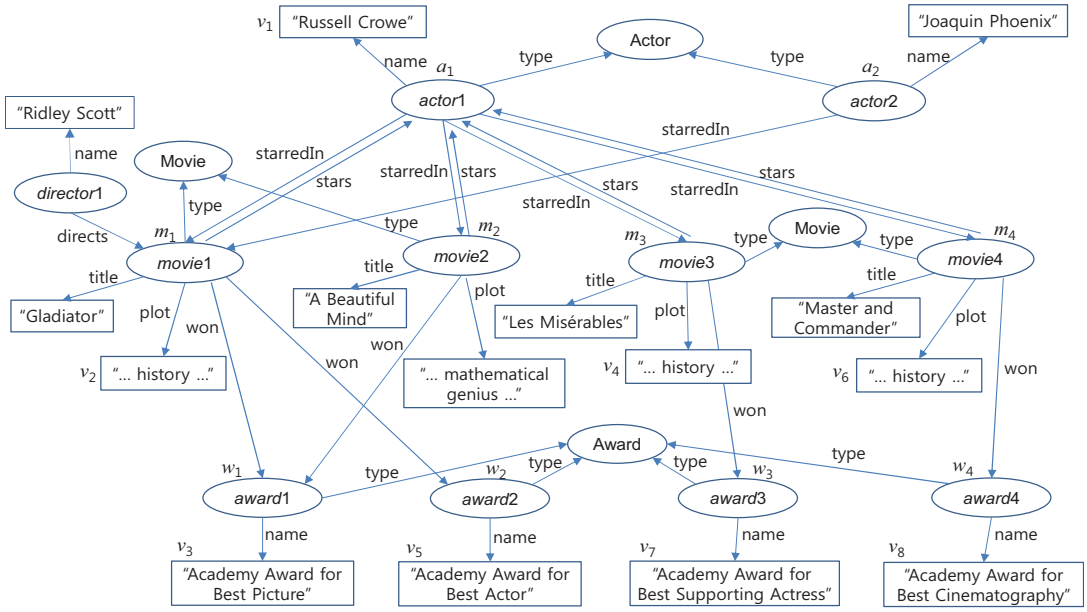


그림 1. 그래프로 표현된 시맨틱 RDF 데이터의 예

I. 서 론

최근 여러 분야에서 지식 베이스(Knowledge Base), 온톨로지(Ontology), 시맨틱 웹(Semantic Web) 등 시맨틱 데이터의 사용이 증가하고 있다. 시맨틱 데이터를 표현하기 위한 표준 방법으로는 RDF(Resource Description Framework)가 널리 사용되고 있다[1]. RDF에서는 (subject, predicate, object) 형식의 트리플(triple)을 이용하여 하나의 시맨틱 정보를 나타낸다. subject는 사물, 객체, 개념 등 리소스를 나타내고 predicate는 그 리소스에 대한 속성(property)을 뜻한다. object는 그 속성의 값으로 다른 리소스나 리터럴(literal) 값이 사용될 수 있다. 이 세 요소들은 모두 URI 형식의 ID를 갖는다. 예를 들어, 대표적인 시맨틱 데이터인 DBpedia[1]에 포함된 RDF 트리플 (http://dbpedia.org/resource/Gladiator_(2000_film), http://dbpedia.org/ontology/Starring, http://dbpedia.org/resource/Russell_Crowe)은 영화 Gladiator가 배우 Russell Crowe를 출연시킴을 의미한다. 각 트리플에 대해 subject와 object를 노드(node)로 나타내고 predicate를 두 노드를 연결하는 이름이 있는 유향 간선(labeled directed edge)으로 나타내면 모든 RDF 트리플들의 집합은 하나의 유향 그래프로 표현될 수 있다. [그림 1]은 영화에 관한 시맨틱 데이터의 일부를 그래프로 나타낸 예이다.

시맨틱 데이터가 널리 사용됨에 따라 대량의 RDF 데이터에서 원하는 정보를 쉽게 찾고 이용하기 위한 검색 기술의 필요성이 커지고 있다. RDF 데이터에 대한 검색을 위해 SPARQL과 같은 패턴 기반 질의 언어와 이를 실행하는 질의 처리 시스템들이 개발되어 사용되고 있다[2]. 그러나 사용자가 RDF 데이터의 스키마를 잘 이해하고 특정 질의 언어의 문법에 맞는 올바른 패턴 질의를 작성해야 한다는 어려움이 있다. 이에 대한 대안으로 단순한 키워드들의 리스트를 질의로 이용하는 검색 방법이 주목받고 있다. 이 방법은 하나 이상의 키워드들이 주어질 때 RDF 그래프에서 그것들을 포함하는 노드들을 찾고 그 노드들을 연결하는 서브-트리(sub-tree)나 서브-그래프(sub-graph)들을 검색하여 반환한다. 기존의 많은 연구들이 대량의 데이터에 대해 질의와 연관성이 높고 간결한 형태의 결과들을 효율적으로 구하기 위해 서브-트리 구조를 이용한다[3-11].

