시그니처 트리를 사용한 의미적 유사성 검색 기법
(Semantic Similarity Search using the Signature Tree)

김 기 성 † 임 동 혁 † 김 철 한 †† 김 형 주 †††
(Kisung Kim) (Donghyuk Im) (Cheolhan Kim) (Hyoungjoo Kim)

요 약 온톨로지의 활용이 늘어나면서 의미적 유사성 검색에 대한 관심이 높아지고 있다. 본 논문에서는 질의 객체와의 의미적 유사성이 높은 객체를 검색하는 최근접 질의 기법을 제안하였다. 의미적 유사성을 측정하는 유사성 함수로는 최적 대응값 방식의 유사도 함수를 사용하였으며 주석 정보에 대한 색인을 위해 시그니처 트리를 사용하였다. 시그니처 트리는 집합 유사성 검색에서 많이 사용되는 색인 구조로서 유사성 검색에 사용하기 위해서는 검색시 각 노드를 탐색하였을 때 발견할 수 있는 유사도의 최대값을 예측할 수 있어야 한다. 이에 본 논문에서는 최적 대응값 방식의 유사도 함수에 대한 예측 최대값 함수를 제안하고 올바른 예측 함수임을 증명하였다. 또한 시그니처 트리에 동일한 시그니처가 중복되어 저장되지 않도록 구조를 개선하였다. 이는 시그니처 트리의 크기를 감소시킬 뿐만 아니라 질의 성능 또한 향상시켜 주었다. 실험의 데이타로는 대용량 온톨로지와 주석 정보 데이타를 제공하는 Gene Ontology(GO)를 사용하였다. 실험에서는 제안한 방법의 성능 향상 외에도 페이지 크기와 노드 분할 방법이 의미적 유사성 질의 성능에 미치는 영향에 대해 알아보았다.

키워드 : 의미적 유사성 검색, 시그니처 트리

Abstract As ontologies are used widely, interest for semantic similarity search is also increasing. In this paper, we suggest a query evaluation scheme for k-nearest neighbor query, which retrieves k most similar objects to the query object. We use the best match method to calculate the semantic similarity between objects and use the signature tree to index annotation information of objects in database. The signature tree is usually used for the set similarity search. When we use the signature tree in similarity search, we are required to predict the upper-bound of similarity for a node; the highest similarity value which can be found when we traverse into the node. So we suggest a prediction function for the best match similarity function and prove the correctness of the prediction. And we modify the original signature tree structure for same signatures not to be stored redundantly. This improved structure of signature tree not only reduces the size of signature tree but also increases the efficiency of query evaluation. We use the Gene Ontology(GO) for our experiments, which provides large ontologies and large amount of annotation data. Using GO, we show that proposed method improves query efficiency and present several experimental results varying the page size and using several node-splitting methods.

Key words : Semantic similarity search, Signature tree

· 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 육성지원 사업(*IITA-2007-C1090-0701-0031*)의 연구결과로 수행되었음

† 학생회원 : 서울대학교 컴퓨터공학부
kskim@idb.snu.ac.kr
dhim@idb.snu.ac.kr
†† 비 회 원 : 서울대학교 컴퓨터공학부
chkim@idb.snu.ac.kr
††† 종신회원 : 서울대학교 컴퓨터공학부 교수
hjk@snu.ac.kr

논문접수 : 2006년 7월 3일
심사완료 : 2007년 9월 28일

정보과학회논문지: 데이타베이스 제34권 제6호(2007.12)

1. 서 론

최근 생물 정보학, 정보 검색, 상품 분류 등 여러 분야에서 의미적 유사성(semantic similarity)[1]에 대한 연구가 많이 이루어지고 있다. 의미적 유사성이란 객체의 의미적 정보를 사용해 객체간의 유사성을 측정하는 것이다. 의미적 유사성을 사용한 질의의 예는 다음과 같다.

- 특정 웹 페이지와 유사한 내용을 갖고 있는 웹 페이지를 검색하라.
- 이 상품과 유사한 기능을 갖는 상품을 검색하라.
- 이 유전자 산물과 유사한 생물학적 기능을 하는 유전자 산물을 검색하라.

이 예에서 알 수 있듯이 의미적 유사성 질의는 다양한 분야에서 활용이 가능하다. 본 논문에서는 이런 의미적 유사성 질의 처리를 위한 기법을 제안한다. 의미적 정보는 주로 온톨로지의 용어를 통해 제공되며 온톨로지는 용어간의 관계 정보를 제공한다. 기존에는 의미적 유사성 측정 방법에 대한 연구가 많이 이루어졌지만[1-6], 최근 온톨로지가 대용량화되고 주석 정보의 양이 급격히 늘어나면서 의미적 유사성 질의에 대한 효율적인 처리 기법의 필요성이 증가하고 있다.

의미적 유사성 검색은 객체를 주석 용어의 집합으로 본다면 집합 유사성 검색과 유사하다. 집합 유사성 검색에서는 유사도 함수로서 주로 자카르드 계수(Jaccard's coefficient), 다이스 계수(Dice's coefficient), 헤밍 거리(Hamming distance)등을 사용한다[7]. 그러나 이들 유사도 함수에서는 원소간의 유사성을 고려하지 않는 반면 의미적 유사성에서는 원소에 해당하는 용어들간의 유사성을 고려해야 한다.

