개념을 이용한 질의 확장에 관한 연구

A Study on Query Expansion Using Concept

김귀정*, 한정수**

건양대학교 컴퓨터공학과*, 천안대학교 정보통신학부**

Gui-Jung Kim(gjkim@konyang.ac.kr)*, Jung-Soo Han(jshan@cheonan.ac.kr)**

요약

검색 집합에 대한 정확한 지식 없이는 대부분의 사용자가 효율적인 질의 형성에 많은 어려움을 겪고 있다. 사실 대부분의 사용자는 그들이 필요한 결과를 얻기 위한 질의를 만드는데 많은 시간을 소비하고 있으며, 이러한 어려움을 극복하기 위한 방법 중의 하나가 초기 질의로부터 더 좋은 질의를 형성해 가는 질의 확장이다. 본 연구에서는 초기 질의의 결과로 검색된 클래스가 가지고 있는 개념을 이용하여 질의를 확장하는 개념 기반 질의 확장 방법을 제안한다. 개념은 시소러스에 의해 확장되며, 질의 확장 과정의 효율성을 평가하기 위하여 시뮬레이션을 통한 최적의 검색 효율을 나타내는 임계치를 설정하고 재현율과 정확도를 비교하였다.

■ 중심어 : | 개념 | 질의 확장 | 시소러스 |

Abstract

Without detailed exact knowledge of a retrieval collection, most users find it difficult to formulate effective queries. In fact, most users may spend large amount of time formulating queries in order to obtain their desired result. A method to overcome this difficulty is to use query expansion that reformulates better query from initial query. In this paper we propose concept based query evaluation method using concept of class that retrieved from initial query. This concept is expanded through thesaurus. For efficiency evaluation of query expansion, we defined most critical value through a simulation and compared precision and recall each other.

■ keyword : | Concept | Query Expansion | Thesaurus |

I. 서 론

소프트웨어 시스템에서 적절한 컴포넌트를 검색하기 위해 효과적인 질의를 형성하는 것은 쉬운 작업이 아니다. 더구나 적은 수(3개 또는 그 이하)의 키워드를 이용하여 필요한 질의를 형성하기 위해서는 질의 표현의 모호함을 극복할 수 있을 만큼 사용자가 키워드를 식별하는 능력이 뛰어나야 하며 이는 객관적인 검색 시스템을 구현하는 가장 큰 문제점 중 하나이다. 질의하고자 하는 질의문을 완전한 문장으로 작성할 수 없는 상황이라면 적은 수의 키워드로 가장 바람직한 소프트웨어 컴포넌트를 검색할 수 있도록 질의를 변환 또는 확장하는 과

접수번호 : #041012-001
접수일자 : 2004년 10월 12일
심사완료일 : 2004년 12월 07일
교신저자 : 김귀정, e-mail : gjkim@konyang.ac.kr

정이 필요하다[1][3]. 확장된 질의는 의미적으로 초기 질의의 구체화된 형태이거나 긴밀하게 연관된 형태이어야 한다. 이를 위해 시소러스와 같은 지식베이스를 이용한 정보검색 방법들이 많이 제안되었으나, 기존의 시소러스 관련 연구의 문제점은 첫째 사용자 질의나 문제 질의로부터 개념적인 연상 과정의 명확성이 모호하여 개념적 용어의 정의가 어려운 점, 둘째 컴포넌트가 증가할수록 노이즈가 많이 발생하는 문제, 셋째 추출된 특성에 대해 도메인 전문가의 시간과 노력이 너무 많이 요구되는 문제 등이 있다. 이처럼 시소러스를 사용하는 일은 많은 검색 시간을 필요로 하며, 심지어는 검색 과정에서 원하는 정보를 찾지 못하는 경우도 발생한다[4][5][8]. 또한 기존의 시소러스 기반 검색은 거의 정보검색을 위한 시소러스이기 때문에 객체지향 컴포넌트의 검색에서는 기존 방법을 그대로 사용할 수 없다[3]. 이에 본 논문에서는 이 단점을 해결하기 위해 객체지향 컴포넌트의 특징에 부합하는 시소러스 기반 검색 시스템을 구축하기 위하여 개념을 이용한 질의 확장 방법을 제안하였다. 이 방법은 시소러스를 이용하여 검색에 대한 일정한 정확도를 보장하면서 재현율을 향상시킬 수 있게 한다. 클래스가 가지는 특성에 따라 개념을 설정하고 각 개념과 개념 사이의 관계를 시소러스로 구축하여 사용자 질의와 정확히 일치하는 클래스뿐 아니라 개념적으로 서로 연관된 유사한 의미를 가지는 클래스까지 검색할 수 있게 한다. 이를 위해 본 연구에서는 효과적인 질의 확장과 검색 노이즈의 감소를 위하여 시뮬레이션을 통한 최적의 검색 효율을 나타내는 임계치를 설정하였다.

II. 관련 연구

1. 시소러스에 의한 질의 확장

개념질의 확장 방법[3]은 적은 수의 키워드를 이용하여 필요한 질의를 형성하기 위해서 시소러스에 의한 질의 확장과 피드백을 동시에 사용한다. 각 용어들은 개념에 속해 있으며, 어떤 용어가 질의로 주어졌을 때 그 용어가 속해 있는 개념의 모든 용어로 질의가 확장되는 방식이다. 이때 시소러스에 의해 개념이 확장되고 한 번의 확장으로 검색이 가능하다. 그러나 이 방법은 도메인에 대한 지식이 없거나 정확한 키워드를 모를 경우에는 사용하기가 용이하지 않기 때문에 이용자가 깊은 이해가 없어도 시소러스의 지식을 보다 잘 활용할 수 있는 그룹화 방법이 필요하다. 또한 분류구조를 초기에 설정하여 사용하기 때문에 고정된 구조를 가지며 새로운 특성의 컴포넌트를 추가하거나 기존의 컴포넌트를 제거할 경우, 전체적으로 시스템 구축방법을 재구성해야 하므로 확장이 어려운 단점이 있다.

