하둡 클러스터 기반의 대용량 정성 공간 추론기의 설계

김종환, 김종훈, 김인철
경기대학교 컴퓨터과학과
e-mail:{click7254, badboyjh, kic}@kgu.ac.kr

Design of a Large-Scale Qualitative Spatial Reasoner Based on Hadoop Clusters

Jonghwan Kim, Jonghoon Kim, Incheol Kim
Dept of Computer Science, Kyonggi University

요 약

본 논문에서는 대규모 분산 병렬 컴퓨팅 환경인 하둡 클러스터 시스템을 이용하여, 공간 객체들 간의 위상 관계를 효율적으로 추론하는 대용량 정성 공간 추론기를 제안한다. 본 논문에서 제안하는 공간 추론기는 추론 작업의 순차성과 반복성을 고려하여, 작업들 간의 디스크 입출력을 최소화할 수 있는 인-메모리 기반의 아파치 스파크 프레임워크를 이용하여 개발하였다. 따라서 본 추론기에서는 추론의 대상이 되는 대용량 공간 지식들을 아파치 스파크의 분산 데이터 집합 형태인 PairRDD와 RDD로 변환하고, 이들에 대한 데이터 오퍼레이션들로 추론 작업들을 구현하였다. 또한, 본 추론기에서는 추론 시간의 많은 부분을 차지하는 이행 관계 추론에 필요한 조합표를 효과적으로 축소함으로써, 공간 추론 작업의 성능을 크게 향상시켰다. 대용량의 공간 지식 베이스를 이용한 성능 분석 실험을 통해, 본 논문에서 제안한 정성 공간 추론기의 높은 성능을 확인할 수 있었다.

1. 서론

최근 들어 링크드 오픈 데이터(Linked Open Data, LOD)의 급격한 증가와 이들을 효과적으로 이용하기 위한 지오-시맨틱 웹(Geo-Sementic Web) 기술의 발전에 따라, 공간 추론 기술과 공간 질의 처리 기술에 대한 연구들도 활발히 진행되고 있다. 공간 추론 기술은 공간 지식베이스의 무결성 점검뿐만 아니라, 새로운 지식의 유도를 통해 공간 지식베이스의 확장과 질의 응답에도 매우 유용하게 활용될 수 있는 기술이다. 그동안 개발되어온 대표적인 정성 공간 추론기들로는 SOWL[1], PelletSpatial[2], CHOROS[3], QUSAR[4], MRQUSAR[5] 등이 있다. SOWL은 시맨틱 웹 추론 규칙 언어인 SWRL로 구현한 시공간 추론기이다. 한편, PelletSpatial은 효율성이 높은 경로 일관성(path-consistency) 알고리즘을 채용한 위상 관계 공간 추론기이며, CHOROS는 방향 관계도 추론 가능하도록 PelletSpatial을 확장한 공간 추론기이다. QUSAR도 PelletSpatial을 확장하여 위상 및 방향 관계 추론이 가능한 공간 추론기이며, 위상 및 방향 관계 서로 간의 상호 교차 일관성 검사 기능도 추가된 공간 추론기이다. 앞서 언급한 시스템들은 모두 단일 컴퓨터 환경에서 동작하기 때문에, 웹 스케일의 대용량 공간 지식을 추론하기에는 성능 면에서 한계가 존재한다. 한편, MRQUSAR [5]는 이러한 한계성을 극복하기 위해 대규모 분산 병렬 컴퓨팅 환경인 하둡(Hadoop) 클러스터 시스템과 맵리듀스(MapReduce) 프로그래밍 환경을 이용해 개발된 대용량 정성 공간 추론기이다. 하지만, 일반적으로 정성 공간 추론 작업은 단위 추론 작업들 간의 높은 순차성과 반복성을 필요로 한다. 따라서 작업들 사이에 많은 디스크 입출력을 요구하는 맵리듀스의 한계성으로 인해, MRQUSAR는 추론 성능 향상에 제한이 있다.

