Journal of Korea Multimedia Society Vol. 16, No. 7, July 2013(pp. 802-809)
http://dx.doi.org/10.9717/kmms.2013.16.7.802

프로그램 코드 분석을 위한 유사도 측정 및 가시화 기법

이영주†, 이정진††

요 약

본 논문에서는 프로그래밍 언어에 정의되는 지정자와 키워드가 프로그램 코드 상에서 연속적인 패턴으로 나타나게 될 때, 해당 연속 패턴들의 빈도와 길이를 측정하여 두 코드 사이의 유사성을 측정하는 기법을 제안한다. 또한, 이러한 분석 결과를 정형적 개념 분석 기법을 이용하여 가시화하는 기법을 제안한다. 제안 기법은 기존의 유사도 측정 기법에서는 고려하지 않았던 단어 인접성을 유사도 측정에 반영한다. 함수 단위로 지정자와 키워드 패턴을 이용하여 함수의 호출 순서나 수행 순서에 상관없이 표절을 탐지할 수 있다. 또한, 유사도 측정 결과는 정형적 개념 분석 기법을 이용하여 격자(lattice)로 시각화되어 사용자의 이해도를 높일 수 있다. 실험 결과 제안 기법은 96%의 표절 탐지 성공률을 보여주었다. 제안 기법은 프로그램 코드 뿐만 아니라 일반 문서의 분석에도 적용될 수 있다.

A Similarity Measurement and Visualization Method for the Analysis of Program Code

Youngjoo Lee†, Jeongjin Lee††

ABSTRACT

In this paper, we propose the similarity measurement method between two program codes by counting the frequency and length of continuous patterns of specifiers and keywords, which exist in two program codes. In addition, we propose the visualization method of this analysis result by formal concept analysis. Proposed method considers adjacencies of specifiers or keywords, which have not been considered in the previous similarity measurements. Proposed method can detect the plagiarism by analyzing the pattern in each function regardless of the order of function call and execution. In addition, the result of the similarity measurement is visualized by the lattice of formal concept analysis to increase the user understanding about the relations between program codes. Experimental results showed that proposed method succeeded in 96% plagiarism detections. Our method could be applied into the analysis of general documents.

Key words: Similarity Measurement(유사도 측정), Formal Concept Analysis(정형적 개념 분석), Pattern Analysis(패턴 분석), Concept Lattice(컨셉 격자)

1. 서 론

인터넷과 스마트폰의 보급과 워드프로세서의 기능 향상으로 디지털화된 자료 및 정보의 공유가 매우 용이해졌고, 인터넷 사용자가 접근이 가능한 웹 문서의 양은 수십억 개 이상으로 추산되고, 현재에도 기

※ 교신저자(Corresponding Author) : 이정진, 주소 : 서울특별시 동작구 상도1동 숭실대학교 컴퓨터학부 정보과학관 328호(156-743), 전화 : 02) 820-0911, E-mail : leejeongjin@ssu.ac.kr
접수일 : 2013년 4월 10일, 수정일 : 2013년 5월 14일
완료일 : 2013년 5월 28일

† 정회원, 삼성전자 생산기술연구소 책임연구원
(E-mail : spicio2@gmail.com)
†† 정회원, 숭실대학교 컴퓨터학부 조교수
※ 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임 (No. 2012R1A1A2043819).

하급수적으로 증가하고 있다[1]. 그러나 이러한 디지털화된 정보의 폭발적인 증가 속도에 비하여 보안이나 복제 방지에 대한 대책은 미흡한 실정이며 인터넷과 스마트폰 등의 다양한 매체의 활성화에 따른 디지털화된 정보의 급격한 증가는 정보 복제를 더욱 쉽게 만드는 주요인이 되고 있다[2]. 디지털화된 정보 중 프로그램 코드는 특히 복제가 빈번하게 일어나고 있고, 특히 프로그램을 교육하는 경우 학생들이 제출하는 프로그램 과제의 표절 여부를 육안으로 검사하기가 어렵기 때문에 이를 자동으로 탐지해주는 시스템이 필요하다[3]. 또한, 컴퓨터 프로그램 코드에 대한 저작권은 지적재산권의 한 종류로서 마땅히 보호되고, 관리되어야 하지만, 그 침해와 피해 사례는 늘어만 가고 있다[4]. 이러한 필요성에 의하여 최근 프로그램 코드들을 분석하여 상호 유사성을 측정하고, 표절 여부를 판단하는 탐지 기법들에 대한 연구가 활발히 수행되고 있다.