객체기반 시소러스[6]는 시소러스에 개념 표현 레벨과 인스턴스 표현 레벨로 구성된 객체지향 패러다임을 적용함으로써 객체들 사이에 존재하는 복잡한 관계성들의 표현에 자동 구축전략을 제공한다. 그러나 이 방법은 효과적인 질의 재형성 과정이 필요하며 실제적으로 응용될 수 있는 검색 시스템으로의 개발이 필요하다. 적응형 시소러스[7]는 스프레이딩 액티베이션 기반의 유사도 측정 방법을 기반으로 신경망을 이용하여 학습 가능한 시소러스를 제안하였다. 제안한 시소러스는 가중치를 효과적으로 조절할 수 있는 장점이 있지만, 표준화된 형태의 시소러스를 자동으로 추출하는 방법과 보다 적절한 활성화 함수를 개발하는 방법이 요구된다. 계층적 시소러스 시스템[8]은 코드에서 계층적 분류를 위한 범주를 설정하고, 컴포넌트가 행위적 특성에 따라 분류되는 방법을 제안하였다. 컴포넌트 특성은 용어의 쌍으로 구성되며, 소프트웨어 디스크립터(software descriptor : SD)로부터 추출된다[9][10]. 특성에 따라 계층적으로 분류된 용어의 쌍은 퍼지 시소러스를 이용하여 유의어 사전을 구축하게 된다. 그러나 컴포넌트의 검색이 아닌 멤버함수와 파라미터를 이용한 클래스 검색이기 때문에 클래스가 증가할수록 멤버함수가 기하급수적으로 증가하여 노이즈가 많아진다는 단점이 있다.

또한 위의 시소러스는 모두 정보 검색을 위한 시소러스이며 거의 대부분의 시소러스 방법이 정보 검색을 대상으로 하기 때문에 객체지향 컴포넌트의 재사용을 위한 검색에는 기존의 방법을 그대로 적용할 수 없다. 더욱이 컴포넌트 검색을 위한 시소러스의 구축은 전통적

인 정보 검색 시스템에서 사용한 통계적 방법을 그대로 사용할 수 없다. 소프트웨어 분석 시 거의 대부분의 컴포넌트들은 자세한 디스크립터(descriptor)를 가지고 있지 않으며, 설령 있다하더라도 용어 출현빈도에 사용한 통계적 방법을 컴포넌트 디스크립터에 그대로 적용할 수는 없다. 정보 검색 시스템에서의 용어와 컴포넌트에서의 객체와의 가장 커다란 차이점은 객체는 그 자신이 소스코드를 가질 수 있으며, 같은 의미의 객체라 할지라도 소스가 같지 않은 경우가 있기 때문에 객체의 연관성을 맺어 주는 것은 매우 중요하다.

2. 사용자 피드백에 의한 확장

가장 많이 사용하는 질의 수정방법 중 하나가 사용자 피드백 방법이다. 사용자 피드백은 사용자 집단의 요구에 적응적으로 반응하기 위하여 시스템을 장기간에 걸쳐 서서히 변화시킴으로써 가능하다. 이 방법은 단일 질의에 대한 최적화보다는 시스템의 전반적인 향상을 목적으로 한다. 사용자 피드백은 클래스 가중치를 변화시키거나 퍼지화 함수의 모양(기울기)을 변화시킴으로써 이루어진다[3][8]. 새로운 함수를 계산하는 방법은 다음과 같다. 이 식은 퍼지화 함수를 변화시키는 방법으로, 수정된 함수값과 이전의 값 간의 차에 사용자 등급정도를 곱해줌으로써 새로운 함수 값이 수정된 값과 점진적으로 가까워지는 형태를 취한다.

$$D_{new}(t) = (1-\beta)D_{old}(t) + \beta D_{corr}(t)$$

파라메타 β는 사용자 등급에 따라 시스템 반영 정도를 다르게 해주는 역할을 한다. 함수 변화는 컴포넌트의 지속적이고 장기적인 선택의 결과로써 이루어지기 때문에 $\beta \ll 1$이며, 사용자 피드백에 의한 퍼지화 함수 모양의 수정은 매우 서서히 이루어지게 된다. 우선순위에 따라 검색된 컴포넌트 중 사용자가 첫 번째 컴포넌트를 선택하지 않고 k번째 컴포넌트를 선택했을 때, 첫 번째부터 (k-1)번째 컴포넌트의 경우에 $D_{corr}(t)$는 다음과 같이 수정된다.

$$D_{corr}(t) = (1+\gamma)t - \gamma \quad , \quad t \le 0.5$$
$$D_{corr}(t) = 2(1+\gamma)(1-t) - \gamma \quad , \quad t > 0.5$$

그리고, 사용자가 선택한 k번째 컴포넌트의 $D_{corr}(t)$는 다음과 같이 수정된다.

$$D_{corr}(t) = 2(1-\gamma)t + \gamma \quad , \quad t \le 0.5$$
$$D_{corr}(t) = (1-\gamma)(1-t) + \gamma \quad , \quad t > 0.5$$

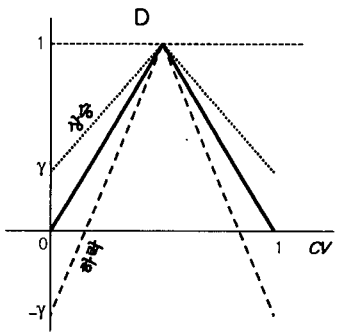

그림 1. 퍼지화 함수의 변화

그림 1은 사용자가 선택한 컴포넌트에 따라 변화하는 퍼지화 함수의 모양을 나타낸 것이다. 파라메타 γ는 애플리케이션 엔지니어에 의해 조절되며, 시스템의 변화 정도를 나타낸다. γ의 값이 크면 클수록, 퍼지화 함수의 모양이 더 급격히 변하게 되어 사용자 피드백의 결과가 시스템에 더 빨리 반영되게 된다.