본 논문에서는 대규모 분산 병렬 컴퓨팅 환경인 하둡 클러스터 시스템을 이용하여, 공간 객체들 간의 위상 관계를 효율적으로 추론하는 대용량 정성 공간 추론기를 제안한다. 본 논문에서 제안하는 공간 추론기는 추론 작업의 순차성과 반복성을 고려하여, 작업들 간의 디스크 입출력을 최소화할 수 있는 인-메모리 기반의 아파치 스파크(Apache Spark) 프레임워크를 이용하여 개발한다. 따라서 본 추론기에서는 추론의 대상이 되는 대용량 공간 지식들을 아파치 스파크의 분산 데이터 집합 형태인 PairRDD와 RDD로 변환하고, 이들에 대한 데이터 오퍼레이션들로 추론 작업들을 구현한다. 또한, 본 추론기에서는 공간 추론 작업의 성능 향상을 위해, 추론 시간의 많은 부분을 차지하는 이행 관계 추론에 필요한 조합표를 효과적으로 축소한다. 본 논문에서 제안하는 정성 공간 추론기의 성능을 분석하기 위해 대용량의 공간 지식베이스를 이용한 성능 분석 실험을 수행하고, 그 결과를 소개한다.

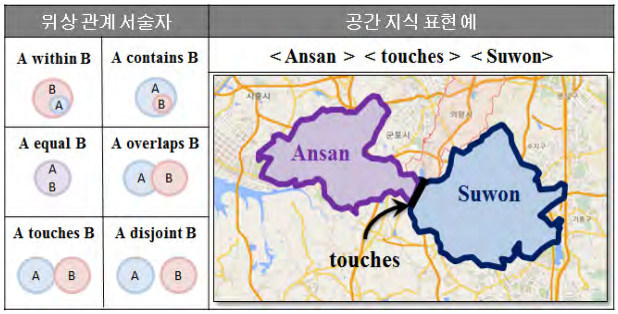
2. 정성 공간 추론

2.1 공간 지식 표현

정성 공간 추론을 하기 이전에 공간 지식을 표현법에 대해서 정의할 필요가 있다. 공간 지식은 시맨틱 웹 표준 온톨로지(ontology) 언어인, RDF/OWL에 따라 주어(subject), 서술자(property), 목적어(object) 형태의 트리플 문장(triple statement)들의 집합으로 표현된다. 주어와 목적어는 실세계에서 도시, 도로, 건물 등과 같은 특정한 장소를 의미하는 공간 객체이다. 서술자는 (그림 1)에서 보는 것과 같이 공간 객체들 간의 위상 관계를 나타낸다. 위상 관계는 OGC 표준의 Simple Feature 이론을 기초로 포함(contains), 분리(disjoint), 겹침(overlaps) 등과 같이 총

※ 본 연구는 미래창조과학부 및 정보통신기술연구진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음.
[10044494, WiseKB: 빅데이터 이해 기반 자가학습형 지식베이스 및 추론 기술 개발]

6가지의 위상 관계를 포함한다. 예컨대 “안산과 수원은 서로 접해있다.”라는 지식은 <Ansan> <touches> <Suwon> 과 같이 트리플 형태의 문장으로 표현 할 수 있다.



(그림 1) 위상 관계 서술자 및 공간 지식 표현 예

2.2 공간 추론 규칙

정성 공간 추론은 기존의 공간 지식들로부터 공간 추론 규칙을 적용하여 새로운 공간 지식을 이끌어내는 것이며, 추론 규칙에 따라 새로운 공간 지식을 얻을 수 있다.