본 논문에서는 시그니처 트리(signature tree)[8]를 사용한 의미적 유사성 검색 기법을 제안한다. 시그니처 트리는 집합 유사성 검색에서도 많이 사용된 색인 기법으로 분기 한정(branch and bound) 방식의 검색 알고리즘을 사용한다. 분기 한정 방식의 알고리즘을 사용하기 위해서는 자식 노드로 탐색을 하였을 때 찾을 수 있는 최대의 유사도를 예측하여야 한다. 이에 우리는 의미적 유사도 함수의 예측 최대값을 구하는 기법을 제안하였다. 또한 기존의 시그니처 트리에서는 동일한 시그니처가 중복 저장될 수 있는데, 이는 시그니처 트리의 사이즈를 증가시킬 뿐만 아니라 검색의 효율성을 떨어뜨린다. 이에 우리는 버켓(bucket)을 사용한 시그니처를 사용하여 질의의 성능을 향상시켰다.

우리가 제안한 기법의 성능을 평가하기 위해 대용량 온톨로지인 Gene Ontology[9]와 유전자 산물(gene product)의 주석 정보를 사용하여 실험을 하였다. 실험에서는 제안한 기법의 성능 측정뿐만 아니라 시그니처 트리의 여러 변수를 조정한 결과도 제시하였다. 우선 질의 크기에 따른 성능 변화와 노드 크기에 따른 성능 변화 결과를 제시한다. 또한 시그니처 트리 구성시 사용하는 노드 분할 알고리즘은 질의 처리 성능에 많은 영향을 주기 때문에[10] 여러 노드 분할 알고리즘을 사용하여 노드 분할 알고리즘에 따른 질의 처리 성능 변화도 제시하였다.

본 논문의 범위는 의미적 유사성 질의 처리에 한정하며 의미적 유사성 결과의 효과성(effectiveness)에 대한 검증은 다루지 않는다. 대신 가장 일반적으로 사용되는 유사도 함수를 사용하였을 때 적용할 수 있는 질의 기법을 제안하였다.

2. 관련연구

[7,11]에서는 집합 데이타에 대한 유사성 질의 기법을 제안하였다. 이들 기법에서는 집합간의 헤밍 거리 함수와 같이 공통 원소의 개수, 공통이지 않은 원소의 개수를 이용한 집합 유사성 함수를 사용한다. 이와 같이 기존의 집합 유사성 문제에서는 원소의 개수만을 고려하고 각 원소 사이의 유사성에 대한 고려를 하지 않는다. 그러나 의미적 유사성 함수에서는 온톨로지 용어간의 유사도를 고려해야 하기 때문에 이들 기법을 그대로 사용할 수 없다. 아직까지 원소간의 유사성을 고려한 유사도 함수에 대한 검색 기법은 제안되지 않았다.

집합간 유사성 검색 기법은 크게 시그니처 테이블을 이용한 방법과 시그니처 트리를 이용한 방법으로 나눌 수 있다. 시그니처 테이블을 이용한 기법에는 의미적 유사도 함수를 적용하는 것이 간단하지 않기 때문에 본 논문에서는 시그니처 트리를 사용하였다.

시그니처 트리의 구조는 다음과 같다. 시그니처 트리는 B-트리와 R-트리와 같은 디스크 기반의 균형 트리이다. 트리의 각 노드는 한 개 디스크 페이지를 나타내며 노드들은 <시그니처, 포인터> 형태의 엔트리를 갖는다. 디렉토리 노드의 포인터는 자식 노드를 가리키며 디렉토리 노드의 시그니처는 포인터가 가리키는 자식 노드에 속한 모든 시그니처의 OR 연산값이다. 말단 노드의 포인터는 시그니처가 나타내는 객체의 아이디를 갖는다.

한 노드가 가질 수 있는 엔트리의 최대 개수를 C라고 하면, 루트 노드를 제외한 모든 노드는 $\left\lceil \frac{C}{2} \right\rceil$ 와 C 사이의 엔트리를 갖고 있어야 한다. 그림 1은 시그니처 트리의 한 예를 보여준다.

3. 데이타 모델과 의미적 유사성 질의

우선 우리가 다루고자 하는 의미적 유사성 질의를 위한 데이타 모델과 의미적 유사성 측정함수에 대해 알아보자. $Ont = <T,E>$는 용어의 집합 T, 용어 간의 관계 집합 E를 갖는 온톨로지를 나타낸다. 두 용어 t, $t' \in T$ 사이에 is-a 관계가 있을 때 $<t,t'> \in E$이다. 용어간 관계는 모두 is-a관계로 가정한다. $S = \{O_1, ..., O_n\}$는 n개의 객체로 이루어진 데이타베이스를 나타낸다. 객체는 주석 용어의 집합으로 표현할 수 있다. $O = \{t_1, ..., t_n\}$는 객체 O의 주석 용어가 $t_1, ..., t_n$임을 나타낸다. 앞으로 O는 객체 O를 나타내며 동시에 객체 O의 주석 용어 집합을 나타낸다.