III. 개념에 의한 클래스 분류

1. 클래스 추출

본 논문은 서로 관련 있는 특성들을 일정한 기준에 따라 그룹화하여 여러 개의 개념 그룹으로 구성하는 방식을 제안한다. 이를 위해 소프트웨어 디스크립터가 이용되는데 소프트웨어 디스크립터는 컴포넌트의 행위적 특성을 전체적인 관점에서 제공해주는 역할을 한다.

```
SD = Concept1 list of [feature]
     Concept2 list of [feature]
     —
     Conceptc list of [feature]
```

그림 2. 소프트웨어 디스크립터의 형태

소프트웨어 디스크립터는 클래스 행위를 나타내는 특성(feature)으로 구성되며, 모든 특성은 하나의 개념(concepts)에 포함된다. 그림 2는 소프트웨어 디스크립터의 형태를 보여준다. 특성은 각 컴포넌트를 나타내는 용어이며, 이 특성은 일정한 기준에 의해 개념으로 분류되어 그룹화된다. 소프트웨어 디스크립터를 구성하기 위해서는 소스코드로부터 클래스 추출이 이루어져야 한다. 클래스 추출은 클래스 구문 분석에 의해 이루어지고, 클래스 구문 분석은 프로그램 소스 파일을 입력받아 먼저 토큰(token) 단위로 클래스 이름을 모두 추출한 후, 처음 클래스명에서 다음 클래스명이 나타날 때까지 비교하면서 클래스 수만큼의 반복으로 클래스 정보를 추출한다. 추출 과정은 소스 코드를 읽어 각 클래스를 추출하고 클래스 사이의 관계 정보를 저장한다. 그림 3은 클래스 구문 분석 알고리즘으로써, "class"라는 단어와 클래스 구조의 끝을 나타내는 "};" 사이의 클래스명을 추출하며, 각 클래스의 추출 개수를 저장한다.

```
1: Set var class_count := 1
2: Repeat
3:   If exist "}"
4:     set var endClass :=  location
5:   Else
6:     exit
7:   Endif
8:   If exist "class"
9:     set var startClass := location
10     set var MC&mm have strings from "class" to "}"
11     increase class_count
12   Endif
13  Forever
14  set var totalc := class_count    /* 전체 클래스 개수 */
    /* class 단위로 분류되어 저장 */
```

그림 3. 클래스 구문 분석

이때, 클래스는 자신이 가지고 있는 클래스 정보와 함께 추출된다. 클래스 정보는 특성, 개념, 도메인 정보, 시소러스 정보로 구성된다. 이를 위해서 클래스들을 구문 분석하여 클래스내의 모든 정보를 상호 관련정보로 구축한다. 또한 각 클래스들의 관계정보도 추출한다. 그림 4는 클래스 정보 표현 구조를 나타낸다. 정보 표현에서 특성과 개념이 리스트로 표현되는 이유는 한 클래스가 여러 개의 특성을 가지고 있고 특성에 대응하는 여러 개의 개념을 가질 수 있음을 의미한다. 그림 5는 클래스 'Button' 정보 표현 예이다. 클래스 'Button'은 'window', 'dialog', 'event', 'toolbar', 'dialogbar'와 같은 특성을 가지고 있으며, 'Control', 'Event', 'Interface'의 개념에 속해 있다. 또한 'Window Application' 도메인에 사용되는 클래스임을 나타내고, 클래스 질의로써 'button', 'press'등을 선택하면 'Button' 클래스가 검색된다. 위와 같이 표현된 클래스 정보는 소스코드 링크정보와 함께 정보저장소에 저장된다. 클래스의 정보는 스트링(예, CLASS), 클래스명, 개념 리스트, 특성 리스트 순으로 저장한다. 또한 전체 클래스 코드를 나타내기 위한 클래스 단위의 코드 정보와 클래스 질의 표현 데이타베이스에도 링크되도록 하였다. 그림 6은 정보저장소 내에 클래스 정보를 저장하는 과정이다.

```
Class-ID : [class name]
·Use : [클래스 질의 표현어 리스트]
·Features : [특성 리스트]
·Concepts : [개념 리스트]
·Domain : [응용 도메인]
·Thesaurus info : [시소러스 정보].
```

그림 4. 클래스 정보 표현

```
Class-ID : Button
·Use : button, press
·Features : window, dialog, event, toolbar, dialogbar
·Concepts :  Control, Event, Interface
·Domain : Window Application
·Thesaurus info : 유의값 0.7이상의 Concept list
```