<표 1> 공간 추론 규칙

Reasoning Rules	If	Then
Equality Rule	<s><p><o>	<s><equal><s>
		<o><equal><o>
Inverse Rule	<s><p><o> (where p are inverse of q)	<o><q><s>
Transitive Rule	<a><p_1><b> and <b><p_2><c> (where p_1 and p_2 are topological properties)	<a><[p_1 | p_2 | ... | p_n]><c> (where p_1 ,... p_n are topological properties, n <6)

본 논문에서 적용한 공간 추론 규칙은 <표 1>과 같이, 동일 관계 추론 규칙(Equality rule), 역 관계 추론 규칙(Inverse rule), 이행 관계 추론 규칙(Transitivity rule) 등이 있다. 동일 관계 추론 규칙은 기존 공간 지식을 이루는 주어와 목적어에 해당하는 공간 객체 각각에 대해서 동일 관계(equal)를 가지는 지식을 유도하는 것이다. 역 관계 추론 규칙은 기존 공간 지식을 이루는 주어와 목적어에 순서를 바꾸었을 때, 동일한 의미를 가지는 지식을 유도하는 것이다. 마지막으로 이행 관계 추론 규칙은 기존의 a와 b간의 지식과 b와 c간의 지식이 존재할 때, a와 c간의 새로운 공간 관계를 가지는 지식을 유도하는 것이다.

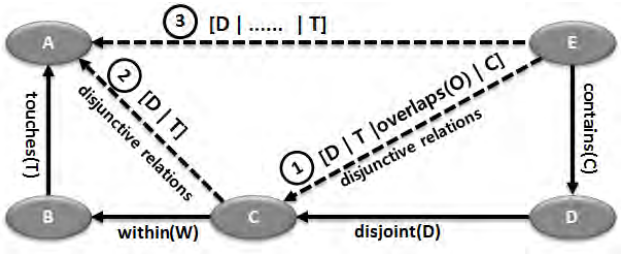
<표 2> 위상 관계 조합 표

	D	T	O	W	C	E
disjoint (D)	[D | T | O | W | C | E]	[D | T | O | W]	[D | T | O | W]	[D | T | O | W]	D	D
touches (T)	[D | T | O | C]	[D | T | O | W | C | E]	[D | T | O | W]	[T | O | W]	[D | T]	T
overlaps (O)	[D | T | O | C]	[D | T | O | C]	[D | T | O | W | C | E]	[O | W]	[D | T | O |C]	O
within (W)	D	[D | T]	[D | T | O | W]	W	[D | T | O | W | C | E]	W
contains (C)	[D | T | O | C]	[T | O | C]	[O | C]	[O | W | C | E]	C	C
equal(E)	D	T	O	W	C	E

예를 들어 “A는 B를 포함한다.” <A> <contains> <B>와 “B는 C와 겹쳐있다.” <B> <overlaps> <C>로부터 이행 관계 추론을 통해 “A는 C와 겹쳐있거나 혹은 C를 포함한다.” 라는 <A> <[overlaps | contians]> <C>를 지식을 유도해낸다. 이행 관계 추론을 통해 새로 유도 되는 위상 관계는 <표 2>와 같은 조합표로 정의할 수 있다. 즉, <표 2>에서 가로 행의 서술자 contains와 세로 열의 서술자 overlaps에 대응되는 새로운 위상 관계 서술자인 [overlaps | within]을 표에서 검색해냄으로써, A와 C의 위상 관계가 [overlaps | within]인 공간 지식을 유도할 수 있다.

2.3 공간 추론의 효율화

공간 추론의 경우에는 생성된 지식을 이용하여 또 다른 지식을 생성한다. 예를 들면 (그림 2)와 같이 공간 추론을 통해 생성된 지식 <E> <[disjoint(D) | touches(T)> <C> 인 ①과 <C> <[disjoint(D) | touches(T) | overlaps(O) | contains(C)]> <A>인 ②처럼 두 공간 객체 사이의 공간 관계가 두 개 이상의 공간 서술자로 구성될 수 있는데, 이를 이접 관계(disjunctive relations)를 가지는 지식이라고 한다. 공간 추론은 이러한 ①과 ②를 가지는 지식을 이용하여 ③과 같은 새로운 지식을 추론해야한다. 따라서 효율적인 공간 추론을 위해서 이러한 이접 관계 서술자를 가지는 지식들 간의 추론을 효과적으로 다룰 수 있어야한다.