기존에 제안된 프로그램 코드 분석 및 표절 탐지 기법은 크게 지문(fingerprint)법과 구조 검사법으로 구분된다. 지문법은 두 개의 코드에 사용된 키워드들의 빈도수나 유사성을 분석하여 프로그램 코드의 통계적인 특징을 추출하고, 이를 기반으로 표절 검사를 수행한다[3,5-7]. 지문법은 분석해야할 프로그램 코드의 길이에 영향을 받지 않고, 프로그램 코드 내에 구문의 순서를 변경하더라도 사용된 지정자(specifier), 키워드(keyword)의 빈도수 등을 기준으로 고속으로 분석이 가능하다. 하지만, 프로그램 내에 일부분을 추가로 삽입한 경우에 부분적인 표절 탐지는 어렵다는 문제점이 있다. Grune 등은 프로그램 코드 내에 토큰들이 다른 곳에서 참조되는 횟수를 계산하여 히스토그램을 생성하고, 이 히스토그램을 비교하여 표절 여부를 판단하였다[5]. 한 등은 식별자의 동일 여부를 고려한 지문법 기반 탐지 기법을 제안하였다[6,7]. 제안 기법은 오픈 소스를 분석하여 표절 여부를 탐지하였고, 구글의 코드 검색 API와 연동하여 분석이 가능하였으나 사용자가 원본 프로그램의 식별자 명을 바꾼다면, 오픈 소스 코드의 표절 탐지가 불가능하다는 문제점이 있다. 손 등은 프로그램 코드들의 유사도를 측정하는 방법을 제안하였고, 프로그램 코드를 계층적으로 군집화하고, 분석 결과를 수형도로 시각화하는 방법을 제안하였다[3]. 문자 개수, 단어 개수, 키워드 개수, 식별자 개수로 구성된 특징을 추출하여 4차원 공간 상의 점으로 표현한 후 두 프로그램 사이의 관계를 두 점 사이의 유클리디안 거리로 모델링하였다. 또한, 수형도에서 사용자가 입력한 임계값이 임의의 수직선으로 표현되어 표절 분석의 민감도를 사용자가 원하는 수준으로 설정할 수 있다는 장점이 있었다. 하지만, 프로그램의 제어 흐름을 고려하지 않았기 때문에 사용자가 프로그램의 실행과 무관한 코드를 삽입하는 경우 표절 여부의 판단에 실패하는 문제점이 있다.

구조 검사법은 제어 구조를 가지고 있는 프로그램 표절 탐지에 많이 사용된다[2]. 구조 검사법은 분석해야할 프로그램 코드가 길수록 얻을 수 있는 정보량이 비례하고, 부분적인 표절 탐지가 가능하다. 특히 일반 문서와 달리 프로그램 코드는 제어 흐름의 변경이 어렵고, 프로그래밍 문법이 정해져 있기 때문에 구조적인 특징이 잘 나타난다[2]. Si 등은 문서의 구조를 먼저 분석하고, 중요도가 높은 키워드를 추출하여 이를 특징 벡터로 비교하는 기법을 제안하였다[8]. 문서 내에 문서의 구조를 포함하는 LaTex 문서에서 구조 트리를 구성하고, 이 트리를 토대로 중요도가 높은 키워드 분포도를 알아내어 이를 비교하였다. 하지만, 이 기법은 LaTex으로 작성된 문서에 대해서만 분석이 가능하다는 제약 조건으로 인하여 프로그램 코드 분석에는 적용이 어렵다. 김 등은 함수 선형화 기법을 사용하여 프로그램 코드의 구조적 특징을 분석하는 기법을 제안하였다[4]. 또한, 관련 정보와 유사 구간을 시각적으로 표시하여 가독성을 증가시켰다. 함수 선형화 기법을 이용하여 프로그램 코드의 실제적인 수행 절차와 같이 토큰들을 재배치하여 의미 없는 코드의 삽입이나 함수의 위치 변경 등에도 강인한 분석 결과를 보여주었다. 하지만, 동적 프로그래밍 구현 기법의 사용에 따른 연산 부담이 크고, 기법의 적용이 Java 언어에만 국한된 단점이 있다. 손 등은 프로그램 코드로부터 parse tree를 추출하여 구조적인 정보를 추출한 후 parse tree들 사이의 유사도를 측정하는 기법을 제안하였다[9]. 기존 기법들에 비하여 변수, 함수, 클래스의 이름을 바꾸거나 함수와 클래스를 융합하거나 분리하는 등의 간단한 표절 기법들에 대하여 좋은 성능을 보여주었으나 다량의 불필요한 코드를 계속 삽입할 경우 제안 기법의 유사도 값이 크게 감소하는 문제점이 있었다.