그림 5. 'Button' 클래스 정보

```
1: Repeat
2: set var m := "CLASS::"
3:     + classname    /* 클래스 이름 */
4:     + "Concept("
5:     + Concept_Names  /* Concept 명 */
6:     + "),Feature("
7:     + Feature_Names      /* Feature 명 */
8:     + ")};"
9: set var thesaurusinfor have  m information
     from thesaurus_table
10: Until end of file
   /*  「Button」 클래스의 관계 정보 예 */
   /* CLASS::Buttonn,
     Concept(Control, Event, Interface)      */
11:  Repeat
12:      Write string "(CLASS(" , class_number, and ")"
13:      Read class name
14: Read source code and class query  「Use DB」
     linked class name
15: Where
       class name links string
       from classname_string_num to  ");" string_num
16: Until end of file
```

그림 6. 클래스 정보 표현 알고리즘

클래스에 의한 컴포넌트의 표현은 컴포넌트의 행위적 특성을 간략하게 표현하고, 사용자로 하여금 컴포넌트의 모듈 별 기능에 대한 이해력을 높이는 장점이 있다. 이는 클래스에 대한 정보를 제공해주고 질의 형성에 도움을 주는 에디터에 의해 용이하게 구현될 수 있도록 하였다.

2. 클래스의 개념적 분류

객체지향 컴포넌트에 대한 시소러스는 전통적인 정보검색 시스템에 이용되는 시소러스 구조 및 관계에 있어서 해석의 차이가 있다. 클래스의 상속관계를 이용하여 개념들 사이의 관계를 자연스럽게 표현해야 한다. 즉, 객체지향 컴포넌트의 특성에 맞는 클래스의 개념적 분류 방법이 필요하다. 이에 따라 본 연구에서는 서로 관련 있는 특성들을 일정한 기준에 따라 그룹화하여 여러 개의 개념 그룹으로 분류하였다. 이러한 분류는 컴포넌트의 행위적 특성을 이용하여 클래스가 사용될 수 있는 여러 경험적 상황을 패싯 항목으로 설정하는 패싯 분류 방법이다[12]. 패싯 분류 방법은 컴포넌트의 확장성을 개선시킨 방법으로 컴포넌트들이 갖는 공통적인 특성을 합성하여 하나의 패싯으로 표현한 후 하나의 컴포넌트를 여러 개의 패싯으로 나타낸다. 이 분류 방법은 컴포넌트들이 갖는 기본적인 클래스만 표현하므로 분류가 간단하며 이해하기가 쉽고, 확장이 용이하다. 컴포넌트의 행위적 특성은 Gamma 설계 패턴 중 행위 패턴(behavioral pattern)을 분류하기 위한 방법에 기반을 둔다[2]. 행위 패턴은 객체의 행위를 조직화, 관리, 연합하는데 사용되는 패턴으로 객체의 기능을 구분하는 알고리즘에 주로 이용된다. 이에 본 연구에서는 각 클래스가 사용될 수 있는 여러 경험적 상황을 클래스 정보로 관리하고 이 정보를 이용하여 패싯 항목으로 설정하는 패싯 분류방법을 사용하였다. 클래스의 사용범위가 한 가지로 국한되지 않고 여러 경우가 가능하며 이에 따른 경험적 솔루션을 제시해준다는 점에서 클래스의 분류와 검색을 위해 클래스 정보를 이용한 패싯 분류는 효율적인 방법이다. 먼저 각 클래스를 표현하는 특성을 소프트웨어 디스크립터로부터 설정한 후, 모든 클래스로부터 나온 특성을 그룹화하여 개념을 생성한다. 그러므로 하나의 클래스는 그 특성에 따라 하나 이상의 개념을 가질 수 있으며, 이는 검색 시 개념을 이용한 시소러스 질의 확장에 이용된다. 그림 7에서와 같이 'Cartesian' 클래스는 'Angle', 'transform', 'Matrix', 'Vector', 'distance_compute', 'Axes', 'Draw', 'Scale', 'label'의 특성을 가지고 있고, 이 특성들은 각각 'Math', 'Linear Math', 그리고 'Graphic' 개념에 포함된다. 그러므로 'Cartesian' 클래스는 3개의 개념에 속하게 된다.

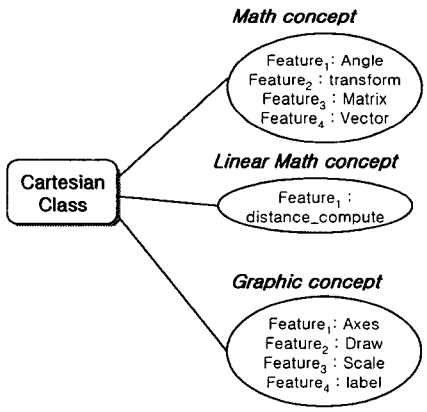

그림 7. 'Cartesian' 클래스의 특성과 개념

클래스는 기능과 개념에 따라 여러 개의 범주로 나눌 수 있다. 이 과정은 도메인 전문가에 의해 행해지며 시스템의 응용에 따라 모든 클래스를 최적으로 표현할 수 있는 개념을 선정한다. 각 개념은 패싯 항목으로 표현되고 이에 따라 클래스가 분류되어지며 시소러스 구축 시 이용되게 된다.

Ⅳ. 개념적 질의 확장

1. 개념 기반 시소러스 구축

본 연구에서는 클래스가 포함된 개념을 확장하기 위하여 개념 기반 시소러스를 제안하였다. 시소러스 구축은 개념과 특성들간의 관련값에 대한 통계적 분석에 의해 이루어진다. 특성과 개념간의 관련값(FCV)이 얻어진 후에, 이를 이용하여 개념과 개념의 관련값(CCV)을 계산한다. 시소러스 구축 과정이다.