1 http://wiki.dbpedia.org/

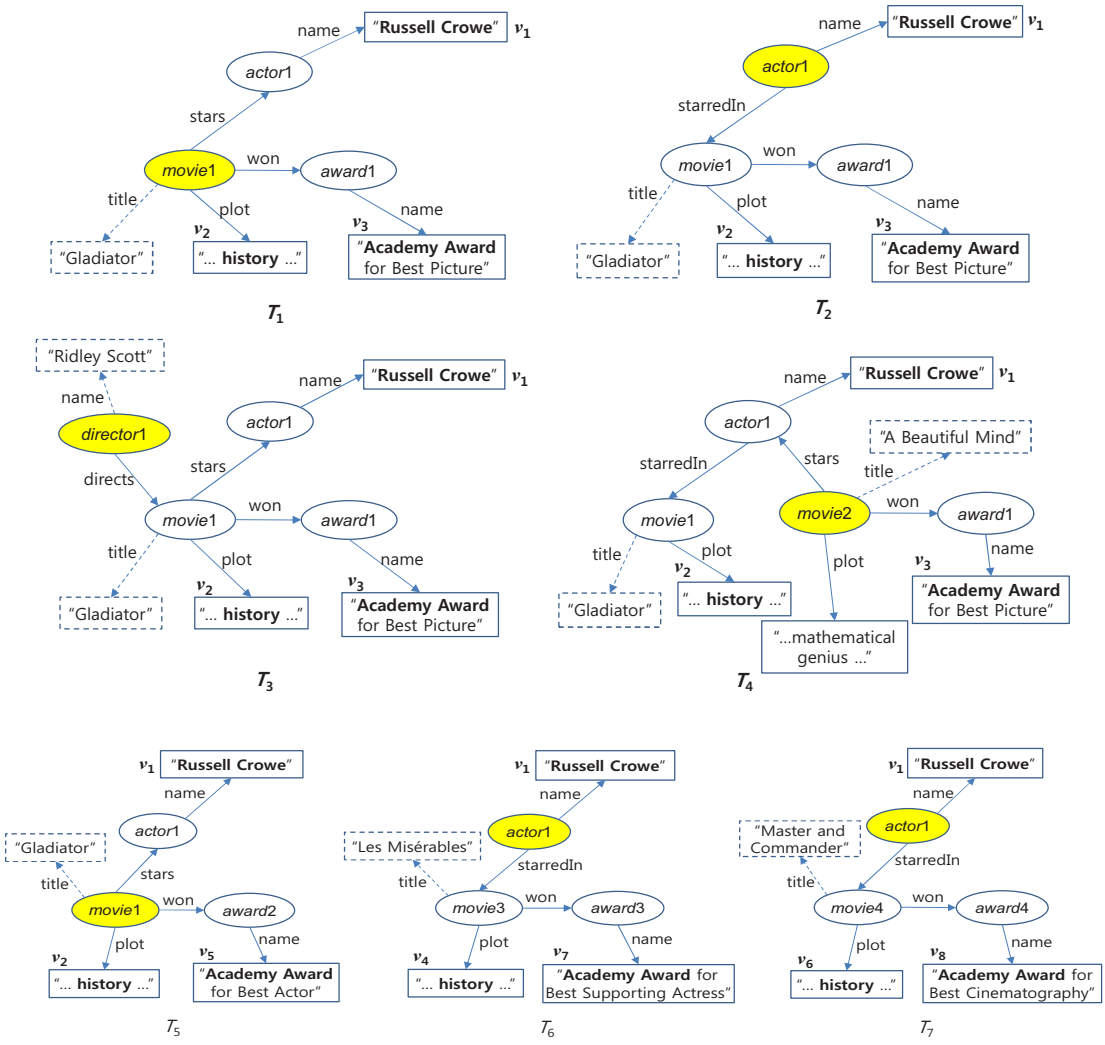


그림 2. 키워드 질의 q = {"Russell Crowe", "history", "Academy Award"}에 대한 결과 트리의 예

예를 들어, [그림 1]의 시맨틱 데이터에서 러셀 크로우, 역사 및 아카데미 상과 관련이 있는 영화 정보를 찾기 위해 키워드 리스트로 구성된 질의 q = {"Russell Crowe", "history", "Academy Award"}를 실행한다고 가정하자. [그림 2]는 이 질의에 대한 결과가 될 수 있는 서브-트리들의 예를 보인다[2]. 그래프에서 주어진 질의 키워드를 하나 이상 포함하는 노드를 콘텐트 노드(content node)라 하고, 그런 노드들을 연결하는 하나의 서브-트리를 결과 트리(answer tree)라 한다. 예를 들어 [그림 2]의 T_1은 콘텐트 노드 v_1, v_2, v_3를 포함하고 노드 m_1을 루트로 갖는 결과 트리이다. 대규모 그래프에는 하나의 질의에 대해 결과 트리들이 매우 많이 존재할 수 있으므로, 일반적으로 각 결과 트리의 질의 연관도(relevance)를 측정하여 질의와 연관성이 가장 높은 top-k 트리들을 찾아 검색 결과로 반환한다.

Top-k 결과 트리 집합을 효율적으로 구하기 위해 개별 루트 시맨틱(distinct root semantic)에 기반한 검색

2 각 결과 트리에서 색깔을 갖는 노드는 루트 노드를 의미한다. 점선으로 표시된 노드와 간선은 영화의 제목을 표시하기 위해 편의상 추가한 것으로, 결과 트리에는 속하지 않는다.

방법들이 연구 및 제안되었다[3][4][6-10]. 이 방법들은 그래프 내의 특정 노드에 대해 그것을 루트로 갖는 결과 트리들이 여러 개 존재할 때 질의에 대한 연관도가 가장 높은 것 하나만을 검색 결과의 후보로서 고려한다. 즉, top-k 결과 트리들은 항상 서로 다른 노드를 루트로 갖도록 제한된다. 이런 방법은 질의와 연관성이 높은 다양한 결과 트리들을 검색할 수 있고 특히 그래프 내에 다수의 노드들과 연결 경로가 존재하는 허브 노드가 있을 때 그것을 루트로 공유하는 많은 서브-트리들이 결과로 생성되는 것을 피할 수 있다. 또 그래프 내에서 탐색해야 할 서브-트리의 수를 줄임으로써 대규모 데이터에 대해 질의를 효율적으로 처리할 수 있다[4].

그러나 개별 루트 시맨틱은 다음과 같은 문제점을 갖고 있다. 많은 간선을 포함하는 복잡한 그래프에서 루트 노드는 서로 다르지만 콘텐트 노드 집합이 동일한 서브-트리들이 다수 존재하고 함께 검색될 수 있는데, 그런 트리들은 의미적으로 유사한 결과를 나타내어 질의 결과의 다양성을 저하시킬 수 있다. 예를 들어 [그림 2]에 도시된 질의 q에 대한 결과 트리 T_1 ~ T_4는 루트 노드는 서로 다르지만 동일한 콘텐트 노드 집합 {v_1, v_2, v_3}을 공유하는 결과 트리들이다. 이들 중 T_1 ~ T_3는 질의에 대해 같은 의미를 가진다. 즉, 배우 러셀 크로우가 역사와 관련된 특정 영화(Gladiator)에 출연했고 그 영화가 아카데미 최우수 작품상을 받았음을 나타낸다. T_3는 그 영화를 감독한 감독이 있다는 정보를 추가적으로 나타내지만 그것은 주어진 질의에 대해 큰 관련이 없다. 한편 T_4는 러셀 크로우가 출연한 다른 영화에 대한 정보를 포함하므로 T_1 ~ T_3와는 다른 구조와 의미를 갖지만, 두 키워드가 각각 서로 다른 영화와 관련되므로 질의에 대한 연관성이 낮다. 결과적으로 T_1 ~ T_4와 같은 결과 트리들은 주어진 질의에 대해 매우 유사하거나 연관성이 낮은 검색 결과를 사용자에게 제공한다.

개별 루트 시맨틱의 또 다른 문제점은 콘텐트 노드 집합이 서로 다르고 의미적으로 유사하지 않지만 동일한 노드를 루트로 갖는 결과 트리들은 질의에 대한 연관도가 높아도 함께 검색될 수 없다는 점이다. 예를 들어, [그림 2]의 T_5는 T_1과 같은 루트 노드(m_1)를 갖지만 키워드 "Academy Award"에 대한 콘텐트 노드가 다른, 즉, 다른 아카데미 상에 대한 정보를 포함하는 결과 트리이다. 또 T_6와 T_7은 T_2와 같은 루트 노드(a_1)를 갖지만 키워드 "history"와 "Academy Award"에 대해 각각 서로 다른 콘텐트 노드를 가짐으로써 역사에 관한 여러 영화들(Les Miserables, Master and Commander)과 그 영화들이 수상한 다양한 아카데미 상들(여우조연상, 촬영상)에 대한 정보를 나타낸다. 또 이 결과 트리들은 질의에 대한 연관성이 높으므로 질의 결과로 함께 검색되는 것이 바람직하다. 그러나 기존 개별 루트 시맨틱에 기반한 방법에서는 이런 검색 결과들, 즉, {T_1, T_5}나 {T_2, T_6, T_7}과 같은 결과를 생성할 수 없다.

