임베딩 기반 인덱스 구조에 대한 검색 성능 평가

황아영, *이다현, **이용주
경북대학교
ayxxng73@gmail.com, *dahyun0917@naver.com, **yongju@knu.ac.kr

Search Performance Evaluation for Embedded-based Index Structure

Ayeong Hwang, *Dahyun Lee, **Yongju Lee
Kyungpook National University

요 약

최근 시맨틱 웹은 Linked Open Data (LOD)의 사용으로 웹 분야에서 주목을 받고 있다. 이에 LOD 등을 중심으로 검색 고도화 연구가 활발히 수행되고 있다. 그러나 LOD 클라우드를 이용한 효율적인 검색 방법이나 활용 방안을 위한 깊이 있는 연구는 상대적으로 매우 부족한 상황이다. 따라서 본 논문에서는 LOD 클라우드를 효율적으로 구성하기 위한 인덱스 구조를 제안하고자 HYBRID R*-tree 인덱스 구조와 단일 인덱스 구조의 성능을 비교하여 평가한다.

1. 서론

최근 “한국판 뉴딜 정책”의 대표 과제인 “데이터 댐” 사업을 통해 공공 데이터 개방 및 AI 학습용 데이터 구축을 촉진하고 있으며, 해외의 경우에도 Open API 프로젝트 등을 통해 엄청난 규모의 초 대용량 빅데이터(BigData) 클라우드를 구축하고 있다. 그러나 많은 예산으로 구축된 오픈 데이터들에 대한 효율적인 검색 방법이나 활용 방안을 위한 깊이 있는 연구는 상대적으로 매우 부족한 상황이다.

또한 시맨틱 웹이 Linked Open Data(LOD)의 사용으로 웹 분야에서 주목을 받고 있는 실정에서, 시맨틱 검색시스템에서 방대한 양의 데이터를 효과적으로 활용하여 LOD 클라우드를 효율적으로 구성하기 위해서는 데이터에 접근하고 검색하는데 걸리는 시간을 단축시키는 것이 중요하다.

이에 본 논문에서는 B+-tree 와 K-dimensional tree 의 단일 인덱스 구조와 R*-tree 에 연결리스트를 결합시킨 새로운 HYBRID 인덱스 구조[1]의 검색 성능을 평가하여 가장 성능이 좋은 인덱스 구조를 제안한다. 특히, 시맨틱 검색을 위해 사용되는 다양한 임베딩 기법을 적용한 모델(즉, Hash, TransR[2], DistMult[3], ConvE[4])을 디자인하여 Chain 쿼리와 Star 쿼리에서의 검색시간을 비교하고자 한다

2. 관련연구

2.1 지식 임베딩 방법의 분류

임베딩의 목적은 다양한 소스로부터 축적한 시맨틱 검색 정보를 사용하여 검색결과를 향상시키는 지식 그래프의 내부 구조를 유지시키고 연속적인 벡터 공간에 트리플, 즉 주어, 목적어, 서술어 데이터 집합들의 연산을 단순화하는 것이다. 이는 벡터화된 데이터들로 데이터 간의 관계를 추출하고 비슷한 데이터 개체끼리 분류하여 지식 그래프를 구축하는데 널리 사용된다. 임베딩 방법은 세 가지 모델로 나뉘는데, 주로 Translational distance model 을 대표하는 TransE[5], TransD[6], TransR[2]이 있고, Semantic matching model 에는 대표적으로 RESCAL[7]과 DistMult[3]가 있으며, Deep learning model 을 대표하는 ConvE[4]와 ConvKB[8]가 있다.

전후 단어들을 참고해서 중심단어의 의미를 예측하는 word2vec[9]에서 영감을 받아 지식 그래프 임베딩에 translation invariance 를 도입한 TransE 임베딩 모델이 제안되었다. 이때 translation invariance 는 input 의 위치가 달라져도 output 이 동일한 값을 갖는 것을 말하며 이를 지식 그래프 임베딩에 도입함으로써 어느 위치에 단어가 있어도 전후 단어를 참고하여 해당 단어의 의미를 예측할 수 있다. TransE 는 대규모 지식 그래프 임베딩에서 큰 발전을 이루었지만 1-N, N-1

및 N-N 과 같은 복잡한 관계를 처리하는 데 여전히 어려움이 있다.