① 개념과 특성으로 이루어진 메트릭스를 구성하여 특성-개념관계값 (Feature-Conceptt Value : FCV)을 계산한다. 특성-개념관계값에 대한 수식은 식(1)에 나타나 있다. 이는 각 개념에 속한 특성의 수는 컴포넌트와 개념과의 관련성을 암시해 준다는 의미에 근거한다.

$$FCV_{l,j} = p_j(l)\frac{feature_{l,j}}{feature_l} \qquad (1)$$

$FCV_{l,j}$: *Concept j*와 *feature l*과의 관계값
$p_j(l)$: *Concept j*에서의 *l* 번째 *feature*의 발생백분율
$feature_{l,j}$: *Concept j*에서 *l* 번째 *feature*의 발생횟수
$feature_l$: 모든*Concept*에서 *l* 번째 *feature*의전체발생횟수

② 개념과 개념 관련값(Concept-Concept Value : CCV)을 계산한다. 표 1은 FCV를 이용하여 CCV를 계산하는 방법을 표현한 것이다. 이는 하나의 특성에 대한 두 개념 사이의 매칭정도를 나타내기 위한 것이다. 계산식은 (2),(3)과 같다.

표 1. FCV를 이용한 CCV

Feature \ Concept	C1	C1	...	CC
A	a1	a2	...	ac
B	b1	b2	...	bc
...	...	...	...	...
N	N2	N2	...	Nc

1) 개념 C1과 C2 사이의 매칭정도를 알아보기 위해 먼저 a1과 a2 사이의 매칭정도 계산

$$m_{a_{12}} = \frac{1}{1+|a1-a2|}, \ (a1 \neq 0,\ a2 \neq 0)$$
$$m_{a_{12}} = 0 \quad , \ (a1 = 0, OR\ a2 = 0) \qquad (2)$$
$$\text{또는 } (a1 = a2 = 0)$$

C1과 C2의 모든 특성에 대해서 계산

$$M_{12} = \sum_{j=a}^{N} mj_{12} \qquad (3)$$

2) 모든 개념에 대해서 시행한다.

위와 같은 과정에 의해 최종적으로 개념과 개념 사이의 관련값, 즉 개념간 유의어 테이블이 구성되어 개념을 기본으로 한 시소러스가 구축된다.

2. 질의 확장

시소러스에 의한 개념 확장 과정은 다음과 같다.

① 기본 개념 추출

질의에 해당하는 클래스를 검색한 후 그 클래스의 특성이 포함된 개념을 추출한다. 예를 들어 질의가 'URL'과 'BufferedReader'일 때 이를 만족하는 클래스 'URLReader' 클래스가 검색된다. 'URLReader' 클래스의 특성을 추출한다. 'Exception', 'URL', 'BufferedReader' 그리고 'inputstream'의 특성이 추출되는데, 이들 특성에 대응하는 개념을 살펴보면 'Exception'은 'Debugging' 개념에 포함되고 'URL'은 'Network' 개념에 포함되며 'BufferedReader'와 'inputstream'은 'BufferedReader' 개념에 포함된다. 그러므로 질의 'URL'과 'Buffered-

Reader'에 의해 선택된 기본 개념은 'Debugging', 'Network', 'BufferedReader'이다.

② 개념 확장

추출된 개념을 시소러스에 의해 확장한다. 기본 개념으로 선택된 'Debugging', 'Network', 'Buffered-Reader'는 시소러스에 의한 CCV 테이블을 이용하여 개념이 확장된다. 각 개념은 모든 개념에 대한 관련값을 가지고 있으며, 이중 0.7 이상에 해당하는 개념들만이 확장의 대상이 된다. 이는 정확도를 적절히 유지하면서도 재현율이 높게 나타나는 임계치 범위를 시뮬레이션 통하여 설정한 것으로, 5절에 자세히 설명한다. 표 2는 'Network'에 대한 유의어 테이블의 일부이다. 이중 관련값이 0.7 이상인 'Internet'과 'HttpFile'이 'Network'에 대한 확장 개념으로 선택되어진다.

표 2. 시소러스 유의어 테이블

concept / concept	···	Internet	HttpFile	Archive	Menu	···
···	···	···	···	···	···	···
Network	···	0.78	0.72	0.48	0.61	···
···	···	···	···	···	···	···

③ 확장된 개념의 불리언 연산

불리언 질의는 3 가지 형태의 질의로 표준화할 될 있다. 단순질의(mono-query), 분리질의 (disjunctive -query), 그리고 결합질의(conjunctive-query)가 그것인데, 단순질의는 질의로 하나의 질의어만이 사용된 경우이며, 두 개 이상의 질의어가 있을 경우 분리질의는 OR의 역할을 하고 결합질의는 AND의 역할을 수행한다. 다음은 3 가지 형태의 질의를 정의한 것이다.

단순질의 : $q_\ell = [c_1 : \alpha_1]$

분리질의 : $q_{\ell(\mathrm{EXP})} = OR[q]_{i=1}^{n}$

결합질의 : $Q = AND[q_{\ell(\mathrm{EXP})}]_{\ell=1}^{m}$

본 연구에서의 기본 개념에 대한 질의 확장은 OR로 연결된 분리질의로 이루어지며, 확장된 개념들에 대해서는 OR 연산이 이루어져 후보 컴포넌트를 검색할 수 있도록 하였다. 즉, 확장된 개념 중 하나 이상을 포함하는 클래스는 모두 후보 컴포넌트로 검색된다. 그후, 후보 컴포넌트와 그 컴포넌트가 가지고 있는 개념들 간의 유사도를 계산하여 가장 관련성이 깊은 컴포넌트를 최종 검색 결과로써 선택하도록 한다[13].