본 논문에서는 프로그램 코드 분석 및 표절 탐지

를 위하여 기존의 지문법과 구조 검사법의 장점을 동시에 취할 수 있는 유사도 측정 및 가시화 기법을 제안한다. 본 논문에서는 프로그래밍 언어에 정의되는 지정자(specifier)와 키워드(keyword)가 코드 상에 연속적인 패턴으로 나타나게 될 때, 특정 지정자와 키워드 연속 패턴들의 길이와 빈도를 측정하여 두 코드 사이의 유사성을 측정한다. 또한, 이러한 분석 결과를 정형적 개념 분석 기법을 이용하여 가시화하는 기법을 제시한다. 기존의 벡터 공간 모델과 같은 단일 단어 기반 분석 기법은 단어의 인접 관계를 고려하지 않지만, 제안 기법은 단어의 인접 관계를 분석하여 지문법의 단점인 코드의 부분적인 표절 탐지의 어려움을 해결할 수 있고, 함수의 호출이나 수행 순서에 무관하게 표절을 탐지할 수 있는 장점을 갖는다. 또한, 유사도 측정 결과는 정형적 개념 분석 기법을 이용하여 격자로 시각화되어 사용자가 프로그램 사이의 관계에 대한 이해도를 높일 수 있다.

본 논문은 다음과 같은 순서로 구성되어 있다. 2장에서는 본 논문에서 제안한 지정자와 키워드 패턴 기반 유사도 측정 기법에 대하여 설명한다. 3장에서는 본 논문에서 제안한 프로그램 코드 간 관계 가시화 기법에 대하여 설명한다. 4장에서는 실험 결과를 기술하고, 5장에서는 결론을 맺는다.

2. 지정자와 키워드 패턴 기반 유사도 측정 기법

2.1 색인 구조와 색인 그래프

본 논문에서는 비교 및 분석해야 하는 프로그램 코드들에 공통으로 사용된 지정자(specifier)와 키워드(keyword)의 연속된 패턴을 프로그램 코드들 사이의 유사도 측정의 근거로 사용한다. 일반적으로 프로그램 코드에서 지정자는 char, int, float, short, void, static, public, new 등이 있고, 키워드는 if, for, return, switch, do, else, this, case, break 등이 있다. 따라서 여기에 사용될 색인 구조 또한 이러한 지정자와 키워드의 공통 연속 패턴을 잘 표현할 수 있어야 한다.

일반적인 문서 사이의 유사도 측정에서 일반적으로 쉽게 사용되는 방법이 벡터 공간 모델(Vector Space Model, VSM)이다. VSM은 모든 문서 집합에서 색인 단어를 추출한 후 각각의 문서를 이 색인 단어의 특징 벡터로 나타낸다. 각각의 특징 벡터의 요소는 해당 문서에 나타난 단어의 빈도수를 사용하여 가중치가 부여되며, 두 문서 간의 유사도는 이러한 특징 벡터를 비교하여 측정된다. VSM은 색인 구조로 단어-문서 행렬을 사용하며 예약어와 프로그램 코드를 각각 단어와 문서에 대응시킬 때 단어-문서 행렬을 하나의 색인 구조의 대안으로 생각할 수 있다. 그러나 이러한 구조는 지정자와 키워드의 연속된 패턴 정보가 표현이 되지 않아 올바른 분석이 힘들다. 따라서 본 논문에서는 기존 단어-문서 행렬과 다른 색인 구조를 사용해야하며 DIG(Document Index Graph) 구조를 프로그램 코드 분석 도메인에 알맞게 보완하여 사용한다[10].

본 논문의 색인 그래프는 방향 그래프 $G=(V,E)$ 구조를 갖는다. 이 때 V,E는 다음과 같다. $V:=\{v_1,v_2,...,v_n\}$, 노드의 집합으로, 노드 v는 전체 프로그램 코드 집합에서 미리 정의된 지정자 또는, 키워드를 나타낸다. 즉, 예를 들면, char, for 등이 될 수 있다. $E:=\{e_1,e_2,...,e_m\}$, 간선의 집합으로, 간선 e는 노드의 순서쌍 $\langle v_i,v_j\rangle$ 를 나타내며, 노드 v_i로부터 노드 v_j로의 연결을 의미한다. 간선 $\langle v_i,v_j\rangle$ 는 어떤 함수에서 지정자 또는, 키워드 v_j가 지정자 또는, 키워드 v_i 뒤에 연속하여 나타남을 의미한다. 다음 그림 1은 두 개의 프로그램 코드에서 추출한 공통으로 사용된 지정자 또는 키워드의 연속된 패턴을 그래프로 표현한 예이다. 본 그림에서 간선은 구분을 위하여 다양한 종류의 화살표로 표시되었다.