본 논문에서는 이런 문제점들을 고려하여 결과 트리들 사이에 루트 노드의 중복을 제한적으로 허용하는 검색 방법을 제안한다. 즉, 같은 루트 노드를 공유하지만 콘텐트 노드 집합이 서로 다른 결과 트리들 중 질의 연관도가 높은 것들이 top-k 결과로 함께 선택될 수 있도록 한다. 단, 결과 트리들의 루트 노드의 중복을 일정 수준 이하로 제한함으로써 다양한 결과를 효율적으로 생성한다. 루트 노드의 최대 중복도는 검색 시스템이나 사용자가 설정할 수 있도록 함으로써 질의 처리의 유연성을 제공한다. 주어진 중복성 제약을 만족하면서 연관도가 가장 큰 top-k 결과 트리 집합을 효율적으로 찾기 위한 경로 인덱스 구조와 질의 처리 알고리즘을 제시한다.

II. 관련 연구

대규모 그래프 구조 데이터에 대한 키워드 검색에서 일반적으로 사용되는 개별 루트 시맨틱은 그래프에 속한 각 노드에 대해 그것을 루트로 갖는 서브-트리들 중 질의 연관도가 가장 큰 것만을 결과 후보로 고려한다. 이 방식에 기반한 방법들 중 양방향 탐색[3]은 결과 트리를 찾기 위해 그래프의 간선들에 대한 역방향 탐색과 순방향 탐색을 동시에 실시한다. BLINKS[4]에서는 양방향 그래프 탐색을 효율적으로 실행하기 위해 각 키워드를 포함하는 노드들에 대한 경로 인덱스를 정의하고 이를 활용한 top-k 검색 알고리즘을 제안하였다. [6]에

서는 대규모 그래프 데이터에 대해 다중-입자(multi-granular) 그래프 표현을 사용하여 그래프 탐색에 따른 I/O 횟수를 줄이는 방법을 제시하였다. [9]에서는 주어진 질의와 관련성이 높은 검색 결과를 제공하기 위해 기존 방법들보다 확장된 결과 트리 구조와 순위 척도를 정의하고 인덱스를 활용한 top-k 질의 처리 알고리즘을 제안하였다. 1장에서 언급한 바와 같이 상기 방법들은 질의 결과에 해당하는 서브-트리들의 루트 노드에 대해서는 중복을 전혀 허용하지 않지만 키워드를 포함하는 콘텐트 노드들에 대해서는 제한이 없기 때문에 콘텐트 노드들이 중복된 유사한 결과들이 생성될 수 있다. 한편 BLINKS[4] 등과 같이 경로 인덱스를 활용하는 방법에서는 개별 루트 시맨틱을 전제로 각 키워드에 대한 최적 경로만 계산하여 색인화 하므로 질의 처리 시 다양한 결과를 구하는 데 어려움이 있다.

질의 결과로 검색되는 결과 트리들 간의 구조적인 유사성은 [7]에서 처음 지적되었다. [12]에서는 결과 트리들의 콘텐트 노드들 사이의 중복성 문제를 다루었고, 주어진 질의 키워드를 포함하는 콘텐트 노드 집합들 사이에 중복이 전혀 없도록 결과 트리들을 구하되 노드들 간의 근접도가 큰 콘텐트 노드 집합들을 순서대로 찾는 알고리즘을 제안하였다. 그러나 콘텐트 노드 집합만 고려하고 이들을 연결하는 구조에 대한 검색이나 중복성은 고려하지 않았다. [10]에서는 개별 루트 시맨틱을 기반으로 연관도가 높은 서브-트리들을 찾되 축소된(reduced) 구조를 갖는 비-중복적인 결과들을 구하는 방법을 제안하였다. [11]에서는 키워드 검색 결과로 서로 중복되지 않으면서 크기가 가장 작은 서브-트리들의 집합을 찾는 방법을 제안하였다. 그러나 이 방법은 콘텐트 노드 집합뿐만 아니라 내부 노드들과 간선들 간에도 중복이 없는 서브-트리들만을 검색하여 지나치게 제한적인 검색 결과를 생성하는 문제점이 있다.

대규모 RDF 데이터에 대한 키워드 검색 방법으로 데이터 그래프를 직접 탐색하지 않고 RDF 트리플들에 대한 구조적인 요약(summary) 그래프를 생성 및 탐색하여 주어진 키워드 질의와 연관성이 높은 SPARQL 패턴 질의들을 생성하는 방법들이 제안되었다[13][14]. 이 방법들은 시맨틱 데이터의 스키마 또는 타입 정보와 SPARQL 언어를 이용함으로써 기존 RDF 저장 시스템의 검색 기능을 활용한다는 장점이 있으나 최종적으로 검색된 결과들의 중복성이나 다양성에 대한 예측과 제어가 어렵다.

III. 루트 노드의 중복을 고려한 키워드 검색의 정의

본 논문에서 질의 처리의 대상이 되는 시맨틱 데이터는 유향 그래프 $G(V, E)$로 나타낸다. V는 각 트리플의 subject와 object에 대응되는 노드들의 집합이고 E는 각 predicate에 대응되는 이름을 갖는 간선들의 집합이다. 키워드 k를 포함한 노드를 k에 대한 콘텐트 노드라 부르고 그런 노드들의 집합을 $V(k)$라 표기한다. 간선은 인접한 subject 노드와 object 노드 사이의 의미적 근접도를 나타내는 가중치를 갖고, 두 노드 사이의 경로의 길이는 그 경로에 속한 간선들의 가중치의 합으로 계산된다. 시맨틱 데이터 그래프에 대한 키워드 질의가 주어질 때 그것에 대한 결과 트리는 다음과 같이 정의된다.

정의 1. (결과 트리) 그래프 $G(V, E)$로 표현되는 시맨틱 데이터에 대한 키워드 질의 $q = \{k_1, k_2, \cdots, k_l\}$가 주어지고, q에 속한 각 키워드를 포함하는 콘텐트 노드들의 집합 $C = \{v_i \mid v_i \in V(k_i), 1 \le i \le l\}$와 노드 $r \in V$이 주어진다고 가정하자. 이 때 r에서부터 C에 속한 각 노드까지 모두 경로가 존재한다고 가정하자. q에 대한 결과 트리 $T(r, C)$는 r을 루트로 갖고 C에 속한 모든 노드를 포함하며 r에서부터 C에 속한 각 노드까지의 최단 경로를 포함하는 G의 서브-트리이다. 단, $T(r, C)$의 단말 노드는 모두 C에 속한다. □

주어진 키워드 질의에 대해 서로 다른 루트 노드와 콘텐트 노드들을 포함하는 결과 트리들이 그래프 내에 다수 존재할 수 있다. top-k 검색에서는 질의와 연관성이 가장 높은 k개의 결과들을 구하기 위해 각 결과 트리의 연관도를 측정하고 순위를 매긴다. 본 논문에서는 [9]에서 제안된 방식을 기반으로 질의 연관도를 계산한

다. 즉, 결과 트리의 질의 연관도는 각 콘텐트 노드의 키워드 연관성과, 루트에서 각 콘텐트 노드까지의 최단 경로를 통한 근접성을 기반으로 다음과 같이 정의된다.