V. 성능 평가

1. 확장 범위의 임계치 설정

검색 시스템의 시소러스 효율성은 질의에 대한 효과적인 확장에 달려 있다. 그러므로 시소러스로 구축되어 있는 모든 관계들에 대한 확장 정도와 그에 따른 효율성을 평가하는 것이 중요하다. 이는 검색 시스템이 어떤 면에 초점을 맞추는가에 크게 의존하는데, 예를 들어 검색의 정확성을 중요 시 하는 시스템일 경우에는 질의 확장에 있어서 높은 유의값을 가지는 관계로 한정지을 것이다. 또한 검색 과정 중 용어 불일치 문제를 해결하고자 하거나 재현율을 높이고자 하는 경우에는 질의 확장 범위를 적절히 넓게 설정해야 할 것이다. 따라서 본 연구에서는 컴포넌트 검색의 재현율을 최대한 보장해줄 수 있는 최적의 질의 확장 임계치를 시뮬레이션을 통해 설정하였다.

질의 확장 임계치를 설정하기 위해 먼저 시소러스 내에서 임의의 개념 10개를 선택하고, 선택된 개념에 대한 유사 확장 집합(similar expended sets)을 수동으로 선정하도록 하였다. 유사 확장 집합이란 한 개념에 대해 유사하거나 기능적으로 함께 사용될 수 있는 개념들의 집합을 의미하며, 유사 확장 집합에 속한 개념들은 시소러스에 있는 개념으로 한정한다. 그 후, 질의로 주워진 10개의 개념에 대해 본 연구에서 제안한 시소러스에 의해 확장된 질의와 유사 확장 집합과의 비교를 통해 정확도와 재현율을 측정하였다. 이때, 질의 확장에 있어 유의값을 0.6에서부터 1.0까지 0.1의 간격으로 주었을 때 각각의 정확도와 재현율을 측정하였다. 정확도와 재현율을 측정하는 방법은 전통적인 정보 검색에서 사용하는 방법과 유사하다.

$$재현률 = \frac{확장된유의어\ 중\ 유사확장집합에\ 속한\ 유의어의수}{유사확장\ 집합의\ 수}$$

$$정확도 = \frac{확장된유의어\ 중\ 유사확장집합에\ 속한\ 유의어의수}{확장된\ 유의어의\ 수}$$

최적의 임계치 선택을 위해 임의로 사용된 10개의 개념은 {'EditWindow', 'CommandTarget', 'FileException', 'Window', 'Menu', 'HttpServer', 'OleClientDoc', 'StringArray', 'Document', 'ToolBar'}와 같다. 표 3은 시소러스에서 임의로 추출한 위의 10개 개념에 대한 임계값 별 검색 효율을 나타낸 것이고, 그림 8은 정확도와 재현율의 평균을 임계값에 따라 그래프로 보여준다.

표 3. 임계값 별 검색 효율

임계치	정확도	재현율
0.60	0.287	0.853
0.70	0.603	0.811
0.80	0.734	0.428
0.90	0.829	0.285
1.00	0.870	0.210

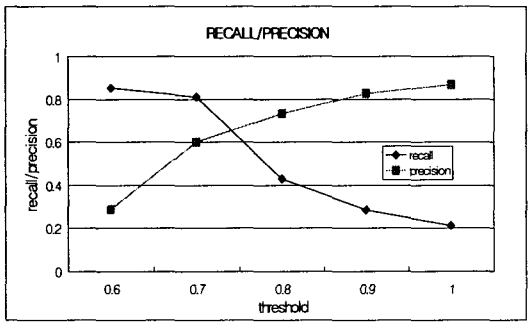

그림 8. 임계값에 따른 정확도/재현율 비교

임계값이 높아질수록 정확도가 좋아지고, 낮아질수록 재현율이 향상됨을 알 수 있다. 그러나 임계값 0.7 미만이 되면 정확도가 0에 가깝게 되어 검색 효율이 떨어지고, 임계값 0.8부터는 0.7과 정확도에 있어서는 별 차이가 나지 않지만 재현율은 상당히 떨어짐을 알 수가 있다. 따라서 본 연구에서는 정확도를 유지하면서 재현율을 최대한 보장할 수 있는 범위인 임계값 0.7 이상을 질의 확장의 범위로 설정하였다. 이에 본 연구에서는 시뮬레이션에 의한 질의 확장의 임계값 설정으로 인해 검색 효율을 최대한 보장할 수 있도록 하였다.

2. 질의 확장 단계

앞 절에서 질의 확장을 위한 최적의 임계값을 설정하는 방법에 대해서 알아보았다. 본 절에서는 임계값 0.7 이상의 질의 확장과 확장 단계에 따른 시소러스의 성능을 평가하도록 한다. 본 절에서는 시소러스의 내의 유의어 관계와 이에 따른 질의 확장 단계의 성능을 평가하는 것이 목적이기 때문에 단일 질의에 대하여 시뮬레이션 하기로 하였다. 방법은 하나의 질의에 대하여 5번의 질의 확장을 시행한 후, 정확도와 재현율의 변화를 비교하는 것이다. 질의로 주워진 클래스에 대하여 임계값 0.7 이상인 모든 클래스를 질의에 포함시키고, 새롭게 질의에 포함된 모든 클래스에 대해서 다시 0.7 이상인 클래스를 질의에 포함시키는 방법으로 총 5번의 질의 확장을 시행한다. 이는 질의가 확장되는 과정에 따라 사용자 질의에 대한 재현율과 정확도가 얼마나 변화하는가를 알기 위함이다. 표 4는 'Window' 개념에 대한 질의 확장 과정을 보여준다.