DistMult model 은 head 와 tail 로 대각 행렬을 사용해 대칭관계만 모델링할 수 있어 일반 지식 그래프에서는 적합하지 않다. Complex 는 대칭 및 비대칭 관계를 처리하기 위해 실수 및 가상 부분같은 복잡한 임베딩을 활용하여 DistMult 를 확장한 것이다. 이는 계산 복잡성을 감소시킬 뿐만 아니라 엔티티 표현 능력도 향상시킨다.

ConvE 는 CNN(Convolutional Neural Network)을 도입하여 주어, 목적어, 서술어의 잠재적인 의미 정보를 포착한다. 합성곱층(Convolutional layer)은 semantic matching model 에 비해 지식 그래프의 특징을 추출하는 데 우월하다.

2.2 인덱스 구조

QUAD[10]는 6 개의 B+tree 인덱스를 사용하여 RDF 데이터를 저장하고 검색한다. B+tree 인덱스는 조인 쿼리의 16 가지 접근 패턴을 모두 포함한다. 이 방식을 적용하면 모든 트리플 페턴을 빠르게 검색할 수 있다. 그러나 단일 인덱스 구조로 접근 패턴을 검색하기 위해 각 트리플이 사전에 6 번 인코딩 되기 때문에 QUAD 는 스토리지와 삽입 속도가 낭비된다

MIDAS-RDF[11]는 K-dimensional tree 를 기반으로 하는 분산 인덱스 구조이다. 다차원 인덱스 구조를 통해 오버레이 네트워크에서 효율적인 범위 검색을 할 수 있다. MIDAS-RDF 는 다양한 패턴 쿼리를 다차원 범위 쿼리로 전환하여 빠르게 RDF 트리플 검색을 할 수 있다. 또한, 라벨링 기법을 사용하여 많은 비용이 드는 transitive closure 연산을 효율적으로 처리함으로써 성능을 개선한다. 그러나 분산 RDF 저장소를 공유하고 쿼리하며, 동기화하는 데 한계가 있다.

새로 제시된 HYBRID 인덱스 구조는 Sorted Lists 와 R*-tree 를 결합함으로써 연결된 데이터를 쿼리하는 검색 방법이다. R*-tree 는 공간 데이터에 대한 다차원 인덱스 구조이다. R*-tree 를 이용하여 불필요한 데이터 스캐닝을 빠르게 제거함으로써 효율적으로 조인 쿼리를 처리할 수 있다. HYBRID 인덱스 구조는 다차원 히스토그램을 확장한 디스크 기반의 3D R*-tree 와 플래시 메모리 기반의 데이터 저장 구조를 결합함으로써 여러 리소스에 분산되어 있는 상호 연결된 데이터를 효율적으로 검색할 수 있다. 이 방법은 다수의 잘못된 히트를 신속하게 제거하기 때문에 조인 쿼리 처리 성능이 크게 향상된다.

3. 실험 방법

3.1 실험 환경

이 논문의 목적은 검색 정확도의 가장 우수한 성능을 가진 조인 쿼리를 찾는 것이다. 임베딩 및 해시 기반의 QUAD, DARQ[12], Midas, 그리고 HYBRID 를 비교하였다. Quad 는 기존의 중앙 집중의 접근 방식이고, Darq 는 실시간 탐색인 분산 접근 방식이며 Midas 는 인덱스 기반 접근 방식, Two-step 은 R*-tree 와 k-d tree 를 결합한 HYBRID 접근 방식이다.

CPU	3.6GHz Intel i7 CPU
Memory	8GB
Language	python, java
OS	Windows 10

<표 1> 실험 환경

SPARQL 을 쿼리 그래프로 변환하여 쿼리 처리를 수행할 수 있다. 쿼리 그래프 구조를 기반으로 SPARQL 쿼리는 7 개의 다른 조인 유형[13]으로 나누어질 수 있다. star 쿼리는 같은 주어 또는 목적어를 사용하는 트리플 형태의 집합이다. 이 논문에서는 같은 주어만을 사용하는 것을 고려해 star 쿼리의 모든 트리플이 동일한 주어를 가지게 한다(only subject-subject join). chain 쿼리와 directed cycle 쿼리는 주어와 목적어가 연결된 트리플 패턴이다(only subject-object join). tree 쿼리와 cycle 쿼리는 주어와 주어가 연결되고 주어와 목적어가 연결된 트리플 집합이다(subject-subject 와 subject-object). single 쿼리는 하나의 주어 또는 목적어로 이루어져있다(only subject 또는 object). complex 는 star 쿼리와 chain 쿼리의 결합으로 무작위로 조합되었다.