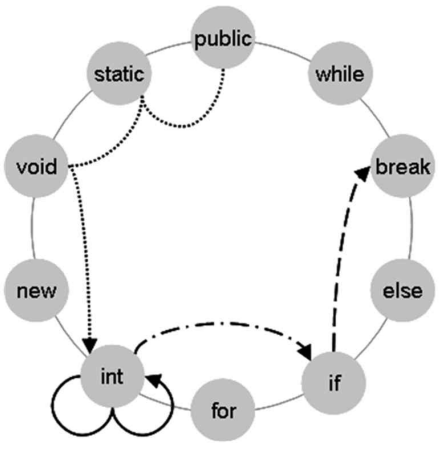


그림 1. 색인 그래프의 예시

2.2 지정자와 키워드 패턴 추출 기법

본 연구에서는 [10]에서 제안된 웹 페이지에서 일

치 구문을 추출하기 위한 알고리즘을 보완하여 프로그램 코드 전처리 과정 및 최장 공통 패턴(longest common pattern)을 찾는 과정을 추가하였다. 먼저 m_{ij}를 i번째 프로그램 코드 p_i내에 포함된 j번째 함수로 정의한다. 프로그램 코드 전처리 과정은 i번째 프로그램 코드 p_i내의 c_i개의 각각의 클래스들에 대하여 코드가 클래스 멤버 함수에 포함되지 않는다면, 비록 코드가 함수로 정확하게 분류되는 것은 아니지만, 구현 편의상 순서대로 코드를 $m_{i0},...,m_{ic_i-1}$ 에 할당한다. 다음으로 각 클래스의 멤버 함수들을 m_{ic_i}부터 할당한다. 다음으로 모든 함수 m_{ij} 내에 지정자와 키워드가 아닌 그 외의 요소들을 삭제한다. 각 프로그램 코드들의 모든 함수들의 지정자와 키워드를 순회하면서 각 프로그램들에서 공통으로 사용되는 지정자와 키워드 패턴을 탐색하여 가장 길이가 긴 것을 중심으로 색인 그래프를 생성 및 갱신한다.

2.3 유사도 측정 기법

일치 구문을 이용한 유사도 측정 기법은 [10]에서 연구된 바 있으나, 빈도 차이를 반영하지 못하는 단점이 있었다. 따라서 본 연구에서는 빈도 차이를 반영할 수 있는 유사도 측정 기법을 제안한다. 제안 유사도 측정 기법은 근본적으로 유사도 측정에 널리 사용되는 식 (1)의 Dice coefficient [11]의 개념에 기반을 두고 있다.

$$D(A,B)=\frac{2|A\cap B|}{|A|+|B|} \tag{1}$$

두 프로그램 코드 상에 공통으로 나타나는 지정자와 키워드의 연속된 패턴의 숫자와 길이와 빈도를 적절하게 가중치를 두어 반영하여 두 프로그램 코드 간 유사도를 다음과 같이 계산할 수 있다.

$$\frac{\sum_{i=1}^{P}[l_i^{\alpha}(f_{1i}+f_{2i})^{\beta}\left(\frac{\min\{f_{1i},f_{2i}\}}{\max\{f_{1i},f_{2i}\}}\right)]}{\sum_j|m_{1j}|+\sum_k|m_{2k}|} \tag{2}$$

위 식에서 $|m_{1j}|$, $|m_{2k}|$는 프로그램 코드 1, 2의 각 함수의 길이이고, f_{1i}, f_{2i}는 프로그램 코드 1, 2에서 공통으로 나타나는 지정자와 키워드의 연속된 패턴의 빈도를 나타내고, l_i는 그 길이를 나타낸다. α는 지정자와 키워드의 연속된 패턴의 길이에 대한 가중치를 얼마나 부여할지에 대한 인자이고, β는 지정자와 키워드의 연속된 패턴의 빈도에 대한 가중치를 얼마나 부여할지에 대한 인자이다.

3. 프로그램 코드 간 관계 가시화

3.1 정형적 개념 분석

정형적 개념 분석은 이항 관계를 분석하기 위한 수학적 기법이다. 정형적 개념 분석은 주어진 자료를 컨셉(concept)이라 불리는 추상 개념으로 구조화한 후, 하위 개념과 상위 개념 관계로 순서화한 컨셉 격자(concept lattice)로 표현한다[12]. 정형적 개념 분석의 입력으로 사용되는 자료는 컨텍스트라고 부르며, 객체의 집합 E(Entity), 속성의 집합 F(Feature), 객체와 속성의 이항 관계 집합 R(Relationship)로 이루어진 트리플 (E,F,R) 이다. 집합이 유한일 때, 분석 대상의 컨텍스트는 표로 나타낼 수 있고, 다음 표 1은 그 사례이다.

주어진 객체의 집합 $E'(\in E)$ 에 속하는 객체의 공통 속성(Common Feature) CF는 다음과 같다.

$$CF(E')=_{\text{def}}\{f\in F\mid\forall e\in E',\ (e,f)\in R\} \tag{3}$$

이와 유사하게, 주어진 속성의 집합 $F'(\in F)$ 에 속하는 속성의 공통객체(Common Entity) CE는 다음과 같다.

$$CE(F')=_{\text{def}}\{e\in E\mid\forall f\in F',\ (e,f)\in R\} \tag{4}$$

예를 들어, 표 1에서 $CF(\{e_4, e_6, e_8, e_9, e_{10}\})=\{c\}$이고, $CE(\{s\})=\{e_1, e_4, e_9\}$이다.