두 온톨로지 용어 t, t'의 유사성은 $Sim_T(t,t')$으로 나타낸다. 온톨로지 용어간의 유사성 $Sim_T(t,t')$은 $f: T \times$

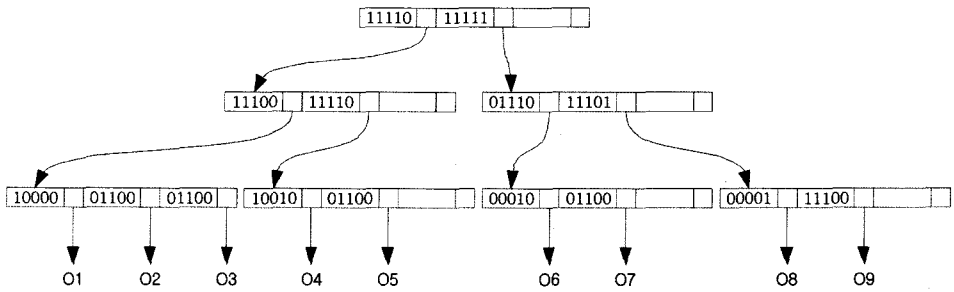

그림 1 시그니처 트리 예제

$T \rightarrow [0,1]$으로 기존에 제안된 여러 방법[1,3]을 사용하여 계산할 수 있다. $Sim(O_1,O_2)$은 두 객체 $O_1 = \{t_1, ..., t_n\}$, $O_2 = \{t_1, ..., t_m\}$에 대한 의미적 유사성을 나타낸다. $Sim(O_1,O_2)$은 $f: S \times S \rightarrow [0,1]$이며 객체간 의미적 유사성은 주석 용어간 유사성을 이용하여 계산한다. 표 1은 $Sim_T(t,t')$을 조합한 여러 $Sim(O_1,O_2)$의 계산 방법을 나타낸다. 여기에서 $m = |O_1|$, $n = |O_2|$이다.

표 1 의미적 유사성 측정 함수

방식	계산식
최대값	$\max_{t \in O_1, t' \in O_2} Sim_T(t,t')$
평균값	$\frac{1}{m \times n} \times \sum_{t \in O_1, t' \in O_2} Sim_T(t,t')$
최적 대응값	$\frac{1}{m+n} \times \left(\sum_{t \in O_1} BM(t,O_2) + \sum_{t \in O_2} BM(t,O_1) \right)$

최대값 방식에서는 $Sim_T(t,t')$ 중 가장 큰 값을 두 객체의 유사도로 계산한다. 이 방법은 가장 큰 유사도를 갖는 용어간의 유사성만을 고려하기 때문에 다른 용어의 정보가 무시되며 유사도가 과다 측정되는 문제점이 있다. 또한 가장 큰 $Sim_T(t,t')$만을 취하기 때문에 $Sim_T(t,t')$의 최대값이 같은 객체에 대해서는 항상 같은 유사도가 된다. 평균값 방식은 모든 용어쌍의 유사도를 반영한다. 최대값 방식에 비해 여러 값을 반영하는 장점이 있으나 이로 인해 유사도가 과소 측정되는 문제점이 있다. 또한 $|O| \geq 2$인 O에 대해서 $Sim(O,O) \leq 1$인 환원성(reflexivity)이 없다는 문제점이 있다.

최적 대응값는 이 두 방식의 문제점들을 완화해준다. 이 방식은 $Sim(O_1,O_2)$을 각 용어의 최적 대응 유사도를 평균하여 계산한다. 한 용어에 대한 최적 대응 유사도(BM)는 다음과 같이 정의한다.

정의 1. 최적 대응 유사도

용어 t의 객체 $O = \{t_1, ..., t_n\}$에 대한 최적 대응 유사도는 다음과 같다.

$$BM(t,O) = \max_{k \in O} Sim_T(t,k)$$

최적 대응값 방식은 환원성을 가지며, 용어간의 차이도 반영할 수 있어 과다 측정을 하지 않는다. 또한 최적 대응만을 고려하기 때문에 과소 측정 문제도 완화할 수 있다. 따라서 본 논문에서는 최적 대응값 방식을 사용하여 객체간의 유사도를 정의하겠다.

정의 2. 객체간 의미적 유사도

두 객체 $O_1 = \{t_1, ..., t_n\}$, $O_2 = \{t_1, ..., t_m\}$의 유사도 $Sim(O_1,O_2)$은 다음과 같이 정의한다.

$$Sim(O_1,O_2) = \frac{1}{m+n} \times \left(\sum_{t \in O_1} BM(t,O_2) + \sum_{t \in O_2} BM(t,O_1) \right)$$

본 논문에서 다루고자 하는 의미적 유사성 질의는 질의 객체 O와 결과 사이즈 k가 주어질 때, 데이타베이스 $S = \{O_1, ..., O_n\}$에서 객체 O와의 유사도 값이 가장 큰 k개의 객체를 구하는 것이다. 제안한 기법을 약간만 수정하면 이와 같은 k 최근접 질의 외에도 범위질의에 대해서도 쉽게 다룰 수 있다.

4. 시그니처 트리

이번 절에서는 의미적 유사성 질의 처리를 위한 시그니처 트리의 구성 방법에 대해 설명한다. 우선 객체의 주석 정보의 시그니처는 다음과 같이 정의한다.

정의 3. 시그니처

객체 O의 시그니처 $sig(O)$는 $|T|$개의 비트로 구성된 비트맵이다. $1 \leq i \leq |T|$에 대해, $t_i \in O$이면 $sig(O)$의 i 번째 비트를 1로 한다. $t_i \notin O$이면 i 번째 비트를 0으로 한다.

시그니처에 대해 다음과 같은 기호를 정의한다.

정의 4. 시그니처 무게 및 포함관계

시그니처 Sig에 대해 $|Sig|$는 1인 비트의 개수를 나타낸다. 이를 시그니처 Sig의 무게라 한다. 또한, 같은 크기의 두 시그니처 Sig_1, Sig_2에 대해, $Sig_1 \subset Sig_2$는 Sig_1에서 1인 비트는 모두 Sig_2에서도 1임을 나타낸다.

질의 처리를 위해 데이타베이스 $S = \{O_1, ..., O_n\}$의 모든 객체에 대해 정의 3에 따라 시그니처를 만들어 시그

니처 트리를 구성한다. 시그니처 트리는 기본적으로 2절에서 설명한 구조를 갖도록 구성한다. 여기서는 시그니처 트리의 구조와 관련해서 질의 처리를 효율적으로 하기 위한 기법에 대해 논하도록 한다.