(그림 2) 공간 추론 예시

기존의 이행 관계 추론 방법은 크게 두 가지로 나눌 수 있다. 첫 번째 방법은 (그림 2) 의 공간 지식 ①, ②의 이접 관계를 구성하는 disjoint(D), touches(T)등과 같은 모든 기본 공간 서술자들 각각에 대해 <표 2>와 같은 조합표를 적용해 조합된 공간 관계들을 구한 후, 이들에 다시 합집합 연산을 수행함으로써 원하는 최종 공간 관계 서술자를 계산하는 것이다. 이 방법은 이접 관계 서술자를 구성하는 모든 공간 서술자들에 대해서 각각 조합 시 매번 조합표를 검색하고, 조합 결과를 다시 합집합 해야 하므로, 계산량이 많은 단점이 있다. 두 번째 방법은 조합 표를 확장하여 모든 이접 관계 서술자들에 대해서 추론을 수행하는 것이다. 위상 관계를 나타내는 모든 이접 관계 공간 서술자 집합의 개수는 기본 공간 서술자의 부분 집합의 개수인 2^6개와 같다. 이를 바탕으로 조합 표를 구성할 경우 표의 크기가 2^6*2^6 이므로, 조합표의 복잡성도 높아지고 조합표를 저장하기 위한 메모리 사용량도 커져, 공간 추론 성능 개선에 어려움이 있다.

<표 3> 축소된 이접 관계 공간 서술자 집합

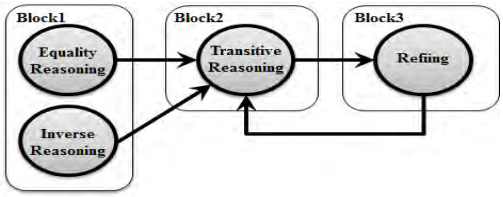
이접 관계 공간서술자 집합
[contains], [overlaps], [within], [touches], [disjoint], [equal], [contains, overlaps], [disjoint | overlaps], [disjoint | touches], [overlaps | within], [overlaps | within | touches], [contains | overlaps | touches], [disjoint, overlaps | within | touches], [contains | overlaps | within | equal], [disjoint | contains | overlaps | touches], [contains | overlaps | within | touches | equal], [disjoint | contains | overlaps | within | touches | equal]

본 논문에서는 앞서 언급한 문제점을 해결하기 위해, 가능한 모든 이접 관계들 중에서 이행 관계 추론을 통해 한 번 이상 등장한 이접 관계들을 기준으로 축소 된 조합 표를 구성하였다. 축소된 이접 관계 서술자 집합은 <표 3>과 같으며, 기존의 $2^6 = 64$ 개의 이접 관계 서술자 집합에서 17개로 이접 관계 서술자 집합이 축소하였다. 이를 통해 조합 표 크기가 기존의 $2^6*2^6 = 64*64$ 개에서 $17*17$로 축소되어, 이행 관계 추론 시 빠른 속도로 추론을 가능하게 하였다.

3. 인-메모리 기반의 대용량 공간 추론기 설계

3.1 공간 추론 작업 순서

순차 처리와 반복 작업으로 이루어진 대용량 공간 추론기를 설계하기 위해 가장 먼저 공간 추론 규칙들 간의 의존성을 파악하여 추론 작업 순서를 정해야한다.