컨텍스트 (E,F,R)의 컨셉은 $CF(E')=F'$, $CE(F')=E'$ 를 만족하는 (E',F')로 정의된다. 즉, 컨

표 1. 컨텍스트의 사례

	composite (c)	even (e)	odd (o)	prime (p)	square (s)
e_1			○		○
e_2		○		○	
e_3			○	○	
e_4	○	○			○
e_5			○	○	
e_6	○	○			
e_7			○	○	
e_8	○	○			
e_9	○		○		○
e_{10}	○	○			

셉 (E',F')에서 객체 E'는 속성 F'를 공통요소로 가지며, 속성 F'는 객체 E'를 공통 요소로 가진다. 컨셉은 다음과 같이 부분 순서를 갖는다.

$$(E_i,F_i) \le (E_j,F_j) \Leftrightarrow E_i \subseteq E_j \quad (5)$$

마찬가지로 다음 관계도 성립한다.

$$(E_i,F_i) \le (E_j,F_j) \Leftrightarrow F_i \subseteq F_j \quad (6)$$

이와 같은 방법으로 모든 컨셉을 순서화하면, 그림 2와 같은 컨셉 격자를 얻을 수 있다. 다이어그램의 하위 노드는 많은 속성을 가진 컨셉을, 상위 노드는 많은 객체를 가진 컨셉을 의미한다. 따라서 하위노드로 갈수록 구체적인 개념이, 상위 노드로 갈수록 일반적인 개념이 나타난다. 이와 같이 컨셉 격자를 이용하면, 주어진 컨텍스트를 시각화 할 수 있어 주어진 대상에 대한 분석이 용이해진다.

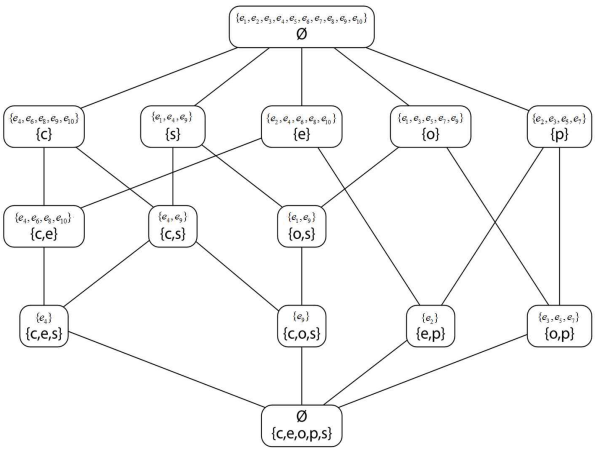


그림 2 컨셉 격자에 대한 예시

3.2 정형적 개념 분석을 이용한 가시화

지정자와 키워드 패턴 추출 기법을 이용하면 프로그램 코드들에 공통으로 사용된 지정자와 키워드의 연속된 패턴으로 정의되는 프로그램 코드 인덱스를 생성할 수 있는데, 인덱스의 빈도를 1 이상과 0으로 구분하면 프로그램 코드와 인덱스는 이항 관계로 표시할 수 있다. 이항 관계로 표시된 프로그램 코드와 인덱스를 각각 객체 집합과 속성 집합으로 놓으면 하나의 컨텍스트를 생성할 수 있다. 컨텍스트에 기반한 컨셉 격자는 프로그램 코드 간의 관계를 부분 순서를 가지는 상위 컨셉과 하위 컨셉 관계로 시각화한다. 하위 컨셉은 상위 컨셉이 가지는 모든 인덱스를 가지며, 상위 컨셉이 가지지 않은 하나 이상의 인덱스를 더 가진다. 따라서 컨셉이 컨셉격자의 아래쪽에 위치할수록 더 많은 인덱스를 가진다. 이와 같은 컨셉 격자의 성질을 이용하면, 다음과 같은 방법으로 프로그램 코드 간의 관계를 시각적으로 이해할 수 있다.