우선 기존의 시그니처 트리에서는 중복된 시그니처에 대한 별도의 처리를 하지 않았기 때문에 이를 개선한 구조를 사용하였다. 같은 시그니처가 트리에 여러 개 존재하면 다음과 같은 문제가 발생한다.

- 시그니처 트리의 사이즈가 증가한다. 중복된 시그니처를 모두 저장하게 되면 $|S|$개의 말단 노드가 생기게 된다. 그러나 중복 시그니처를 한번만 저장하면 $|S|$보다 적은 말단노드가 생기게 되어 전체 시그니처 트리의 사이즈가 감소할 수 있다. 시그니처 트리 사이즈의 감소는 질의 처리시 I/O 비용을 줄일 수 있다.
- 동일한 시그니처가 흩어져 존재할 수 있다. 시그니처 트리에서는 유사한 시그니처끼리 모이게 구성해야 상위 노드에서의 시그니처 무게를 감소시킬 수 있다. 그러나 중복된 시그니처를 허용할 경우에는 동일한 시그니처가 흩어져 존재할 수 있고 트리의 전체적인 시그니처 무게를 증가시킬 수 있다.

이와 같이 같은 시그니처의 중복 저장은 질의 처리의 효율성을 저하시킨다. 따라서 본 논문에서는 버켓을 사용하여 중복된 시그니처가 시그니처 트리에 나타나지 않도록 하였다. 그림 2는 버켓을 사용한 시그니처 트리의 예제이다. 말단 노드의 포인터가 직접 객체의 아이디를 가리키지 않고, 버켓을 가리키게 한다. 각 버켓은 해당 시그니처를 가진 객체의 아이디를 갖고 있다.

이와 같이 버켓을 만들어 구성할 때에는 삽입 방법도 변형해야 한다. 중복된 시그니처가 아닌지 알기 위해 삽입 전에 삽입하려는 시그니처를 트리안에서 검색 한다. 이는 시그니처의 검색 알고리즘으로 처리할 수 있다. 만일 시그니처가 검색되었다면 해당 시그니처가 가리키는 버켓에 삽입하려는 객체의 아이디를 추가한다. 검색되지 않았을 때에는 삽입 알고리즘을 실행한다. 이와 같이 삽입시에 검색에 필요한 추가 비용이 추가적으로 생기지만 질의 처리시에 얻는 이득이 이런 추가 부담을 상쇄할 수 있다.

버켓을 사용할 때에는 질의 처리시에 다음과 같은 이득이 있다. 우선 위에서 말한 현상을 경감시켜 시그니처 트리를 탐색하는 비용을 줄일 수 있다. 또한 유사성 검색시 유사도 계산 회수를 감소시킨다. 그림 1과 같이 중복된 시그니처가 흩어진 경우에는 이들 객체에 대해 각각 유사도 계산을 하게 된다. 하지만 그림 2의 예제에서 O2, O3, O5, O7은 같은 시그니처를 갖고 있기 때문에 주석 정보가 같음을 알 수 있다. 따라서 유사도 계산을 한번만 할 수 있다.

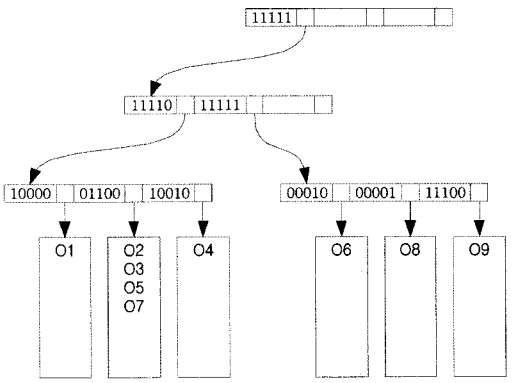

그림 2 버켓을 사용한 시그니처 트리 예제

다음으로 질의 성능과 관련된 시그니처 트리의 구조 이슈로는 노드 선택 알고리즘과 노드 분할 알고리즘이 있다. 질의 처리를 효율적으로 하기 위해서는 유사한 시그니처가 같은 말단 노드에서 모일 수 있도록 시그니처 트리를 구성해야 한다[7]. 시그니처 삽입 시에 이와 같은 목적을 달성하기 위해서는 효과적인 노드 선택 알고리즘과 노드 분할 알고리즘이 필요하다. 새로운 시그니처를 삽입할 때에는 루트 노드에서부터 트리를 탐색해 알맞은 말단 노드를 찾게 된다. 이때 사용하는 알고리즘이 노드 선택 알고리즘이다. 만일 시그니처를 삽입할 말단 노드에 빈자리가 없는 경우에는 노드를 두개로 분할하여야 하여 이때 노드 분할 알고리즘을 사용한다. 이 두 알고리즘은 질의 처리의 효율성에 큰 영향을 준다. 시그니처 삽입 알고리즘은 [7]에서 사용한 방식을 사용한다. 단 앞에서 언급한대로 삽입 알고리즘 전에 검색 알고리즘을 수행한다. 검색 결과 삽입하고자 하는 시그니처가 이미 트리에 있다면 해당 버켓에 객체의 아이디를 추가함으로써 삽입을 끝낸다. 해당 시그니처가 트리에 없다면 삽입 알고리즘을 계속 진행하여 추가하도록 한다.