(그림 3) 공간 추론의 의존성

(그림 3)은 공간 추론 규칙들 간의 의존성을 나타낸 그림이다. 동일 관계 추론(Equality Reasoning)은 지식 베이스에 포함된 공간 객체의 개수에 따라 얻어질 수 있는 지식의 수가 미리 정해지고, 역 관계 추론(Inverse Reasoning)이나 이행 관계 추론(Transitive Reasoning) 후에도 새로운 공간 객체가 신규로 등장할 가능성이 없다. 또한 동일 관계 추론을 통해 얻어진 지식에 대해서 역 관계 추론 시 나오는 지식이 자기 자신이므로, 동일 관계 추론과 역 관계 추론 간의 의존성이 존재하지 않는다. 따라서, 두 추론 간의 어느 것이 먼저 수행되어도 상관없다. 이행 관계 추론의 경우, 동일 및 역 관계 추론을 통해 얻어진 지식이 이행 관계 추론에서 이용되어 새로운 지식을 유도하게 되므로, 이 작업들 보다 나중에 수행되어야한다. 특히 동일 및 역 관계 추론을 먼저 수행한 후, 이행 관계 추론을 하게 되며, 추론 된 지식의 역 관계 추론 결과도 함께 생성되므로 추가적으로 역 관계 추론이 요구되지 않는다. 또한 동일 관계 추론 통해 생성 된 지식과 이행 관계 추론을 수행하면 기존의 지식들이 생성된다. 따라서, 이전의 지식 베이스와 새로 생성된 지식 베이스 간의 합집합(union) 연산이 필요 없다. 그러므로 불필요한 추론 연산이 줄어들게 되어, 추론 작업의 효율성을 높였다. 그리고 추론 작업으로 생성된 지식들 간의 불확실성을 줄이기 위해 이행 관계 추론 후에는 관계 정제(Refining) 작업을 수행해야한다. 마지막으로 새롭게 추론된 지식이 있는지 판단하고, 추론된 지식이 있다면 이행 관계 추론과 관계 정제 작업을 반복하고, 추론된 지식이 없다면 추론을 마친다.

3.2 동일 및 역 관계 추론

먼저, 동일(equality) 및 역(inverse) 관계 추론 전에 초기 공간 지식들을 키(key)를 <서술자>로 값(value)을 <주어, 목적어>로 하는 인-메모리 기반의 분산 데이터 집합인 PairRDD로 변환한다. 그 후, MapToPair 오퍼레이션을 각각 적용하여 동일 관계들과 역 관계를 가지는 지식을 생성한다. 예를 들어 <contains, <A, B>> (“A는 B를 포함한다.“) 라는 지식에 대해 동일 관계들인 <equal, <A, A>>, <equal, <B, B>>와 역 관계인 <within, <B, A>>를 각각 생성한다.

Algorithm 1. Equality Reasoning

Input : OriginalPairRDD <property, <subject, object>
Output : SubjectPairRDD <equal, <subject, subject>>,
ObjectPairRDD <equal, <object, object>>

```
1. key(property) -> key*(equal)
2. if(flag == True){
3.       value(subject, object) -> value*(subject, subject)
4.       return <key*, value*>
5. }else{
6.       value(subject, object) -> value*(object, object)
7.       return <key*, value*>
8. }
```

(그림 4) 동일 관계 추론 알고리즘

동일 관계 추론 작업의 알고리즘은 (그림 4)와 같다. 먼저 인-메모리 기반의 분산 데이터 집합인 초기 지식(OriginalPairRDD)으로부터 키(key)인 <서술자>를 동일 관계 서술자인 <equal> 로 변경한다. 그리고 <주어, 목적어> 형태로 이루어진 값(value)을 <주어, 주어> 형태로 변환한다. 반대로, <목적어, 목적어> 형태로 변환하여 <equal, <주어, 주어>>, <equal, <목적어, 목적어>> 인 <키, 값> 형태로 변환한다.