정의 2. (결과 트리의 질의 연관도) 키워드 질의 $q = \{k_1, k_2, \cdots, k_l\}$에 대해, 루트 노드 r과 콘텐트 노드 집합 $C = \{v_i \mid v_i \in V(k_i), 1 \le i \le l\}$를 포함하는 결과 트리 $T(r, C)$의 질의 연관도는 다음과 같다.

$$rel(T(r,C),q) = Z \sum_{1 \le i \le l} rel(r, v_i, k_i)$$
$$= Z \sum_{1 \le i \le l} rel(r, v_i) rel(v_i, k_i) \quad \square$$

위 정의에서 $rel(r, v_i)$는 루트 r에 대한 콘텐트 노드 v_i의 근접도를 나타내고, r에서 v_i까지의 최단 경로의 길이에 반비례하는 함수로 정의된다. 또 $rel(v_i, k_i)$는 키워드 k_i를 포함하는 콘텐트 노드 v_i의 키워드 연관도를 나타내며 정보 검색에서 널리 사용되는 TF-IDF 방법을 기반으로 정량적으로 계산될 수 있다[15]. 이 두 값의 곱으로 정의되는 $rel(r, v_i, k_i)$는 키워드 k_i에 대한 루트-콘텐트 노드 경로의 연관도를 나타낸다. Z는 결과 트리 T의 질의 연관도를 [0, 1] 사이의 값으로 정규화하기 위한 파라미터이다. 위와 같이 정의되는 연관도 척도를 기반으로 각 질의 키워드에 대해 연관도가 가장 높은 루트-콘텐트 노드 경로들을 찾음으로써 질의 연관도가 높은 결과 트리들을 효율적으로 검색할 수 있다.

질의에 대한 결과 트리들의 집합이 주어질 때 그것의 루트 중복도는 다음과 같이 정의된다.

정의 3. (결과 트리 집합의 루트 중복도) 질의에 대한 결과 트리 집합 $A = \{T_1, T_2, ..., T_n\}$에 대해 A에 속한 트리들의 루트 노드들의 집합을 R_A라 할 때, A의 루트 중복도는 다음과 같다.

$$dup_r(A) = \frac{|A| - |R_A|}{|A| - 1} = \frac{n - |R_A|}{n - 1} \quad \square$$

즉, 루트 중복도는 같은 루트 노드를 공유하는 결과 트리들이 많을수록 커진다. 루트 중복도는 [0, 1] 범위에 속한다. 모든 결과 트리들이 서로 다른 루트 노드를 가질 경우 루트 중복도는 0이 되고 이것은 개별 루트 시맨틱을 적용한 결과와 같다. 반면, 모든 결과 트리들이 같은 루트 노드를 공유할 경우 그 값은 1이 된다.

예를 들어, [그림 2]에서 질의 q에 대한 결과 트리들 중 노드 *movie*1을 루트로 갖는 결과 트리 T_1과 T_5를 선택하고 노드 *actor*1을 루트로 공유하는 결과 트리 T_2, T_6, T_7을 선택하여 검색 결과를 생성할 경우 이 결과 트리 집합 $R = \{T_1, T_2, T_5, T_6, T_7\}$의 루트 중복도는 $dup_r(R) = (5 - 2)/(5 - 1) = 0.75$ 이다. 만일 이 트리들과 루트 노드가 서로 다른 T_4를 R에 추가할 경우 루트 중복도는 $dup_r(R \cup \{T_4\}) = (6 - 3)/(6 - 1) = 0.6$이 된다.

질의에 대한 top-k 결과 트리 집합이 가질 수 있는 루트 중복도의 최대값이 시스템에 미리 정의되거나 또는 질의와 함께 사용자로부터 주어진다고 가정하면, 이 값을 이용하여 루트 노드가 제한적으로 중복된 결과 트리 집합을 정의할 수 있다.

정의 4. (제한적인 루트 중복성을 갖는 결과 트리 집합) 키워드 질의 q와 최대 루트 중복도 r_m이 주어질 때 $(0 < r_m < 1)$, q에 대한 결과 트리들의 집합 $A = \{T_1, T_2, ..., T_n\}$가 $dup_r(A) \le r_m$을 만족하면 A를 제한적인 중복성을 갖는 결과 트리 집합이라 한다. $\square$

본 논문에서 제안하는 검색 방법은 위와 같이 루트 노드가 제한적으로 중복되면서 질의에 대한 연관성이 가장 큰 top-k 개의 결과 트리들의 집합을 찾는다.

IV. 질의 처리 방법

1. 인덱스 구조

주어진 키워드 질의를 효율적으로 처리하기 위해 시맨틱 데이터 그래프 내에 존재하는 노드-키워드 경로들에 대한 인덱스를 미리 생성한다. 경로 인덱스는 기존 연구에서 사용된 것들과 유사하게 키워드 경로들에 대한 역 리스트(inverted list)와 해시 맵(hash map)으로 구성된다[4, 9, 10]. 그러나 개별 루트 시맨틱에 기반한 방법들은 각 루트 노드에 대해 하나의 결과 트리만

고려하므로 인덱스 생성 시 각 노드에서부터 각 키워드를 포함하는 콘텐트 노드들에 대한 경로들 중 거리가 가장 짧은 것 하나만 선택하여 저장한다. 따라서 기존 인덱스로는 루트 노드를 공유하는 두 개 이상의 결과 트리들을 함께 찾을 수 없다. 본 논문에서는 루트 노드의 중복을 허용하면서 질의 연관도가 가장 높은 것들을 효율적으로 검색하기 위해 기존의 인덱스 구조를 확장한다.

주어진 그래프에 속한 노드 n과 키워드 k에 대해, n으로부터 k를 포함하는 콘텐트 노드들에 대한 최단 경로들을 구하고 그 경로들의 키워드 연관도를 계산한다. 그리고 (n, k) 쌍을 키로 사용하여 그 경로들에 대한 (v, rel) 정보를 해시 맵에 저장한다. 이 때, v는 각 경로의 마지막 노드인 콘텐트 노드를, rel은 그 경로의 키워드 연관도 값 $rel(n, v, k)$을 뜻한다(정의 2 참조). 이 구조를 노드-키워드 맵이라 하고 $NKMap$으로 나타낸다. 일반적으로 각 (노드, 키워드) 쌍에 대해 여러 콘텐트 노드들과 그것에 대한 경로들이 존재할 수 있다. 본 논문에서는 인덱스의 크기와 top-k 질의 처리의 효율성을 고려하여 각 (노드, 키워드) 쌍에 대해 일정 개수의 연관도가 가장 높은 경로들만 선택적으로 저장한다.

또한 top-k 결과 트리들을 효율적으로 탐색하기 위해 각 (노드, 키워드) 쌍에 대해 연관도가 가장 큰 경로 정보를 각 키워드에 대한 역 리스트에 나누어 저장한다. 키워드 k에 대한 역 리스트 $L(k)$는 각 노드 n으로부터 키워드 k를 포함하는 콘텐트 노드들에 대한 최적 경로를 구하고 그런 경로들을 연관도 값의 내림차순으로 정렬하여 저장한다[4]. 이와 같은 경로 인덱스를 활용함으로써 질의 처리 시 비효율적인 그래프 탐색을 피하고 연관도가 높은 결과 트리들을 효율적으로 찾을 수 있다.

2. 질의 처리 알고리즘

제안하는 질의 처리 방법은 다차원 속성을 갖는 객체들에 대한 top-k 질의 처리에 널리 사용되는 임계 알고리즘(threshold algorithm)을 기반으로 한다[16]. 앞 절에서 기술한 인덱스 구조를 활용하여 그래프 내의 모든 경로 및 서브-트리들을 탐색하지 않고 효율적으로 결과 후보 트리들을 구한다. 루트 중복도 조건을 만족하면서 질의 연관도가 가장 큰 후보 트리들을 찾고 유지함으로써 제한적인 루트 중복성을 갖는 top-k 결과 트리 집합을 생성한다. 제안하는 알고리즘은 다음과 같다.