노드 분할 알고리즘은 [10]에서 좋은 성능을 보인 큐빅(cubic) 분할 알고리즘과 계층 클러스터링 알고리즘을 사용하여 분할 알고리즘에 따른 질의 처리 성능을 분석하였다. 자세한 내용은 실험 결과에서 다루도록 한다.

5. 의미적 유사성 질의 처리

이번 절에서는 앞 절에서 설명한 시그니처 트리를 이용한 유사성 질의 기법에 대해 설명한다. 질의는 주로 k-최근접 질의에 대해 설명한다. 그러나 범위 질의에 대해서도 약간의 변형으로 처리할 수 있다.

질의 처리시에는 우선 질의 객체 Q의 시그니처를 만든다. 앞으로는 객체 Q에 대한 시그니처 역시 Q로 표기한다. 질의 처리는 분기 한정 방식의 알고리즘을 사용한다. 즉 루트 노드에서부터 트리를 탐색하며 각 노드

마다 유사도의 예측 최대값을 구한다. 예측 최대값이란 그 엔트리의 자식 노드로 분기하였을 때 찾을 수 있는 유사도의 최대값을 말한다. 만일 예측 최대값이 현재 구한 k개 결과 중 가장 낮은 유사도 보다 낮은 값이면 그 노드로는 탐색하지 않아도 됨을 알 수 있다.

시그니처 트리에서 예측 최대값은 다음의 조건을 만족해야 한다. $Sim(Q, Sig)$는 두 시그니처가 나타내는 객체의 유사도를 나타낸다.

조건 1. 예측 최대값의 조건

질의 시그니처 Q와 디렉토리 노드의 엔트리 시그니처 ES의 예측 최대값 $ExpSS(Q, ES)$는 모든 $Sig \subset ES$와 Q에 대해 다음 조건을 만족해야 한다.

$$ExpSS(Q,ES) \geq \max_{Sig \subset ES} Sim(Q,Sig)$$

이 예측값이 현재까지 찾은 k개 결과의 유사도 값 중 가장 낮은 유사도보다 큰 경우에만 해당 엔트리의 자식 노드로 분기한다. 만일 이미 찾은 k개 결과의 유사도보다 예측값이 낮다면 그 노드로 분기하여도 결과에 해당하는 객체가 없다는 것을 알 수 있기 때문이다.

트리 탐색은 최상-우선 검색(Best-first search) 방식[12]을 사용한다. 최상-우선 검색은 우선순위 큐(priority queue)를 사용한다. 이 큐는 <엔트리 e, $ExpSS(Q,e.Sig)$> 튜플을 갖고 있다. 검색을 시작하면 루트 노드의 모든 엔트리를 큐에 넣는다. 큐의 첫째 원소는 항상 $ExpSS(Q,e.Sig)$ 값이 가장 큰 엔트리를 갖고 있다. 검색의 각 스텝 마다 첫째 원소를 얻고, 엔트리가 가리키는 노드를 읽어 들여 노드 안의 엔트리 들을 큐에 넣는다. 만일 엔트리가 가리키는 노드가 말단 노드이면 질의 객체와의 유사도를 계산한다. 그림 3은 전체 알고리즘을 나타낸다.

이와 같이 질의 처리를 하기 위해서는 조건 1을 만족하는 예측 최대값 함수가 필요하다. 만일 예측 최대값이 너무 느슨한 상위 한계값일 경우에는 가지치기가 거의 일어나지 못해 질의 성능을 떨어뜨리게 된다. 본 논문에서는 유사성 질의 처리를 위해 정의 3의 함수를 사용한다. 앞으로 $ExpSS(Q,ES)$는 정의 3의 함수를 지칭하도록 한다. 정의 3의 함수의 성능은 실험을 통해 살펴보기로 한다.

정의 3. 최적 대응 유사도 함수에 대한 예측 최대값

최적 대응 유사도 함수에 대한 질의 시그니처 Q와 디렉토리 노드의 엔트리 시그니처 ES의 예측 최대값 $ExpSS(Q,ES)$은 다음과 같다.

$$ExpSS(Q,ES) = \frac{\sum_{t \in Q} BM(t,ES) + |ES| \cdot \max_{t \in Q} BM(t,ES)}{|Q| + |ES|}$$

알고리즘 1 질의처리

```
procedure SimilaritySearch(Signature query, int k) {
  PriorityQueue entries;
  Heap results(k);
  for all entries e of root node {
    expSS = ExpSS(Q, e.sig);
    entries.add(e, expSS);
  }
  while(entries.isEmpty())
  {
    Entry e = entries.pop();
    expSS = ExpSS(Q, e.sig);
    if(expSS < results.lowestSimilarity) break;
    Node n = e.pointer;
    if n is a directory node then {
        for all entries e of node n {
            expSS = ExpSS(Q, e.sig);
            entries.add(e,expSS);
        }
    }
    else
    {
        for all entries e of n
            results.offer(e, SS(Q, e.sig));
    }
  }
  return results;
}
```

그림 3 질의 처리 알고리즘

이제 정의 3의 함수가 실제로 조건 1을 만족하는지에 대해 알아보자. 우선 유사도 함수를 구성하는 BM값은 다음과 같은 성질을 갖는다.