'Window' 개념을 5번 확장한 결과, 'Window', 'FrameView', 'DaoRecordView', 'CtrlView', 'ScrollView', 'RecordView', 'View', 'AnimateCtrl' 모두 8개의 질의로 확장되었다. 이와 같은 방법으로 10개의 개념에 대해 시행한 결과의 정확도와 재현율 비교가 표 5와 그림 9에 나타나 있다. 사용한 개념은 5.1에서와 같다. 확장 과정이 진행될수록 정확도가 떨어지고, 재현율이 향상됨을 알 수 있다. 그러나 확장이 한번도 이루어지지 않은 경우에는 정확도는 높지만, 재현율이 낮아 검색 효율이 떨어지며, 확장이 3번 이상 이루어진 경우에는 재현율에 있어서는 별 차이가 나지 않지만 정확도가 상당히 떨어짐을 알 수가 있다. 따라서 본 연구에서는 정확도와 재현율을 적절한 수준에서 동시에 만족할 수 있도록 질의 확장을 한번으로 설정하였다.

표 4. "Window" 클래스 확장 과정

확장 과정	질의
Q1	Window
Q2	Window View : 0.84 AnimateCtrl : 0.79
Q3	Window View FrameView : 0.93 ScrollView : 0.77 CtrlView : 0.85 AnimateCtrl
Q4	Window View FrameView CtrlView ScrollView RecordView : 0.80 AnimateCtrl
Q5	Window View FrameView CtrlView ScrollView RecordView DaoRecordView : 0.92 AnimateCtrl

표 5. 확장 과정의 검색 효율

확장 과정	검색 효율	
	정확도	재현율
Q1	0.821	0.265
Q2	0.598	0.793
Q3	0.234	0.834
Q4	0.205	0.849
Q5	0.156	0.856

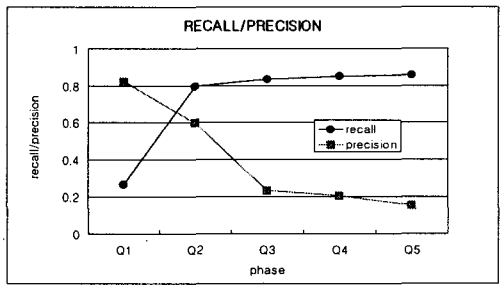

그림 9. 확장 과정에 따른 정확도/재현율 비교

3. 제안한 시소러스 평가

본 연구에서는 시스템의 효율성을 평가하기 위하여 검색 효율성의 기준이 되는 재현율과 정확도를 측정하였다. 측정 방법은 제안한 개념 기반 시소러스에 의해 검색했을 때와 시소러스로 구축되지 않았을 경우의 검색에 대해서 재현율과 정확도를 비교하였다. 재현율은 전체 컴포넌트 중 적절한 컴포넌트의 수에 대한 검색된 적절한 컴포넌트의 수, 정확도는 검색된 전체 컴포넌트 수에 대한 검색된 적절한 컴포넌트의 수를 이용하여 측정하였다. 질의는 임의로 30개를 선정하였고, 시소러스를 사용하지 않은 것과 본 연구에서 제안한 방법으로 시소러스를 사용한 경우의 재현율을 0.1 단위로 변화시키면서 정확도의 변화를 측정한 후, 정확도의 평균을 비교하였다[14]. 표 6은 두 경우의 정확도와 재현율의 비율을 나타낸 것이다. 이 실험에서 본 연구에서 제안한 시소러스에 의한 개념 기반 검색은 시소러스를 사용하지 않은 검색에 비해 효율성에 있어서 22.3%(((0.769-0.629)/0.629) *100) 정도 크게 향상되었음을 보여주고 있다.

표 6. 재현율과 정확도의 비율

Recall	No-시소러스 Precision	개념 기반 시소러스 Precision
0.1	0.68	1.00
0.2	0.70	0.95
0.3	0.78	0.92
0.4	0.75	0.82
0.5	0.72	0.86
0.6	0.63	0.72
0.7	0.59	0.71
0.8	0.54	0.65
0.9	0.48	0.57
1.0	0.42	0.49
평균 Precision	0.629	0.769
향상된 평균 비율	-	22.3%

표 7. 시소러스 비교 분석

항목 / 시스템	검색대상	범주분류	질의연산 방법	구성항목
No Thesaurus	함수	단순 category	string matching	함수형 데이터형
객체기반 시소러스[6]	사물	단순 category	불리언 연산	개념
신경망 시소러스[7]	문서	단순 category	확장 불리언 연산	용어
계층적 시소러스[8]	클래스	context	context 조합	함수명 인자명
제안한 시소러스	클래스	concept	불리언 연산	특성 개념

표 7의 시소러스 비교 분석을 보면 타 시스템은 검색 대상이 함수, 사물, 문서로 국한되어 있기 때문에 진정한 의미의 객체지향 컴포넌트의 검색이라고 볼 수 없다. 특히 계층적 시소러스는 상속관계를 고려한 객체지향 시소러스이긴 하지만, 코드에서 추출한 함수명과 인수명을 가지고 시소러스를 구축하기 때문에 데이터 양이 너무 방대해져 구축과 관리, 갱신에 어려움이 따르고, 숙련된 전문가도 사용하기 어려운 단점이 있다.