Algorithm 2. Inverse Reasoning

Input : OriginalPairRDD <property, <subject, object>
Output : InversePairRDD <inverseProperty, <object, subject>>

```
1. key(property) -> tempProperty
2. inverse(tempProperty) -> inverseProperty
3. key(property) -> key*(inverseProperty)
4. value(subject, object) -> value*(object, subject)
5. return <key*, value*>
```

(그림 5) 역 관계 추론 알고리즘

그리고 역 관계 추론 작업의 알고리즘은 (그림 5)와 같다. 동일 관계 추론 작업과 동일하게 우선 초기 지식(OriginalPairRDD)으로부터 키인 <서술자>를 <역 관계 서술자> 로 변환한다. 그리고 <주어, 목적어> 형태로 이루어진 값을 <목적어, 주어> 형태로 변환한다. 마지막으로 동일 및 역관계 추론을 통해 생성된 지식을 이행 관계 추론 및 관계 정제 작업에 이용하기 위해 합집합(Union) 오퍼레이션을 이용하여 데이터 집합(UnionPairRDD)을 확장한다. 여기서 주어와 목적어는 공간 객체이며, 서술자는 위상 관계 서술자를 의미한다.

3.3 이행 관계 추론

먼저 이행 관계 추론 전에 동일 및 역관계 추론을 통해 확장된 지식 베이스로부터 이행 관계 추론 조건을 성립하는 지식들을 찾기 위한 조인(Join) 오퍼레이션을 먼저 수행한다. 이는 추론이 가능한 지식들 끼리 동일한 노드의 효과적으로 지식 분할을 하므로, 추론 작업을 위한 네트워크 통신을 줄일 뿐만 아니라, 분산 처리의 장점을 최대한 살리는 장점이 있다. 그 후, MapToPair 오퍼레이션을 이용하여 공간 지식들 간의 이행 관계를 생성한다. 예를 들어, <contains, <A, B>>, 와 <contains, <B, C>> 라는 두 공간 지식에 대해서 <contains, <A, C>> (“A는 C를 포함한다.”)라는 새로운 지식을 생성한다.

Algorithm 3. Transitive Reasoning

Input : JoinPairRDD <<subject or object>, <<object^1, property^1>, <subject^2, property^2>>>
Output : TransitivePairRDD <property, <subject^2, object^1>>

```
1. value((object1, property1), (subject2, property2))
2. -> value(subject2, object1)
3. compose(property2, property1) -> composedProperty
4. key(subject1 or object2) -> key*(composedProperty)
5. return <key*, value*>
```

(그림 6) 이행 관계 추론 알고리즘

앞서 언급했던 것처럼 먼저 이행 관계 추론 전의 조인 오퍼레이션을 하기 위해서는 확장된 데이터 집합(UnionPairRDD)으로부터 MapToPair 오퍼레이션를 이용하여, 키인 <서술자>를 <주어> 또는 <목적어>로 변환한다. 그리고 <주어, 목적어>로 이루어진 값을 <목적어, 서술자> 또는 <주어, 서술자> 형태로 변환한다. 그 후, 조인 오퍼레이션을 통해 <<주어 or 목적어>, <<목적어^1, 서술자^1>, <주어^2, 서술자^2>>> 와 같은 형태의 동일한 키를 가지고 지식 쌍을 값으로 가지는 데이터 집합(JoinPairRDD)으로 변환한다. 마지막으로 조인 오퍼레이션을 통해 얻어진 데이터 집합에 MapToPair 오퍼레이션을 적용하여 이행 관계 추론을 수행한다. 이행 관계 추론 작업의 알고리즘은 (그림 6)과 같다. 키를 서술자^2, 서술자^1 들을 축소된 조합표를 바탕으로 구한 새로운 공간 서술자(composedProperty)로 변환한다. 그리고 값은 <주어^2, 목적어^1>로 변환한다.