정리 1. BM 값의 성질

두 객체 O_1, O_2의 BM값에 대해 다음이 성립한다.

$$for\ all\ \ t \in O_1,\ \ BM(t,O_2) \leq \max_{k \in O_2} BM(k,O_1)$$

증명)

모순을 통해 증명한다. 다음을 만족하는 $t \in O_1$가 존재한다고 가정하자.

$$BM(t,O_2) > \max_{k \in O_2} BM(k,O_1)$$

BM의 정의에 의해 다음을 만족하는 $t' \in O_2$이 존재한다.

$$BM(t,O_2) = \max_{k \in O_2} Sim(t,k) = Sim(t,t') > \max_{k \in O_2} BM(k,O_1)$$

위의 식을 만족하는 $t' \in O_2$에 대해서는 다음이 성립하게 된다.

$$BM(t',O_1)=\max_{t\in O_1} Sim(t,t') \le \max_{t'\in O_2} BM(t',O_1) < Sim(t,t')$$

그러나 이는 모순이 되며 따라서 위의 정리가 성립함을 알 수 있다. □

위의 정리를 이용하면 다음이 성립함을 쉽게 알 수 있다.

$$\frac{\sum_{t\in Q} BM(t,ES)+|ES|\cdot\max_{t\in Q} BM(t,ES)}{|Q|+|ES|} \ge \frac{\sum_{t\in Q} BM(t,ES)+\sum_{t\in ES} BM(t,Q)}{|Q|+|ES|}$$

즉, $ExpSS(Q,ES)\ge Sim(Q,ES)$임을 알 수 있다. 정리 2에서는 시그니처의 포함 관계와 $ExpSS$의 관계를 알려준다.

정리 2. $ExpSS(Q,ES)$의 성질

모든 $Sig\subset ES$에 대해 다음이 성립한다.

$$ExpSS(Q,ES)\ge ExpSS(Q,Sig)$$

증명)

$$ExpSS(Q,ES)=\frac{\sum_{t\in Q} BM(t,ES)+|ES|\cdot\max_{t\in Q} BM(t,ES)}{|Q|+|ES|}=\frac{\alpha+\beta}{m+n},$$

$$ExpSS(Q,Sig)=\frac{\sum_{t\in Q} BM(t,Sig)+|Sig|\cdot\max_{t\in Q} BM(t,Sig)}{|Q|+|Sig|}=\frac{\alpha'+\beta'}{m+n'}$$

이라 하자.

부등식 $ExpSS(Q,ES)-ExpSS(Q,Sig)\ge 0$에 위를 대입하면, $(\alpha-\alpha')+(\beta-\beta')\ge\frac{\alpha+\beta}{m+n}$이 된다. 우선, $Sig\subset ES$이기 때문에 $|Sig|=|ES|$이면 $Sig=ES$이고 따라서 $ExpSS(Q,ES)-ExpSS(Q,Sig)\ge 0$가 성립한다.

$|Sig|<|ES|$인 경우 $\max_{t\in Q} BM(t,Sig)\le\max_{t\in Q} BM(t,ES)$이며, $\beta-\beta'\ge\max_{t\in Q} BM(t,ES)\ge\frac{\alpha+\beta}{m+n}$이다. 또한 $\alpha-\alpha'\ge 0$임도 쉽게 알 수 있다. 따라서 $(\alpha-\alpha')+(\beta-\beta')\ge\frac{\alpha+\beta}{m+n}$이 성립함을 알 수 있다. □

정리 2를 이용하면 모든 $Sig\subset ES$ 다음이 성립함을 알 수 있다.

$$ExpSS(Q,ES)\ge ExpSS(Q,Sig)\ge SS(Q,Sig)$$

따라서 $ExpSS(Q,ES)$를 예측 최대값으로 사용할 수 있음을 알 수 있다.

6. 실험 결과

시스템 구현은 Java SDK 1.5를 사용하였다. 실험에 사용한 데이타는 2006년 5월 버전 GO 데이타를 사용하였다. 온톨로지를 구성하는 용어는 7,928개이며, 객체에 해당하는 유전자 산물의 개수는 모두 1,670,726개이다. 실험 데이타에서 한 객체당 주석 용어의 개수는 평균 2.33개이며 최대 주석 용어의 개수는 22개이다. 실험 환경은 PentiumM 1.6GHz CPU, 1G RAM이며 WindowsXP 운영체제를 사용하였다.

질의 성능에 영향을 주는 요인으로 시그니처 트리의 구조, 페이지 크기, 노드 분할 방법에대한 실험을 하였다. 시그니처 트리의 구조는 버켓을 사용한 것과 사용하지 않은 것을 말하며, 페이지 크기는 한 노드가 차지하는 바이트 용량을 말한다. 노드 분할 방법은 큐빅 분할과 계층적 분할을 사용하였다. 각 분할 방법의 자세한 내용은 뒤에서 설명하도록 한다.

우선 제안한 시그니처 트리가 유사성 질의 성능을 향상시키는지 실험하였다. 이를 위해 기존의 시그니처 트리와 버켓을 이용한 시그니처 트리를 구성하였다. 표 2는 시그니처 트리를 구성하는데 걸린 시간과 시그니처 트리의 사이즈를 나타낸다. STREE_ORG은 기존의 시그니처 트리 구성 방법을 사용한 것이고 STREE_BUC은 버켓을 사용한 시그니처 트리를 나타낸다. STREE_BUC의 경우 버켓의 용량도 포함하였다. 크기가 약 40배 차이가 나는 이유는 GO 데이타의 특성상 주석 정보가 동일한 객체가 많기 때문이다. 크기의 차이에 비해 구성 시간의 차이는 그다지 크지 않은데, 이는 앞서 설명한대로 버켓을 사용할 때에 삽입시 추가적으로 생기는 부담 때문이다.