알고리즘 1. 제한적인 중복성을 갖는 top-k 검색

Input: $q = \{k_1, k_2, \cdots, k_l\}$, k, r_m
Output: q에 대한 루트 중복성을 갖는 top-k 결과 트리 집합
Variables: 크기가 k인 우선순위 큐 Q, 노드 집합 $Closed$, 실수 배열 $curRel[1..l]$, 리스트 배열 $P[1..l]$

```
1   Q ← ∅, Closed ← ∅, curRel ← {0.0}
2   while (an entry exists in a list in L(q)) {
3     Select a list L(k_i) from L(q) in a round-robin manner.
4     Read an entry (n, rel) at the current scan
        position in L(k_i).
5     curRel[i] ← rel
6     if (n ∉ Closed) {
7       P[i] ← ∅ (1≤i≤l)
8       for-each (k_j ∈ q) {
9         Look up a list of (v, rel) for (n, k_j) in NKMap.
10:       if (no entry is found from NKMap for (n, k_j))
11:         goto line 42
12:       else store the list into P[j].
13:     }
14:     for-each (combination C = {(v_1, rel_1),
                  (v_2, rel_2), ..., (v_l, rel_l)} of the entries,
                  one from each list in P[i] (1≤i≤l)) {
15:       T_new ← T(n, C)
16:       if (|Q| < k) {
17          Q' ← Q ∪ {T_new}
18:         R_Q' ← a set of the root nodes of
                    answer trees in Q'
19:         if ((|Q'|-|R_Q'|)/(k-1) ≤ r_m) Q.insert(T_new)
20:         else {
21:           Q_s ← a set of answer trees in
                    Q∪{T_new} which share the
                    root node with other trees
22:           T' ← an answer tree in Q_s with
                    a minimum relevance score
23:           if (T' ≠ T_new)
24:             Q.replace(T', T_new)
25:         }
26:       }
27:       else {
28:         T_k ← Q.find-min()
29:         if (rel(T_new) > rel(T_k)) {
30            R_Q ← a set of the root nodes of
                    answer trees in Q
31:           if (root(T_new) ∉ R_Q or dup_r(Q) ≤ r_m - 1/(k-1))
32:             Q.replace(T_k, T_new)
33            else {
                Lines 34-37 are equal to lines 21-24.
38            }
39          }
40        }
41      }
42      Closed ← Closed ∪ {n}
43    }
44    if (|Q| = k and
        rel(Q.find_min()) ≥ Σ_{1≤i≤l} curRel[i]) break
45  }
46  Build answer trees using the entries in Q.
End
```

위 알고리즘에서는 키워드 질의 q와 함께 최대 루트 중복도 r_m이 입력으로 주어진다. q에 속한 각 키워드에 대응되는 역 리스트의 집합을 $L(q)$라 할 때, 알고리즘은 $L(q)$에 속한 리스트들을 병렬적으로 순차 스캔(scan)하면서 항목들을 하나씩 읽어 들인다(2-4행). 그 항목이 노드 n에 대해 처음으로 발견된 노드-키워드 경로를 나타내면, 모든 질의 키워드에 대해 노드-키워드 맵을 조회하여 n으로부터 각 키워드까지 연관도가 가장 큰 경로들의 집합을 구한다(6-13행). 모든 질의 키워드들에 대해 하나 이상의 경로가 존재하면 그 경로들을 조합함으로써 n을 루트로 갖는 여러 결과 트리들을 구할 수 있다(14행). 각 결과 트리가 top-k 결과의 후보가 될 수 있는지 여부를 판단하여 만일 그렇다면 기 정의된 우선순위 큐 Q에 저장하여 유지하고 그렇지 않으면 무시한다(15-40행). Q에는 질의 처리 과정에서 도출된 결과 트리들 중 루트 중복도 제약을 만족하면서 질의 연관도가 가장 큰 k개의 트리들의 정보가 저장된다. 이 때 각 트리의 질의 연관도 값을 우선순위로 사용한다.

새로운 결과 트리 T_{new}가 top-k 결과의 후보가 될 수 있는지 여부는 다음과 같이 판단할 수 있다. 만일 Q에 저장되어 있는 결과 트리의 개수가 k보다 작으면 그 결과 트리들과 T_{new}를 포함한 결과 트리 집합 $Q' = Q \cup \{T_{new}\}$에 대해 다음 조건을 검사한다.

$$dup'_r(Q') = \frac{|Q'| - |R_{Q'}|}{k-1} \leq r_m \qquad (1)$$

이 때, R_Q은 Q에 속한 결과 트리들의 루트 노드들의 집합을 가리킨다. 만일 위 조건이 만족되면 T_{new}를 Q에 추가하여 top-k 후보 트리 집합을 갱신한다(19행). 위 조건이 만족되지 않는 경우는 T_{new}의 루트 노드가 Q에 있는 기존 결과 트리들 중 하나와 중복된 경우로, 만일 T_{new}를 Q에 포함시키면 그 후 다른 어떤 결과 트리들을 추가해도 최종적인 top-k 결과 트리 집합의 루트 중복도가 r_m을 초과하게 된다. 이 때 Q에 속하고 다른 트리와 루트 노드를 공유하는 것들 중에서 질의 연관도가 가장 작은 것을 선택하여 T_{new}와 교체하면 top-k 후보 트리들을 유지할 수 있다(20-25행).

만일 Q에 k개의 top-k 후보 트리들이 존재할 경우, 다음과 같은 방법으로 새로운 결과 트리 T_{new}의 삽입 여부와 Q에서 제외시킬 결과 트리를 결정한다. 먼저 우선순위 큐 Q에 저장된 기존 top-k 결과 트리들 중 연관도가 가장 작은 트리 T_k를 찾는다. 만일 T_{new}의 연관도가 T_k보다 작으면 T_{new}는 top-k 결과에 속할 수 없으므로 배제한다(28-29행).

새로운 트리 T_{new}가 Q에 속한 어떤 트리와 루트 노드를 공유할 때 Q에서 유일한 루트 노드를 갖는 트리를 선택하여 T_{new}와 교체하면 결과 트리 집합의 루트 중복도가 증가한다. 그 외의 경우에는 Q에 속한 임의의 트리와 T_{new}를 교체해도 루트 중복도가 증가하지 않는다. 또 루트 중복도가 증가하는 경우에도 항상 기존 값보다 $1/(k-1)$ 만큼 증가하므로 기존의 루트 중복도가 $r_m - 1/(k-1)$ 보다 작거나 같으면 문제가 되지 않는다.

따라서 T_{new}가 기존 Q에 속한 어떤 트리와도 루트 노드를 공유하지 않거나 또는 Q의 루트 중복도가 $r_m - 1/(k-1)$ 보다 작거나 같으면, 즉,

$$root(T_{new}) \notin R_Q \text{ 또는 } dup_r(Q) \leq r_m - \frac{1}{k-1} \qquad (2)$$

을 만족할 경우, Q에 속한 트리들 중에서 질의 연관도가 가장 작은 것을 선택하여 T_{new}로 교체하면 루트 중복도 제약을 만족하면서 연관도가 가장 큰 새로운 top-k 결과 트리 집합이 생성된다(30-32행).