유사도가 높은 프로그램 코드들은 인덱스로 사용하는 지정자와 키워드 패턴을 많이 공유하므로 컨셉 격자 상에서 서로 가까운 컨셉에 속하게 되며, 유사도가 낮은 프로그램 코드는 지정자와 키워드 패턴을 적게 공유하므로 서로 먼 컨셉에 속하게 된다. 동일 컨셉에 속한 프로그램 코드는 완전히 동일한 지정자와 키워드 패턴을 가지며, 유사도 차이는 공통 지정자와 키워드 패턴의 빈도로만 결정된다. 유사도 측정 방법에서 빈도가 미치는 영향은 비교적 작은 편이므로, 같은 컨셉에 속한 프로그램 코드 간의 유사도는 매우 높게 될 것이다. 컨셉 격자의 아래쪽에서 가까운 컨셉에 속한 프로그램 코드일수록 공유하는 지정자와 키워드 패턴이 많아지므로, 컨셉에 속한 프로그램 코드 간의 유사도가 높아지게 된다. 즉, 하위 컨셉에 속한 프로그램 코드 간의 유사도는 상위 컨셉에 속한 프로그램 코드 간의 유사도보다 높게 된다. 특히 최하위 부분에 위치한 컨셉에 속한 프로그램 코드들은 표절일 확률이 매우 높게 된다. 그러나 최상위 부분에 위치한 컨셉에 속한 프로그램 코드들은 일반적으로 사용되는 코드에서 추출된 지정자와 키워드 패턴을 가지고 있으므로 표절이 아니다.

4. 실험 결과

먼저 본 논문에서 제안한 지정자와 키워드 패턴 추출 기법을 이용하여 C++ 언어로 작성된 최대공약수와 최소공배수를 구하는 프로그램 코드에 대하여 원본 코드 1과 원본 코드 1에 대하여 함수를 분할하고, 코드를 재배치하는 변형이 가해진 코드 2에 대하여 지정자와 키워드 패턴을 추출하면 지정자와 키워드 패턴의 종류와 길이, 빈도에 관하여 표 2와 같은 분석 결과를 얻을 수 있다. 표 2에서 볼 수 있듯이 다수의 지정자와 키워드 패턴이 공통으로 두 코드에서 검출되는 것을 확인할 수 있고, 제안 기법이 표절 탐지에 유용함을 확인할 수 있었다.

식 (2)의 유사도 측정을 위해서는 지정자와 키워드의 연속된 패턴의 길이에 대한 가중치와 지정자와

표 2. 지정자와 키워드 패턴 분석 결과

지정자와 키워드 패턴	l_i	f_{1i}	f_{2i}
void main void int new int int new int	9	1	1
int int int int	4	1	2
int if else	3	1	1
while if break else	4	1	1
void main void int	4	1	1
int int int	3	2	4
int if	2	1	1
if break	2	1	1

키워드의 연속된 패턴의 빈도에 대한 가중치의 인자 설정이 필요하다. 전자의 가중치 값이 후자의 가중치 값보다 매우 크게 될 경우 제안 유사도 측정 기법은 구조 검사법의 성향을 보여주게 되고, 반대로 후자의 가중치 값이 전자의 가중치 값보다 매우 크게 될 경우 지문법의 성향을 보여주게 된다. 따라서 두 가중치 값은 유사한 범위의 값으로 설정되어야 제안 유사도 측정 기법이 구조 기반 검사법과 지문법의 성향을 동시에 내포할 수 있다. 지정자와 키워드의 연속된 패턴의 길이에 대한 가중치 인자 α는 1.4로 지정자와 키워드의 연속된 패턴의 빈도에 대한 가중치 인자 β는 1.2로 실험적으로 최적의 값으로 설정하였다.

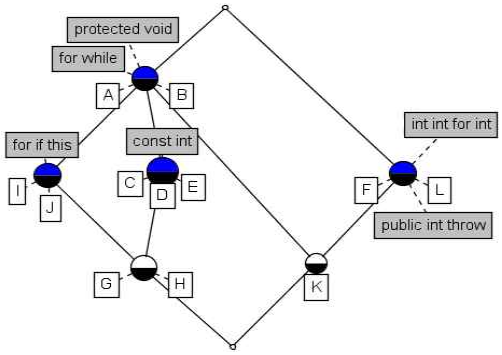
다음으로 본 논문에서 제안한 알고리즘 및 유사도 측정 기법을 사용하여 $n \times n$ 행렬에 대한 고유 값과 고유 벡터를 구하는 프로그램 코드 간 유사도를 측정하였다. C++ 언어로 작성된 원본 프로그램 코드에 대하여 멤버 함수의 분할 및 통합, 코드 재배치, 코딩 스타일 변형, 의미 없는 코드의 삽입 등으로 원본 코드 표절한 코드를 총 100개 작성하여 실험하였다. 제안 기법은 총 100개의 코드 중 96개의 코드를 표절로 탐지하여 96%의 탐지 성공률을 보여주었다. 손 등의 방법[3]에서 제안한 유클리디안 거리에 기반을 둔 유사도 측정 기법으로 동일한 실험을 반복한 결과 탐지 성공률이 최대가 되도록 사용자가 최적의 임계값을 설정하였음에도 불구하고, 83%의 탐지 성공률을 보여주었다. 또한, 다양한 알고리즘들을 C++ 언어로 작성한 10개의 서로 다른 원본 프로그램 코드들에 대하여 제안 기법과 손 등의 방법[3]은 10개 모두 표절이 아닌 것으로 분석하여 0%의 오탐지율을 보여주었다.