표 2 시그니처 트리 크기와 구성시간

	STREE_ORG	STREE_BUC
크기(MB)	5689.465	140.150
구성시간(s)	31491.466	28456.147

질의 성능은 시그니처 트리를 탐색하며 읽어 들이는 페이지의 개수를 사용해 측정하였다. 성능 측정은 질의 시그니처의 무게를 변화하며 측정하였으며 각 시그니처 무게 마다 20개의 임의의 질의를 생성하여 평균을 계산하였다. 임의의 질의를 구성하기 위해 임의의 용어를 선정하여 질의를 구성하였다. 이 실험에서는 페이지 크기는 4K, 노드 분할 방법은 큐빅 분할을 사용하였다.

그림 4의 결과에서 알 수 있듯이 버켓을 사용한 시그니처 트리를 사용할 때에 질의 성능이 향상됨을 알 수 있다. 또한 이 결과에서 주목할 점은 두 경우 모두 주석 용어 개수가 늘어남에 따라 I/O횟수가 증가하였다는 것이다. 이는 주석 용어의 개수가 늘어날수록 예측값 함수 값의 크기가 커지는 경향이 있고 결과적으로 가지치기 가능성이 낮아지기 때문이다.

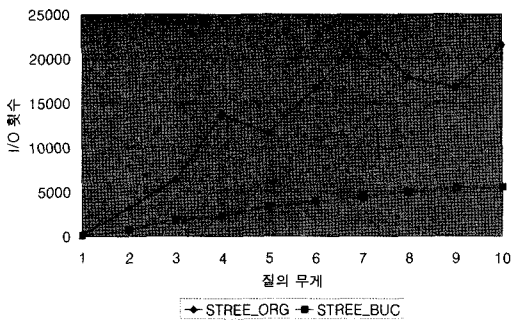

그림 4 시그니처 트리 구조에 따른 질의 성능

다음으로는 한 노드의 크기에 따른 질의 성능을 테스트 하였다. 이 실험에서는 STREE_BUC를 사용하였으며 큐빅 분할을 사용하였다.

노드 크기에 따라 한 노드에 저장할 수 있는 엔트리의 개수가 정해진다. 우리가 사용한 시그니처를 4K 노드에는 4개, 8K 노드에는 8개, 16K 노드에는 16개의 노드를 저장할 수 있었다. 노드 개수가 늘어날수록 한 노드에 저장할 수 있는 엔트리의 개수는 늘어나지만 반면에 부모 노드 시그니처의 무게가 그만큼 커진다. 그림 5의 결과를 보면 노드 크기를 시킬수록 읽어들이는 노드의 개수는 줄어들지만 실제 로딩되는 양은 증가하고 있음을 알 수 있다.

다음으로 노드 분리 방법에 따른 질의 성능을 알아보았다. 노드 분리 방법은 시그니처를 이용한 질의 처리 성능에 상당한 영향을 준다. 올바른 노드 분리 방법을 사용함으로써 질의 처리시에 시그니처 트리를 탐색하는 비용을 줄일 수 있다. 이에 노드 분리 방법에 따른 성능 변화를 측정하였다. 이 실험에서는 SIG_BUCKET을 사용하였다.

실험에 사용한 노드 분리 방법은 큐빅 분할과 계층적 분할이다. 이 두 분할 방법을 선택한 이유는 [10]에서 이 두 분할 방법의 성능이 가장 좋게 나타났기 때문이다. 큐빅 분할에서는 분할하려는 엔트리 집합에서 두 개의

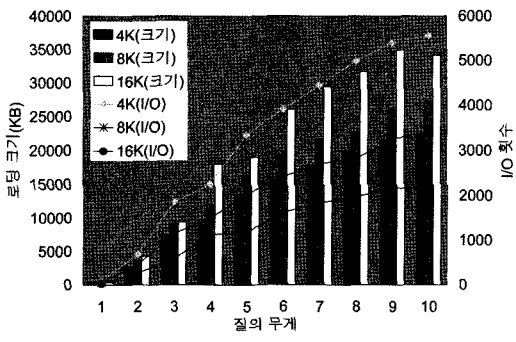

그림 5 노드 크기에 따른 질의 성능

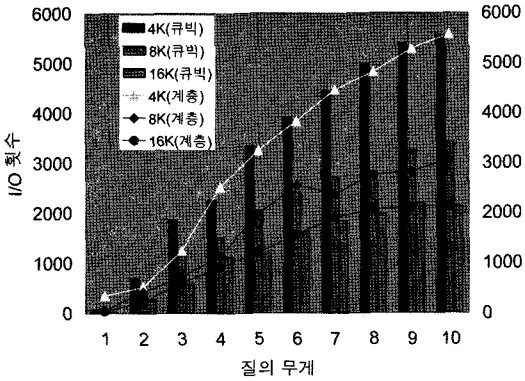

그림 6 노드 분할에 따른 질의 성능

시드(seed) 엔트리를 선정하여 각각을 두 노드에 배정한다. 그 후 나머지 엔트리들을 이 두 선정된 엔트리 중 차이가 적은 엔트리가 속한 노드로 배정한다. 이때 시드 엔트리를 분할하려는 엔트리 집합의 모든 엔트리 쌍에 대해 시도하게된다. 계층적 분할은 계층적 클러스터링 방식을 사용하여 엔트리 집합을 두 노드로 분할한다.