그러나 위 식 (2)를 만족하지 않을 때 T_{new}는 Q에 속한 어떤 트리와 루트를 공유하므로 앞의 식 (1)이 위배되었을 때와 같이 Q에서 다른 트리와 루트 노드를 공유하는 트리들 중 하나를 선택하여 T_{new}와 교체하면 Q의 루트 중복도가 증가하지 않는다. 따라서 그런 트리들 중 질의 연관도가 가장 작은 것을 선택하여 T_{new}와 교체하면 루트 중복도 제약을 만족하면서 질의 연관도가 가장 높은 결과 트리들을 유지할 수 있다(33-38행).

위와 같은 질의 처리 과정에서 $L(q)$에 속한 역 리스트들로부터 k개 이상의 결과 트리들이 발견되고 그것들의 연관도가 리스트들에서 아직 발견되지 않은 모든 결과 트리들의 연관도보다 크거나 같다면 우선순위 큐 Q에 포함된 결과 후보 트리들이 질의에 대한 top-k 결과 트리 집합이 된다. 알고리즘의 44행에서 이 조건을 검사하여 만족되면 질의 처리를 즉시 종료하고 top-k 결과를 사용자에게 반환한다(44-46행).

예를 들어, [그림 1]의 시맨틱 데이터에 대한 질의 q = {"Russell Crowe", "history", "Academy Award"}가 주어질 때, 질의 결과에 허용된 최대 루트 중복도(r_m)가 0.6이라고 가정하고 위 방법을 통해 top-5 결과를 구하는 과정을 살펴본다. [그림 3]은 그래프에 대해 미리 생성된 역 리스트 및 노드-키워드 맵의 일부이다. 위 알고리즘에서는 주어진 역 리스트들을 교대로 읽어 각 질의 키워드에 대한 노드-키워드 경로들을 찾는다.

Round 1. 역 리스트 L_1에서 처음 조회되는 경로 항목은 (m_1, v_2)이다. 노드 m_1과 각 질의 키워드에 대해 노드-키워드 맵을 조회하여 m_1으로부터 각 키워드까지 연관도가 가장 큰 경로들의 리스트들을 구한다(8-12행). 이 키워드 경로들을 조합함으로써 m_1을 루트로 갖는 결과 트리 $T(m_1, \{v_1, v_2, v_3\})$, $T(m_1, \{v_1, v_2, v_5\})$, $T(m_1, \{v_1, v_2, v_7\})$를 구할 수 있다. 이 세 트리는 우선순위 큐 Q에 차례대로 저장된다. 그러나 m_1을 루트로 갖는 다른 결과 트리 $T(m_1, \{v_1, v_2, v_8\})$, $T(m_1, \{v_1, v_4, v_3\})$ 등을 Q에 삽입하면 식 (1)을 위배하게 된다($dup'_A(Q')$ = (4 - 1)/(5 - 1) = 0.75 > 0.6). 따라서 Q에는 m_1이 루트이면서 질의 연관도가 가장 큰 세 개의 트리만이 저장될 수 있다.

Round 2. 다음 역 리스트 L_2에서 첫 항목 (a_1, v_1)을 읽은 후 *NKMap*을 통해 a_1을 루트로 갖는 결과 트리들을 구한다. 첫 번째 트리인 $T(a_1, \{v_1, v_2, v_3\})$은 기존 Q의 트리들과 루트 노드가 다르므로 식 (1)을 만족하고 따라서 Q에 추가된다(18행). Q의 트리들을 질의 연관도 순으로 [$T(m_1, \{v_1, v_2, v_3\})$, $T(m_1, \{v_1, v_2, v_5\})$, $T(a_1, \{v_1, v_2, v_3\})$, $T(m_1, \{v_1, v_2, v_7\})$]이 된다. 그러나 그 다음 트리 T_{new} = $T(a_1, \{v_1, v_2, v_5\})$를 Q에 추가하게 되면 $dup'_A(Q')$ = 0.75가 되므로, 앞에서 기술한 바와 같이 루트 노드를 공유하는 트리들 - 현재 Q에 있는 트리들과 새로운 트리가 모두 해당됨 - 중 질의 연관도가 가장 작은 것, 즉, $T(m_1, \{v_1, v_2, v_7\})$를 선택한다. 이것을 Q에서 삭제하고 T_{new}를 저장한다. a_1을 루트로 갖는 다른 결과 트리들이 모두 T_{new}보다 연관도가 크지 않을 경우 위 과정을 통해 제외된다. Q = [$T(m_1, \{v_1, v_2, v_3\})$, $T(m_1, \{v_1, v_2, v_5\})$, $T(a_1, \{v_1, v_2, v_3\})$, $T(a_1, \{v_1, v_2, v_5\})$]가 된다.

Inverted Lists:
L_1("history"): [(m_1, v_2), (m_3, v_4), (m_4, v_6), (a_1, v_2), ...]
L_2("Russell Crowe"): [(a_1, v_1), (m_1, v_1), (m_2, v_1), (m_3, v_1), ...]
L_3("Academy Award"): [(w_1, v_3), (w_2, v_5), (w_3, v_7), (w_4, v_8), ...]

NKMap:
(m_1, "Russell Crowe") → [(v_1, *rel*)]
(m_1, "history") → [(v_2, *rel*), (v_4, *rel*), (v_6, *rel*)]
(m_1, "Academy Award") → [(v_3, *rel*), (v_5, *rel*), (v_7, *rel*), (v_8, *rel*)]
(a_1, "Russell Crowe") → [(v_1, *rel*)]
(a_1, "history") → [(v_2, *rel*), (v_4, *rel*), (v_6, *rel*)]
(a_1, "Academy Award") → [(v_3, *rel*), (v_5, *rel*), (v_7, *rel*), (v_8, *rel*)]
(w_1, "Russell Crowe") → [ ]
(w_1, "history") → [ ]
(w_1, "Academy Award") → [(v_3, *rel*)]
(m_3, "Russell Crowe") → [(v_1, *rel*)]
(m_3, "history") → [(v_4, *rel*), (v_2, *rel*), (v_6, *rel*)]
(m_3, "Academy Award") → [(v_7, *rel*), (v_3, *rel*), (v_5, *rel*), (v_8, *rel*)]
(m_4, "Russell Crowe") → [(v_1, *rel*)]
(m_4, "history") → [(v_6, *rel*), (v_2, *rel*), (v_4, *rel*)]
(m_4, "Academy Award") → [(v_8, *rel*), (v_3, *rel*), (v_5, *rel*), (v_7, *rel*)]
...

그림 3. [그림 1]의 그래프 구조에 대한 경로 인덱스

Round 3. 세 번째 역 리스트 L_3에서 첫 번째 항목 (w_1, v_3)를 읽은 후 *NKMap*을 조회하면 w_1을 루트로 갖는 결과 트리는 존재하지 않음을 알 수 있다.