3.4 관계 정제

먼저 이행 관계 추론 전에 추론 작업을 통해 얻어진 지식 가운데 관계 정제 조건을 성립하는 지식들을 찾기 위한 그룹화(Grouping) 연산을 먼저 수행한다. 관계 정제 작업에 대상이 되는 지식은 2개 이상이 될 수 있으므로, 조

인 오퍼레이션 대신 그룹화(GroupByKey) 오퍼레이션을 이용하여 각각의 노드에 효과적으로 지식을 분할한다. 따라서, 관계 정제 작업시 분산 처리의 장점을 최대한으로 살릴 수 있다. 그 후, MapToPair 오퍼레이션을 적용하여 공간 지식들 간의 불확실성을 줄여준다. 예를 들어, <contains, <A, B>>, <[contains|overlaps], <A, B>> 라는 두 공간 지식에 대해서 관계 정제를 통해 <contains, <A, B>> 라는 하나의 공간 지식을 생성한다.

앞서 언급했던 것처럼 먼저 관계 정제 작업 전의 그룹화 오퍼레이션을 하기위해 이행 관계 추론을 통해 얻어진 데이터 집합(TransitivePairRDD)로부터 MapToPair 오퍼레이션을 이용하여, 키인 <서술자>를 <주어+목적어>로 변환한다. 그리고 <주어, 목적어>로 이루어진 값을 <서술자, <주어, 목적어>> 형태로 변환한다. 구 후, 그룹화 오퍼레이션을 통해 <<주어+목적어>, iterable<서술자, <주어, 목적어>>>와 같은 형태의 데이터 집합(GroupPairRDD)으로 변환한다.

Algorithm 4. Refining

Input : GroupPairRDD <<subject, object>, Iterable values values = <property, <subject, object>>
Output : RefinedPairRDD <property, <subject, object>>

```
1. value(iterable values) -> value*(subject, object)
2. loop(iterable values){
3.       // values.property Type = Long
4.       tempProperty = min(values.property)
5. }
6. key(subject, object) -> key*(tempProperty)
7. return <key*, value*>
```

(그림 7) 관계 정제 작업 알고리즘

마지막으로 그룹화 오퍼레이션을 통해 얻어진 데이터 집합에 MapToPair 오퍼레이션을 이용하여 관계 정제 작업을 수행한다. 관계 정제 작업의 알고리즘은 (그림 7)과 같다. 각각의 서술자들은 서로 간의 연관성이 존재한다. 이를 바탕으로 이접 관계를 구성하는 기본 공간 서술자가 많을수록 인덱스 값을 크게 지정하여, 관계 정제 작업 시 가장 작은 인덱스 값을 가지는 서술자를 정제된 지식에 서술자(tempProperty)로 정하였다. 이는 공통된 주어와 목적어를 가지는 이접 관계들 간의 공통부분을 구하기 위한 교집합 연산을 간소화 시키므로 추론 작업시 효율적인 연산을 할 수 있다.

4. 실험 및 성능 평가

본 논문에서는 공간 객체들 간의 위상 관계를 효율적으로 추론하는 대용량 정성 공간 추론기의 성능을 평가하기 위해서, 대용량의 가상 지식 베이스와 실세계 공간 지식 베이스인 XB(ExoBrain) 온톨로지를 이용하여 성능 평가 실험을 수행하였다. 또한 실험 환경은 1개의 마스터(master) 노드와 9개의 슬레이브(slave) 노드로 구성되어 있으며, 각 노드는 3.5GHz, 4Core Cpu와 8GB 메인 메모리, 1TB 하드 디스크로 구성하였다.