3.2 평가 기준

본 연구에서의 실험은 실제 benchmark 데이터 셋을 기반으로 하는 모든 조인 쿼리 유형들을 포함한다. 또한 결과를 얻기 위해 Lehigh university benchmark(LUBM) 데이터 셋을 사용한다. LUBM dataset 은 230,061 의 트리플, 38,334 개의 주어, 17 개의 서술어, 29,635 개의 목적어를 포함하며 총 36.7MB 이다. 하지만 LUBM 쿼리의 버전은 2009 년 이후에 업데이트되지 않았다. 쿼리의 복잡성이 증가함에 따라, 이전 버전은 모든 조인 쿼리 유형을 지원할 수 없다. 이 문제를 해결하기 위해 모든 조인 쿼리 유형을 지원하도록 기존 version 을 수정한다. 본 연구에서는 먼저 Q1 부터 Q15 까지 15 개의 쿼리를 디자인한다. 모든 조인 쿼리 유형들을 나열해 조인 쿼리 성능을 비교한다. SPARQL 조인 쿼리 유형은 star(Q1, Q3, Q4, Q5, Q11, Q13), chain(Q6, Q10), directed cycle(Q2), cycle(Q15), tree(Q9), single(Q14)와 complex(Q7, Q8, Q12)이다.

각 조인 쿼리 성능을 비교하기 위해 SPARQL 조인 쿼리 유형들을 나열한 후 실험한다. 검색 시간이 짧을수록 조인 쿼리의 성능이 더 우수하다. 따라서 해당 논문에서는 일반적인 star 와

chain 유형의 검색 성능을 비교했다. 그림 1 은 두 유형의 조인 성능을 보여준다. 실험 결과를 보면 상대적으로 R*-tree 가 B+-tree 와 k-dimensional tree 보다 검색하는데 적은 시간이 걸리는 것을 확인할 수 있다. 또한 B+-tree 보다는 k-d tree 가 검색 시간이 조금 길다. 따라서 R*-tree 의 성능이 가장 좋고 k-d tree 의 성능이 상대적으로 좋지 않은 것을 확인할 수 있다.

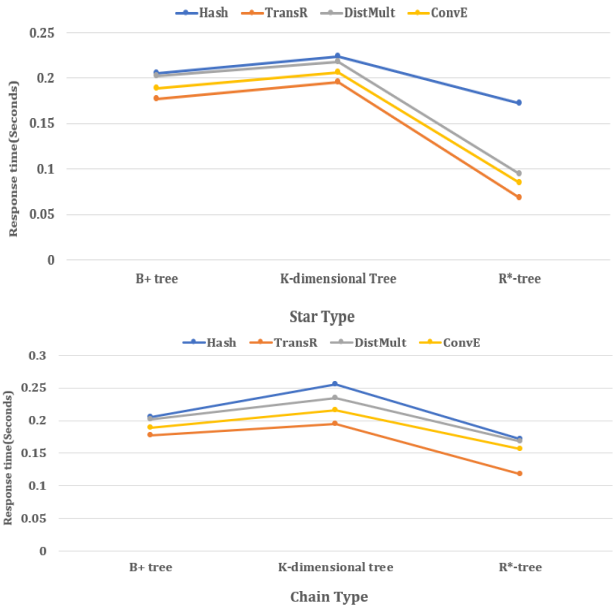

그림 1. 조인 검색 성능 비교

4. 결론

최근 시맨틱 웹은 Linked Open Data (LOD)의 사용으로 웹 분야에서 주목을 받고 있다. 이에 관한 다수의 기술과 방법론의 최적화를 위해서 LOD 클라우드를 효율적으로 구성하는 것이 필요하다. 이에 이전 연구에서 R*-tree 를 이용한 새로운 HYBRID 인덱스 구조를 제안하였고, 본 논문에서 해당 인덱스 구조의 성능을 기존의 단일 인덱스 구조와 성능을 비교하였다.