Ⅵ. 결 론

본 논문은 특정 클래스로부터 개념적으로 서로 연관있는 클래스를 검색하기 위하여 개념 기반 시소러스를 통한 질의 확장 방법을 제안하였다. 제안한 방법은 클래스가 가지는 특성에 따라 클래스를 개념으로 분류하고 모든 개념에 대해 개념별 관련값을 이용하여 개념 기반 시소러스 유의어 테이블을 구성하였다. 초기 질의에 의해 검색된 클래스의 개념은 시소러스를 통해 확장되는데, 이때 임계치 이상의 유의값을 가지는 개념이 선택된다. 질의 확장의 임계치를 조절하여 효율성을 시뮬레이션한 결과, 유의값 0.7이상인 경우에 재현율과 정확도에 있어서 평균 81.1%, 60.3%이상을 보였으며, 질의 확장 과정이 1회인 경우에는 79.3%, 59.8%의 값을 보여줌으로써 시뮬레이션 결과 가장 좋은 임계치를 설정할 수 있었다. 이로 인해 제안한 개념 기반 질의 확장의 효율성이 시소러스를 사용하지 않았을 때의 검색에 비해 효율성에 있어 22.3% 정도 크게 향상되었음을 보여주었다. 본 연구는 개념을 이용한 질의 확장을 통하여 후보 컴포넌트까지 검색하여 컴포넌트 선택의 범위를 넓힐 수 있었으며, 클래스 라이브러리에 대한 개념을 이용하여 컴포넌트들을 검색하고 우선순위로 표현하기 때문에 컴포넌트 조립시 보다 효율적이다. 따라서 본 연구는 객체지향 컴포넌트의 검색을 효율적으로 수행할 수 있는 개념 기반 시소러스 질의 확장 방법을 제안하였으며, 시뮬레이션을 통하여 그 유용성을 입증하였다.

그러나 본 연구는 클래스의 수가 적은 도메인이나 특정 범주에 클래스가 몰려있는 환경에서는 효율성이 떨어지는 경향이 있으며, 또한 컴포넌트들에 대한 정확한 기능 정의에만 한정되어 있어서 더 많은 컴포넌트의 이해를 위한 정보가 요구된다.

참 고 문 헌

[1] D. Harman, "Overview of the first TREC conference," IIn Proceedings of the 16th Annual International ACM SIGIR Conference, 1993.

[2] E.Gamma, R.Helm, R.Johnson, and J.Vlissides, "Design Pattern : Elements of Reusable Object-Oriented Software," Addison-Wesley, 1995.

[3] F.A.Grootjen, and Th.P. van der Weide, "Conceptual Query Expansion," Jau., 2004.

[4] R. Rada, H. Mili, E. Bickenell and M. Blettner, "Development and Application of a Metric on Semantic Nets," IEEE Transaction on System, Mand Cybernetics Vol.19, No.1, pp. 17-30. 1989.

[5] H. Chen, T. Tim and D. Fye "Automatic Thesaurus Generation for an Electronic Community System," Journal of the American Society for Information Science, Vol.46, No.3, pp. 175-193, 1995.

[6] 최재훈, 한종진, 박종진, 양재동, "구조적인 시소러스 구축을 지원하는 객체 기반 정보 검색 모델", 한국정보과학회논문지, Vol.24, No.11, pp. 1244-1256, Nov. 1997.

[7] 최종필, 최명복, 김민구, "신경망을 이용한 적응형 시소러스", 한국정보과학회논문지, Vol.27, No.12, pp. 1211-1218, Dec. 2000.

[8] E. Damiani, M. G. Fugini and C. Bellettini, "Aware Approach to Faceted Classification of Object-Oriented Component," ACM Transaction on Software Engineering and Methodology, Vol.8, No.4, pp. 425-472, Oct. 1999.

[9] Y. S. Maarek, D. M. Berry and G. E. Kaiser, "An information retrieval approach for automatically constructing software library," IEEE Transaction on Software Engineering, Vol.18, No.8, pp. 800-813, Aug. 1996.

[10] D. Batory and S. O'Malley, "The design and implementation of hierarchical software systems with reusable components," ACM Transaction on Software Engineering and Methodology, Vol.1, No.4, pp. 355-398, Oct. 1992.

[11] B.A. Burton, R.W. Aragon, S.A. Bailey, K.D koehler and L.A. Mayers, "The Reusable Software Library," IEEE Software, pp. 25-33, July 1987.

[12] R. Prieto-Diaz, "Implementing Faceted Classification for Software Reuse," Proceedings of the 2th International Conference on Software Engineering, pp. 300-304, March 1990.

[13] 김귀정, 한정수, 송영재, "컴포넌트 검색을 지원하는 퍼지 기반 시소러스 구축", 한국정보처리학회논문지, 제10-D권, 제5호, pp. 753-762, 2003.

[14] B. Y. Ricardo and R. N. Berthier, "Modern Information Retrieval," Addison-Wesley, 2000.

저 자 소 개

김 귀 정(Gui-Jung Kim) 정회원

- 1994년 : 한남대학교 전자계산공학과(공학사)
- 1996년 : 한남대학교 전자계산공학과(공학석사)
- 2003년 : 경희대학교 전자계산공학과(공학박사)
- 2001년~현재 : 건양대학교 컴퓨터공학과 조교수

<관심분야> : CBD, CASE 도구, 컴포넌트 검색

한 정 수(Jung-Soo Han) 정회원

- 1990년 : 경희대학교 전자계산공학과(공학사)
- 1992년 : 경희대학교 전자계산공학과(공학석사)
- 2000년 : 경희대학교 전자계산공학과(공학박사)
- 2001년~현재 : 천안대학교 정보통신학부 조교수

<관심분야> : CBD, 컴포넌트 관리, CASE 도구