<표 4> 추론 된 지식 수 및 추론 시간 비교

Ontology	Number of Initial Knowledge	Number of Derived Knowledge	Reasoning Time (min)	
			MRQUSAR	Our System
SG1000K	1,000,000	7,183,160	31.7	2.9
SG2000K	2,000,000	14,384,766	36.6	6.9
SG3000K	3,000,000	21,609,782	41.6	13
SG4000K	4,000,000	28,783,002	48.2	19
SG5000K	5,000,000	36,019,500	54	26

첫 번째 실험에서는 대용량의 가상 지식 베이스 양을 증가하면서 본 논문에서 제안하는 대용량 공간 추론기의 추론된 지식 수, 맵리듀스 프레임워크 기반의 대용량 정성 공간 추론기인 MRQUSAR와 추론 시간 비교를 통한 성능 분석 실험을 수행하였다. 실험 결과는 <표 4>와 같이 약 500만개의 공간 지식으로부터 새로 추론된 지식이 약 3,600만개 넘는 것을 확인 할 수 있었다. 또한, 추론 시간의 경우에는 기존의 맵리듀스 프레임워크 기반의 MRQUSAR[5]보다 약 평균 28분 정도 빠른 것을 확인 할 수 있었다. 이를 통해 본 논문에서 제안하는 대용량 정성 공간 추론기는 뛰어난 지식 베이스 확장 능력과 빠른 공간 추론 능력을 확인 할 수 있었다.

(그림 8) 추론 된 지식 검증

두 번째 실험에서는 실세계 공간 지식 베이스인 XB 온톨로지를 이용하여 제안하는 공간 추론기를 통해 추론된 지식이 올바른 지식인지를 검증하기 위한 실험을 수행하였다. (그림 8)은 공간 추론 결과의 일부분을 보여주고 있다. (그림 8)에서와 같이 "미국이 세인트루이스를 포함한다." <xbr:미국> <xbp:sp_contains> <xbr:세인트루이스>와 "미국이 브롱크스를 포함한다." <xbr:미국> <xbp:sp_contains> <xbr:브롱크스> 그리고, <xbr:미국> <xbp:sp_contains> <xbr:프리들리> 라는 추론된 지식이 있을 때, 실제 지도와 비교를 통해 추론 된 지식이 참(true)임을 확인 할 수 있다. 이를 통해 본 논문에서 제안하는 공간 추론기가 올바른 공간 관계 지식들을 추론해 낼 수 있음을 확인 할 수 있다.

5. 결론

본 논문에서는 하둡 클러스터 시스템을 이용한 대용량 정성 공간 추론기를 제안하였다. 추론 작업의 순차성과 반복성을 고려하여, 작업들 간의 디스크 입출력을 최소화할 수 있는 인-메모리 기반의 아파치 스파크 프레임워크를 이용하여 개발하였다. 또한, 본 추론기에서는 추론 시간의 많은 부분을 차지하는 이행 관계 추론에 필요한 조합표를 효과적으로 축소함으로써, 공간 추론 작업의 성능을 크게 향상시켰다. 향후 연구에서는 좀 더 효율적인 분산 데이터 오퍼레이션 기능들을 제공하는 아파치 스파크 라이브러리들을 이용해, 공간 추론기의 성능을 보다 개선시켜볼 계획이다.

참고문헌

[1] S. Batsakis and E.G.M. Petrakis, "SOWL: A Framework for Handling Spatio-Temporal Information in OWL 2.0." Rule-Based Reasoning, Programming, and Applications. Springer Berlin Heidelberg, pp. 242-249, 2014.

[2] M. Stocker and E. Sirin, "PelletSpatial: A Hybrid RCC-8 and RDF/OWL Reasoning and Query Engine." OWLED, 2009.

[3] G. Christodoulou, "CHOROS: A Reasoning and Query Engine for Qualitative Spatial Information", Dissertion Thesis, Technical University of Crete, Greece, 2011.

[4] S. Nam and I. Kim, "A Qualitative Spatial Reasoner Supporting Cross-Consistency Checks between Directional and Topological Relations", Journal of KIISE : Computing Practices and Letters, Vol.20. No.4, pp.248-252, 2014.

[5] S. Nam and I. Kim, "Design and Implementation of Large-Scale Spatial Reasoner Using MapReduce Framework", Transactions on KIPS : Software and Data Engineering, Vol.3, No.10, pp. 397-406, 2014.