Hash, TransR[2], DistMult[3], ConvE[4]의 임베딩 방법을 적용했을 때, B+-tree, K-dimensional tree, R*-tree 중에 가장 우수한 성능의 인덱스 구조를 판별하기 위해 실험을 하였다. LUBM 쿼리를 수정해서 Q1 부터 Q15 까지 15 개의 쿼리를 디자인하여 실험을 진행하였다. SPARQL 조인 쿼리 유형들(star, chain, directed cycle, cycle, tree, single, complex) 중 star 쿼리 유형과 chain 쿼리 유형의 검색 성능을 비교하였다.

실험 결과 R*-tree 가 B+-tree 와 K-dimensional tree 에 비해 상대적으로 우수한 검색 성능을 가지고 있는 것을 확인할 수 있었다. 또한 B+-tree 보다는 k-d tree 가 검색 시간이 약간 늦다. 따라서 R*-tree 의 성능이 가장 좋고 k-d tree 의 성능이 상대적으로 좋지 않은 것을 확인할 수 있다. 결과적으로 R*-tree 의 인덱스구조를 사용함으로써 LOD 조인 쿼리의 검색 성능을 효과적으로 향상시킬 수 있다. 하지만 실험 환경의 데이터 모델에 따라서 결과는 상이할 수 있다. 따라서 다음 연구에서는 우리는 더 많은 인덱스 구조를 적용하여 조인 쿼리의 성능을 향상시키고 LOD 클라우드 최적화에 기여할 것이다.

감사의 글

이 논문은 2016 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2016R1D1A1B02008553). 본 연구는 2021 년 과학기술정보통신부 및 정보통신기획평가원의 SW 중심대학사업의 연구결과로 수행되었음(2021-0-01082).

참고문헌

[1] Sun, Yu Xiang, Lee, Yong-Ju, Storage and Retrieval Solution based on RDF Data Cloud. (2019)

[2] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu and Xuan Zhu, Learning Entity and Relation Embeddings for Knowledge Graph Completion, Proc. the National Conference on Artificial Intelligence, (2015). 2181-2187.

[3] Bishan Yang1, Wen-tau Yih, Xiaodong He, Jianfeng Gao and Li Deng, Embedding entities and relations for learning and inference in knowledge bases, Proc. the International Conference on Learning Representations. (2015), 1-12.

[4] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp and Sebastian Riedel, Convolutional 2D Knowledge Graph Embeddings, Proc. 32nd AAAI Conference on Artificial Intelligence. (2018), 1811-1818.

[5] Antoine Boreds, Nicolas Usunier, Alberto Garcia-Buran, Translating Embeddings for Modeling Multi-relational Data, Proc. CNRS. (2013).

[6] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu and Jun Zhao, Knowledge Graph Embedding via Dynamic Mapping Matrix, Proc. 53rd Annual Meeting of the Association for Computational Linguistics. (2015), 687-696.

[7] Maximilian Nickel, Volker Tresp and Hans-Peter Kriegel, A Three-Way Model for Collective Learning on Multi-Relational Data Maximilian, Proc. 28th International Conference on Machine Learning. (2011), 809-816.

[8] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen and Dinh Phung, A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network,

Proc. the North American Chapter of the Association for Computational Linguistics. (2018), 327-333.

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, Efficient estimation of word representations in vector space, Proc. ICLR Workshop. (2013), 1-13.

[10] Harth, A.; Decker, S. Optimized Index Structures for Querying RDF from the Web. In Proceedings of the 3rd Latin American Web Congress (LA-Web), Washington, DC, USA, 1 October-2 November 2005; pp. 71-81.

[11] Tsatsanifos, G.; Sacharidis, D.; Sellis, T. On Enhancing Scalability for Distributed RDF/S Stores. In Proceedings of the 14th International Conference on Extending Database Technology, Uppsala, Sweden, 21-24 March 2011; pp. 141-152.

[12] Quilitz, B.; Leser, U. Querying Distributed RDF Data Sources with SPARQL. In Proceedings of the 5th European Semantic Web Conf. (ESWC), Lecture Notes in Computer Science, Canary Islands, Spain, 27 June 2008; Volume 5021, pp. 524-538.

[13] http://downloads.dbpedia.org/2016-04/