Round 4. 역 리스트 L_1에서 두 번째 항목 (m_3, v_4)을 읽는다. m_3를 루트로 갖는 첫 번째 트리 $T(m_3, \{v_1, v_4, v_7\})$을 Q에 추가할 경우 식 (1)을 만족하므로 Q에 추가한다. Q = [$T(m_1, \{v_1, v_2, v_3\})$, $T(m_1, \{v_1, v_2, v_5\})$, $T(m_3, \{v_1, v_4, v_7\})$, $T(a_1, \{v_1, v_2, v_3\})$, $T(a_1, \{v_1, v_2, v_5\})$]이 된다. 이제 $|Q| = k$이고 m_3를 루트로 갖는 다른 트리 $T(m_3, \{v_1, v_4, v_3\})$ 등은 모두 Q에 있는 트리들보다 질의 연관도가 작으므로 제외된다.

Round 5 & 6. L_2와 L_3에서 조회되는 다음 항목의 시작 노드는 각각 m_1, w_2로, 이미 앞에서 처리되었거나 결과 트리가 존재하지 않는 노드이다.

Round 7. 다시 L_1에서 세 번째 항목 (m_1, v_6)을 조회한다. m_1를 루트로 갖는 결과 트리들 중 첫 번째 트리 $T(m_1, \{v_1, v_6, v_8\})$은 Q에 있는 모든 트리들과 루트를 공유하지 않으므로 식 (2)를 만족한다. 따라서 Q에서 연관도가 가장 작은 트리 $T(a_1, \{v_1, v_2, v_5\})$을 새 결과 트리와 교체한다(31행). Q에 저장된 결과 트리 집합은 Q = [$T(m_1, \{v_1, v_2, v_3\})$, $T(m_1, \{v_1, v_2, v_5\})$, $T(m_3, \{v_1, v_4, v_7\})$, $T(m_1, \{v_1, v_6, v_8\})$, $T(a_1, \{v_1, v_2, v_3\})$]이 된다.

V. 성능 평가

본 논문에서 제안한 방법의 효과와 성능을 평가하기 위해 실 데이터를 이용하여 검색 실험을 실시하였다. 실험을 위해 IMDB[3]에서 2000~2012년 사이에 제작된 약 12만 개의 영화에 대해 영화제목, 감독, 출연배우, 장르, 플롯 등의 정보를 나타내는 약 300만 개의 트리플들을 추출하여 사용하였다. 이 시맨틱 데이터는 약 109만 개의 노드와 307만 개의 간선들로 구성된 그래프로 표현된다. 이 그래프에 포함된 키워드의 개수는 약 576만 개이고, 서로 다른 키워드는 약 35만 개이다. 실험을 위해 본 논문에서 제안한 방법과 기존의 BLINKS 방법[4]을 Java 언어로 구현하였다. 그래프 생성 및 노드들 간의 최단 경로 계산을 위해 JGraphT[4] 라이브러리를 이용하였고, TF-IDF 방법을 통해 각 노드의 키워드 연관도를 계산하기 위해 Apache Lucene[5] 라이브러리를 활용하였다. 실험은 3.0GHz Intel Xeon CPU 4개와 130GB RAM으로 구성된 SMP 서버에서 실행되었다.

표 1. 테스트 질의

ID	키워드 리스트
Q1	history, rome, action
Q2	time, travel, future
Q3	russell_crowe, rome, battle
Q4	hero, rescue, disaster
Q5	hitchcock, mystery, thriller
Q6	artificial_intelligence, robot, war
Q7	western, cowboy, sheriff
Q8	disaster, survival, nonfiction
Q9	monster, alien, animation
Q10	explosion, collapse, rescue

실험에서는 영화에 대한 다양한 정보를 검색하기 위한 50여 개의 키워드 리스트를 정의하여 본 논문에서 제안한 방법(DupRoot)과 기존의 개별 루트 시맨틱에 기반한 질의 처리 방법(DistRoot)을 각각 실행하였다. [표 1]은 두 방법에 의해 검색된 결과 트리들의 질의 연관도와 기존 검색 방법에 의한 결과 트리들의 콘텐트 노드 중복도가 상대적으로 큰 질의들을 나타낸다. [그림 4]는 제안한 방법을 통해 구한, 각 질의에 대한 top-k 결과 트리 집합에서 루트 노드가 중복된 트리들의 개수(즉, k - 서로 다른 루트 노드의 개수)를 보인다. 최대 루트 중복도가 0.5일 때 top-10, top-20, top-30 결과의 중복된 루트 노드의 평균 개수는 각각 3.6개, 8.9개, 13.7개이고 평균 루트 중복도는 각각 0.40, 0.47, 0.47이다.

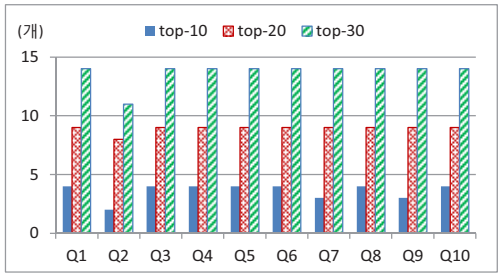
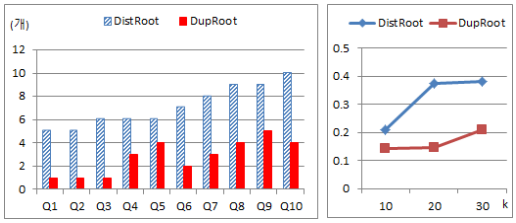


그림 4. 루트 노드가 중복된 결과 트리의 개수

[그림 5-(a)]는 두 방법으로 구한 top-20 결과 트리들 중 콘텐트 노드 집합이 다른 트리와 중복된 것들의 개수를 비교한다. 기존 개별 루트 시맨틱 기반의 방법은 평균 약 7.1개의 결과 트리가 중복된 콘텐트 노드 집합을 갖는 반면, 제안한 방법의 결과에서는 약 2.8개의 결과 트리들만이 중복된 콘텐트 노드 집합을 가진다. 평균 중복도는 각각 37.4%, 14.7%로, 제안한 방법에서 약 22.6% 포인트 감소하였다. 이것은 제안한 방법의 결과에서 루트 노드를 공유하는 트리들은 반드시 서로 다른 콘텐트 노드 집합을 갖기 때문에 콘텐트 노드 집합이 중복된 트리들이 결과로 생성될 가능성이 줄어든다. [그림 5-(b)]는 검색 방법과 검색 결과의 크기에 대해 콘텐트 노드 집합의 평균 중복도를 비교한 것으로, 모든 검색 결과 크기에 대해 제안한 방법의 결과에서 콘텐트 노드 집합의 중복이 덜 발생함을 알 수 있다.