컨셉 격자를 통한 프로그램 코드 간 관계 시각화 및 해석을 위하여 임의의 프로그램 코드 12개를 사용하였다. 실험을 위하여 임의의 프로그램 코드 12개를 이용하여 공통으로 사용된 지정자와 키워드 패턴을 추출하고, 컨텍스트를 생성하였다. 그리고, 컨텍스트를 이용하여 컨셉 격자로 프로그램 코드 간의 관계를 시각화하였다. 컨셉 격자 생성은 Concept Explorer [13]를 사용하였다.

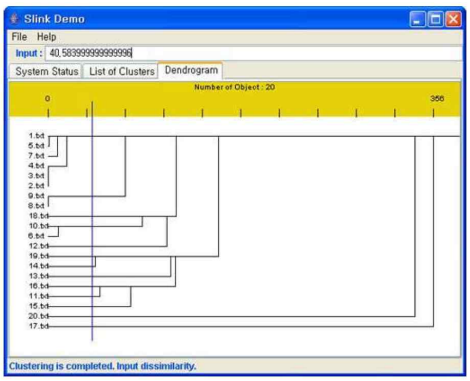
그림 3(a)와 그림 3(b)에서 각각 제안 기법과 기존 기법[3]의 가시화 결과를 비교하였다. 그림 3(b)는 임계값 40을 적용한 수형도[3]이고, 유클리디안 거리 값의 범위가 0부터 356 까지에 대하여 거리 값이 356에 가까울수록 표절하지 않은 프로그램 코드이고, 0에 가까울수록 표절의 가능성이 높아진다. 제안 기법의 가시화 결과에 비하여 기존 기법[3]의 가시화 결과는 프로그램 코드들 사이의 직관적인 관계 파악이 어렵고, 관계들 사이의 거리에 따른 정성적인 정도 비교도 어렵다.

이러한 기존 기법에 비하여 그림 3(a)의 컨셉 격자는 본문에서 제시한 세 가지 방법으로 직관적으로 프로그램 코드들 사이의 관계를 파악할 수 있다. 프로그램 코드 G, H와 프로그램 코드 I, J가 속한 컨셉은 하나의 간선을 통하여 직접 연결되어 있다. 반면 프로그램 코드 G, H와 프로그램 코드 A, B가 속한 컨셉은 두 개의 간선을 거쳐 연결되어 있다. 즉, 프로그램 코드 G, H와 프로그램 코드 I, J의 유사도에는 한 개의 공통 패턴 const int의 유무에 따른 차이가 반영된 반면, 프로그램 코드 G, H와 프로그램 코드 A, B의 유사도에는 두 개의 공통 패턴 const int, for if this의 유무에 따른 차이가 반영되어 있다. 따라서 프로그램 코드 G, H와 프로그램 코드 I, J 간의 유사도는 프로그램 코드 G, H와 프로그램 코드 A, B 간의 유사도보다 높을 가능성이 크다. 프로그램 코드 G, H는 하나의 컨셉에 속한다. 즉, 프로그램 코드 G, H는 완전히 동일한 공통 패턴을 가지며, 그 빈도에서만 차이가 있을 수 있다. 단순한 빈도의 차이는 유사도에 미치는 영향이 작은 편이므로 프로그램 코드 G, H는 유사도가 매우 높을 가능성이 크다. 따라서 프로그램 코드 G, H는 표절로 의심할 수 있다. 프로그램 코드 G, H와 프로그램 코드 A, B는 각각 하나의 컨셉에 속한다. 그러나 프로그램 코드 G, H는 프로그램 코드 A, B에 비하여 컨셉 격자 상에서 아래 쪽에 위치한다. 즉, 프로그램 코드 G, H 사이에는 프로그

램 코드 A, B 사이보다 공유하는 공통 패턴이 많다. 따라서 프로그램 코드 G, H의 유사도가 프로그램 코드 A, B의 유사도보다 높을 가능성이 매우 크다.



(a)



(b)

그림 3. 제안 기법과 기존 기법[3]의 가시화 결과 비교. (a) 컨텍스트를 이용하여 생성한 컨셉 격자. (b) 임계값 40을 적용한 수형도[3].