그림 6은 노드 분할에 따른 질의 성능을 나타낸다. 결과 상으로는 노드 분할에 따른 성능 차이가 뚜렷하지는 않다. 또한 [10]에서는 큐빅 분할의 성능이 가장 우수하였지만 결과에서는 계층 분할에서 I/O횟수가 적은 경우가 많다. 그러나 노드 사이즈가 커질수록 큐빅 분할의 성능이 약간 좋아지고 있음을 알 수 있다. 이런 현상의 원인으로는 노드 분할 알고리즘이 질의 성능에 영향을 주기 위해서는 한 노드안에 포함한 엔트리의 개수가 많아야 하기 때문으로 분석된다. 4K 노드의 경우 4개의 엔트리가 포함되어 있기 때문에 이렇게 작은 수의 엔트리들을 분리하는 데에 있어서 노드 분할 알고리즘의 성능차이가 뚜렷이 나타나지 못하는 것이다. 16K 노드에서는 계층분할이 좀더 좋은 성능을 보이고 있음을 알 수 있다.

7. 결 론

본 논문에서는 시그니처 트리를 사용한 의미적 유사성 질의 처리의 기법을 제안하였다. 시그니처 트리는 B-트리와 같은 디스크 기반의 균형 트리로서 집합 유사성 문제에서 많이 사용되고 있다. 의미적 유사성 질의 처리에 시그니처 트리를 사용하기 위해서 우리는 최적 조합 유사도 함수에 대한 예측 최대값을 구하는 함수를 제안하였다. 이 예측 최대값의 성능은 시그니처 트리의 탐색 도중 가지치기를 결정하는 요인으로서 질의 성능에 영향을 준다. 제안된 예측 함수의 효율성은 실험을 통해 살펴보았다. 또한 중복된 시그니처 저장이 질의 성능에 미치는 영향을 알아보았으며 버켓을 사용한 구조를 제안하

였다. 이 개선된 구조는 실험에 사용한 GO와 같이 중복된 주석 정보가 많이 존재하는 경우 상당한 성능 향상을 보여주었다. 버켓을 사용한 구조에서는 삽입시에 중복된 시그니처를 검색하는 과정이 필요하기 때문에 시그니처 구성 시간이 증가하지만 이는 질의 처리 성능 향상 정도에 비해 미미한 수준이다. 또한 시그니처 트리를 구성할 때 변수가 되는 노드 크기와 노드 분할 알고리즘의 선택에 대해서 다양한 실험을 통해 이들 변수가 질의 성능에 미치는 영향에 대해서 살펴보았다.

온톨로지를 사용한 유사성 검색은 앞으로 다양한 분야에서 활용될 것으로 기대된다. 향후 연구로는 다양한 온톨로지를 사용하였을 때의 검색기법과 주석 타입등과 같은 여러 주석 정보를 사용한 유사성 함수를 처리할 수 있는 기법이 필요하다.

참 고 문 헌

[1] Resnik, P., *Semantic Similarity in a Taxonomy: An Information-based Measure and its Application to Problems of Ambiguity in Natural Language.* Journal of Artificial Intelligence Research, 1999. 11: p. 95-130.

[2] Varelas, G., E. Voutsakis, and P. Raftopoulou. *Semantic Similarity Methods in Wordnet and their Application to Infomation Retrieval on the Web.* in *WIDM.* 2005. Bremen, Germany.

[3] Lin, D. *An Information-theoretic Definition of Similarity.* in *15th International Conf. on Machine Learning.* 1998. San Francisco, CA.

[4] Jiang, J.J. and D.W. Conrath, *Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy,* in *International Conference Research on Computational Linguistics.* 1997: Taiwan.

[5] Rada, R., et al., *Development and Application of a Metric on Semantic Nets.* IEEE Transactions on Systems, Man and Cybernetics, 1989. 19(1): p. 17-30.

[6] Lee, J.H., M.H. Kim, and Y.J. Lee, *Information Retrieval based on Conceptual Distance in is-a Hierarchies.* Journal of Documentation, 1989. 49(2): p. 188-207.

[7] Mamoulis, N., D.W. Cheung, and W. Lian. *Similarity Search in Sets and Categorical Data Using the Signature Tree. in 19th International Conf. on Data Engineering.* 2003.

[8] Uwe, D., *S-tree: a dynamic balanced signature index for office retrieval, in Proceedings of the 9th annual international ACM SIGIR conference on Research and development in information retrieval.* 1986, Pisa, Italy.

[9] Ashburner, M., et al., *Gene Ontology: tool for the unification of biology.* Nat Genet, 2000. 25(1): p. 25-29.

[10] Tousidou, E., A. Nanopoulos, and Y. Manolopoulos, *Improved Methods for Signature- Tree Construction.* The Computer Journal, 2000. 43(4): p. 301-314.

[11] Charu, C.A., L.W. Joel, and S.Y. Philip, *A new method for similarity indexing of market basket data, in Proceedings of the 1999 ACM SIGMOD international conference on Management of data.* 1999, ACM Press: Philadelphia, Pennsylvania, United States.

[12] G, R.H. sli, and S. Hanan, *Distance browsing in spatial databases.* ACM Trans. Database System., 1999. 24(2): p. 265-318.

김 기 성

2003년 서울대학교 응용화학부(학사). 2003년~현재 서울대학교 전기, 컴퓨터공학부 대학원 박사과정. 관심분야는 데이타베이스, 시맨틱웹 , 온톨로지, 생물정보학

임 동 혁

정보과학회논문지 : 소프트웨어 및 응용 제 34 권 제 5 호 참조

김 칠 한

2003년 8월 경북대학교 통계학과 학사 2003년 8월 경북대학교 컴퓨터공학과 학사. 2007년 8월 서울대학교 전기컴퓨터공학부 석사. 2007년 8월~현재 대우증권 정보통신센터 차기시스템부 재직. 관심분야는 Spam detection, Database Applications, Semantic web, Web2.0 Technologies

김 형 주

정보과학회논문지 : 소프트웨어 및 응용 제 34 권 제 5 호 참조