[그림 6-(a)]는 각 테스트 질의에 대해 두 방법에 의한 top-20 결과들의 평균 질의 연관도 값을 비교한다. 각 질의에 대해 제안한 방법으로 검색한 결과의 평균 질의 연관도가 기존 방법에 의해 생성된 결과보다 모두

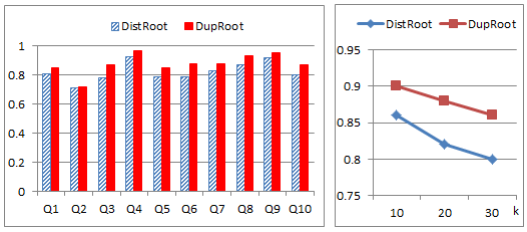
3 http://www.imdb.com/
4 http://jgrapht.org/
5 http://lucene.apache.org/



(a) 중복된 콘텐트 노드 집합을 갖는 트리의 개수(k = 20)

(b) 콘텐트 노드 집합의 평균 중복도

그림 5. 검색 결과의 콘텐트 노드 집합의 중복도



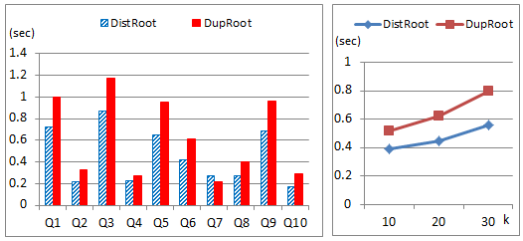
(a) top-20 검색 결과의 평균 질의 연관도

(b) 검색 결과 크기에 대한 평균 질의 연관도

그림 6. 검색 결과의 질의 연관도

높게 나타났다. 전체 질의에 대한 평균 연관도 값은 기존 방법보다 약 8.3% 증가하였다. 이는 제안한 방법에서 같은 루트 노드를 공유하지만 질의에 대한 연관성이 큰 결과 트리들이 함께 검색될 수 있도록 허용했기 때문이다. [그림 6-(b)]는 각 검색 결과 크기 k에 대해 검색 결과의 평균 질의 연관도를 비교한다. 검색 결과의 크기가 증가함에 따라 평균 질의 연관도는 감소하는 경향을 보이나, 모든 크기에서 제안한 방법의 검색 결과의 연관도가 기존 방법의 결과보다 더 높음을 알 수 있다.

[그림 7-(a)]는 각 질의에 대해 두 방법으로 top-20 결과를 구하는데 소요된 실행 시간을 나타내고, [그림 7-(b)]는 각 방법과 결과 크기에 대한 평균 질의 처리 시간을 계산하여 비교한 것이다. 그림에 나타난 바와 같이 본 논문에서 제안한 방법이 기존 방법에 비해 좀 더 많은 시간이 걸렸다. top-20 검색의 경우 평균 처리 시간이 기존 방법보다 약 37.5% 증가했다. 이것은 제안한 방법이 같은 노드를 루트로 공유하는 여러 결과 트리들을 함께 검색하기 위해 더 많은 결과 후보 트리들을 탐색하고 비교하는 데서 비롯된 결과이다.



(a) top-20 검색 실행 시간

(b) 검색 결과 크기에 대한 평균 검색 시간

그림 7. 검색 실행 시간

VI. 결론

본 논문에서는 시맨틱 RDF 데이터에 대한 키워드 질의에 대해 제한적인 루트 중복성을 가지면서 연관도가 높은 결과 트리들을 효율적으로 검색하기 위한 방법을 제안하였다. 결과 트리의 질의 연관도와 결과 트리 집합의 루트 중복도 척도를 정의하고, 대규모 데이터에 대한 효율적인 검색을 위한 경로 인덱스를 정의하였다. 이를 활용하여 키워드 질의에 대해 루트 노드가 제한적으로 중복되고 연관도가 높은 결과 트리 집합을 구하는 top-k 질의 처리 알고리즘을 제시하였다. 제안한 방법은 사용자가 루트 중복성을 제어할 수 있고 루트 노드들의 제한적인 중복 하에 다양하고 연관도가 높은 결과 트리들을 효율적으로 구할 수 있는 장점이 있다.

주어진 질의에 대한 결과 트리들의 다양성과 질의 연관성은 모두 사용자의 검색 만족도를 충족시키기 위해 필요하나 서로 상충(trade-off) 관계가 있다. 본 연구에서는 결과 트리들의 루트 노드의 중복을 허용하되 결과들의 다양성 확보를 위해 최대 중복도에 대한 제한을 이용하였다. 보다 효과적인 검색을 위해 검색 결과의 구조적인 중복도와 질의 연관도를 유연하게 조정하거나 균형을 맞추어 최적의 결과 트리 집합을 도출하는 방법에 대한 연구를 후속 연구로 진행할 계획이다. 또 그래프에서 동일한 루트를 공유하는 결과 트리들을 효율적으로 찾기 위한 색인 방법과 콘텐트 노드들 간의 중복성을 고려한 검색 방법에 관한 연구가 필요하다.

참 고 문 헌

[1] RDF - Semantic Web Standards, https://www.w3.org/2001/sw/wiki/RDF

[2] SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/

[3] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar, "Bidirectional expansion for keyword search on graph databases," Proc. of the 31st Int. Conf. on VLDB, pp.505-516, 2005.

[4] H. He, H. Wang, J. Yang, and P. S. Yu, "BLINKS: ranked keyword searches on graphs," ACM SIGMOD Conference, pp.305-316, 2007.

[5] B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin, "Finding top-k min-cost connected trees in databases," Proc. of ICDE, pp.836-845, 2007.

[6] B. B. Dalvi, M. Kshirsagar, and S. Sudarshan, "Keyword search on external memory data graphs," Proc. of the VLDB Endowment, Vol.1, No.1, pp.1189-1204, 2008.

[7] K. Golenberg, B. Kimelfeld, and Y. Sagiv, "Keyword proximity search in complex data graphs," Proc. of ACM SIGMOD Conference, pp.927-940, 2008.

[8] J. X. Yu, L. Qin, and L. Chang, "Keyword search in relational databases: a survey," Bulletin of the IEEE CS Technical Committee on Data Engineering, Vol.33, No.1, pp.67-78, 2010.

[9] C. Park and S. Lim, "Efficient processing of keyword queries over graph databases for finding effective answers," Information Proc. and Management, Vol.51, No.1, pp.42-57, 2015.

[10] 박창섭, "그래프 데이터에 대한 비-중복적 키워드 검색 방법," 한국콘텐츠학회논문지, 제16권, 제6호, pp.205-214, 2016.

[11] C. Liu, L. Yao, J. Li, R. Zhou, and Z. He, "Finding smallest k-Compact tree set for keyword queries on graphs using mapreduce," World Wide Web, Vol.19, No.3, pp.499-518, 2016.

[12] M. Kargar, A. An, and X. Yu, "Efficient duplication free and minimal keyword search in graphs," IEEE Trans. on Knowledge and Data Engineering, Vol.26, No.7, pp.1657-1669, 2014.

[13] T. Tran, S. Rudolph, P. Cimiano, and H. Wang, "Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data," Proc. of the 25th ICDE, pp.405-416, 2009.

[14] W. Le, F. Li, A. Kementsietsidis, and S. Duan, "Scalable Keyword Search on Large RDF Data," IEEE Trans. on Knowledge and Data Engineering, Vol.26, No.11, pp.2774-2788, 2014.

[15] S. Buttcher, C. Clarke, and G. Cormack, *Information retrieval: implementing and evaluating search engine*, MIT Press, 2010.

[16] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," Journal of Computer and System Sciences, Vol.66, No.4, pp.614-656, 2003.

저 자 소 개

박 창 섭(Chang-Sup Park) 정회원

- 1995년 2월 : KAIST 전산학과 (공학사)
- 1997년 2월 : KAIST 전자전산학과(공학석사)
- 2002년 2월 : KAIST 전자전산학과(공학박사)
- 2002년 3월 ~ 2005년 2월 : KT 책임연구원
- 2005년 3월 ~ 2009년 2월 : 수원대학교 교수
- 2009년 3월 ~ 현재 : 동덕여자대학교 컴퓨터학과 교수

<관심분야> : 데이터베이스, 정보 검색, 시맨틱 웹