5. 결 론

본 논문에서는 프로그래밍 언어의 지정자와 키워드 패턴을 사용하여 프로그램 코드들 간의 유사성을 측정하고, 이 측정 결과를 정형적 개념 분석을 이용하여 해석하고 가시화하는 방법론을 제시하였다. 실험 결과 제안 기법은 96%의 표절 탐지 성공률을 보여주었고, 기존 기법[3]의 83%의 표절 탐지 성공률에 비하여 큰 향상이 있었다. 제안 기법은 기존의 유사도 측정 기법에서는 고려하지 않았던 단어 인접성을 유사도 측정에 반영하였다. 또한, 함수 단위로 지정자와 키워드 패턴을 이용하여 함수의 호출 순서나 수행 순서에 상관없이 표절을 탐지할 수 있다. 공통으로 사용된 지정자와 키워드 패턴을 이용하여 프로그램 코드를 컨셉 격자로 가시화함으로써 프로그램 코드 간 관계에 대한 이해도를 높일 수 있었다. 향후 연구로는 컨셉 격자에서 컨셉 간의 거리에 프로그램 코드 사이의 유사도를 반영하는 연구를 계획 중이다

참 고 문 헌

[1] A. Barabasi, R. Albert, and H. Jeong, “Scale-free Characteristics of Random Networks: the Topology of the World-wide Web,” *Physica*, Vol. 281, No. 1, pp. 69-77, 2000.

[2] 김영철, 최재영, “구문트리에서 키워드 추출을 이용한 프로그램 유사도 평가,” 정보처리학회 논문지, 제12권, 제2호, pp. 109-116, 2005.

[3] 손기락, 문승미, “계층적 군집화 기법을 이용한 소스 코드 표절 검사,” 정보교육학회논문지, 제11권, 제1호, pp. 91-98, 2007.

[4] 김은혜, 이송아, 허준, 한경숙, 오용철, “자바소스코드 유사도 측정 시스템,” 한국정보과학회 학술발표논문집, 제34권, 제2호, pp. 536-539, 2007.

[5] D. Grune and M. Huntjens, “Het Detecteren van Kopieën bij Informatica-practica,” *Informatie*, Vol. 31, No. 11, pp. 864-867, 1989.

[6] 한소정, 용환승, “오픈 소스코드 표절 탐지 기법,” 한국정보처리학회 추계학술발표대회 논문집, 제15권, 제2호, pp. 1459-1461, 2008.

[7] 한소정, 오픈 소스코드 표절 탐지 기법, 이화여자대학교 석사논문, 2009.

[8] A. Si, H.V. Leong, and R.W.H. Lau, “CHECK: a Document Plagiarism Detection System,” *Proc. the 1997 ACM Symposium on Applied Computing*, pp. 70-77, 1997.

[9] 손정우, 박성배, 이상조, 박세영, “Parse tree kernel을 이용한 소스코드 표절 검출,” 한국컴퓨터종합학술대회 논문지, 제33권, 제1호, pp. 157- 159, 2006.

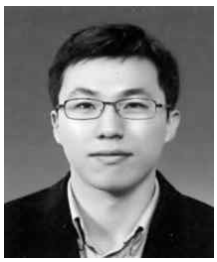
[10] K.M. Hammouda and M.S. Kamel, “Efficient Phrase-Based Document Indexing for Web Document Clustering,” *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 10, pp. 1279-1296, 2004.

[11] 이정진, 이호, 김정곤, 이창경, 신영길, 이윤철,

이민선, “동적 MR 영상에서 비강체 정합과 감산 기법을 이용한 자동 전립선 분할 기법,” 멀티미디어학회논문지, 제14권, 제3호, pp. 348-355, 2011.

[12] P. Boucher-Ryan and D. Bridge, “Collaborative Recommending using Formal Concept Analysis,” *Research and Development in Intelligent Systems XXII*, Vol. 19, No. 1, pp. 205-218, 2006.

[13] S.A. Yevtushenko, “System of Data Analysis Concept Explorer,” *Proc. the 7th national conference on Artificial Intelligence KII-2000*, p. 127-134, 2000.



이 영 주

2005년 2월 서울대학교 산업공학과 학사
2007년 2월 서울대학교 산업공학과 석사
2010년 2월 서울대학교 산업공학과 박사
2010년 3월～현재 삼성전자 생산기술연구소 책임연구원
관심분야 : 패턴인식, 영상처리, 컴퓨터 보조 진단, 의료영상, 고성능 컴퓨팅

이 정 진

2000년 2월 서울대학교 기계항공공학부 학사
2002년 2월 서울대학교 컴퓨터공학부 석사
2005년 3월 New York Institute of Technology 경영학 석사
2008년 8월 서울대학교 컴퓨터공학부 박사
2007년 10월～2009년 2월 울산대학교 의과대학 영상의학과 연구교수
2008년 1월～2010년 5월 (주)클리니컬 이미징 솔루션 기술이사
2009년 3월～2013년 2월 가톨릭대학교 디지털미디어학부 조교수
2013년 3월～현재 숭실대학교 컴퓨터학부 조교수
관심분야 : 컴퓨터 그래픽스, 변형체 모델링, 3차원 가상내시경 및 가상 수술