계층적 군집화 기법을 이용한 소스 코드 표절 검사

손기락*, 문승미
한국외국어대학교 컴퓨터및정보통신공학부, 한국외국어대학교 교육대학원
컴퓨터교육전공

요 약

인터넷 통신의 발달 및 워드프로세서의 기능 향상으로 인해 일선 교육현장에서의 표절은 심각한 문제가 되고 있다. 본 연구에서는 C, C++, Java 등으로 작성된 프로그램 소스 코드들의 유사도를 측정하는 방법을 제시하고, 소스 코드를 계층적으로 군집화하고 표절 결과를 수형도로 시각화하는 방법을 제시한다. 채점자는 시각화된 수형도를 보고 임계값을 설정하여 표절 그룹을 분리할 수 있다. 실제 데이터에서 효과를 알아보기 위해서 학부 1학년생 컴퓨터 개론 및 실습 과목 강의 중에 제출된 과제물 프로그램을 이용하여 실험해 보았으며, 유용하고 현실성 있는 방법임을 확인하였다.

Hierarchical Clustering Methodology for Source Code Plagiarism Detection

Kirack Sohn, Seungmi Moon
Hankuk University of Foreign Studies, School of Computer and Information Communication Engineering, Hankuk University of Foreign Studies, College of Education, Dept. of Computer Education

ABSTRACT

Plagiarism is a serious problem in school education due to current technologies such as the internet and word processors. This paper presents how to detect source code plagiarism using similarity based on string comparison methods. The main contribution is to use hierarchical agglomerative clustering technique to classify plagiarism groups, which are then visualized as a dendrogram. Graders can set an empirical threshold to the dendrogram to navigate plagiarism groups. We evaluated the performance of the presented method with a real world data. The result showed the usefulness and applicability of this method.

* 본 논문은 2006년도 한국외국어대학교 학술연구비 지원에 의하여 연구되었음

1. 서론

미국 룻거스 대학의 도날드 맥케이브 교수의 1993년 조사 결과에서 1960년대 4명당 1명 꼴이던 시험에서의 부정행위가 지난 1993년의 조사에서 배로 증가했다"고 말하면서 분명히 시험 부정행위가 증가하고 있음을 주장했다. 또한, 명예 관련 규약이 없는 대학에서는 53%의 학생들이 1년 동안 시험에서 한 번 이상 부정행위를 저지른 경험이 있다고 시인했고 3명 중 2명의 학생이 다른 사람의 페이퍼를 베껴낸 적이 있다고 말한 반면 명예 규약을 가지고 엄격히 실시하고 있는 학교에서는 29%의 학생들이 시험 부정행위를, 그리고 42%의 학생이 페이퍼 베끼기를 경험한 바 있다고 고백했다고 한다. 이를 계속적으로 방치한다면 무분별하고 비양심적인 표절은 관행처럼 계속 될 것이다. 이것은 개인적 표절을 떠나 윤리적 문제가 될 뿐만 아니라 대학생들의 창의력과 사고력을 떨어뜨리고 대학의 연구발전 기능을 감소시킨다는 점에서 문제가 되고 있다[3].

실제로 프로그램 교육 수업의 경우 학생들이 제출하는 프로그램 과제의 표절 여부를 일일이 육안으로 검사하기가 어렵기 때문에 이를 자동으로 탐지해주는 시스템이 필요하다. 표절 검사는 기본적으로 모든 문서 쌍의 표절 여부를 검색해야 하기 때문에 여러 연구를 통하여 빠른 수행속도와 높은 정확도를 가진 여러 표절 탐색 시스템이 설계되었으나 채점자의 요구와 의도는 고려되지 않은 채로 군집(Cluster)의 정해진 수준 및 범위에 따라 측정값이 나오고 표절 여부가 판단되는 문제점이 있었다.

따라서, 본 연구주제는 기계적인 방법으로 프로그램 과제 사이의 표절여부를 판정해주는 것이 아니라 채점자의 요구 및 의도에 따라 군집의 수준 및 범위는 달라 질 수 있으며, 그에 따라 정확도 역시 달라 질 수 있음을 확인하고 사용자가 임계값을 주어 최종적으로 판단하도록 고안하였다.

본 논문에서는 이러한 문제를 해결하기 위해 가장 일반적이면서 점진적인 특성을 반영하는 알고리즘인 SLINK[6]를 이용하여 문서들을 계층적 군집화(Hierarchical Clustering)하고 결과를 수형도(Dendrogram)로 표현하는 계층적 군집화 시스템을 설계하고 구현한다.

2. 관련 연구

2.1 프로그램 소스 코드의 표절 방법

여러 소스 코드 표절 검사 방법들을 앞에서 정의한 소스 코드 표절 방법론에 맞추어 어떻게 설계한 것인지 살펴보고, 각 소스 코드 표절 검사 방법의 특성과 장단점을 비교할 필요가 있다[1][4].

2.1.1 지문 이용 방법

일반적인 문서에서 표절을 확인하는 방법은 두 개의 문서에서 사용된 단어들의 유사성을 살피거나 사용된 단어의 빈도수 등을 비교하는 지문법을 들 수 있다. 문서의 표절 검사에서 이런 통계학적인 방법을 사용하는 이유는 문서 길이에 영향을 받지 않고, 가장 쉽게 문서의 지문(Fingerprint)을 얻어낼 수 있기 때문이다. 그러나 이 방법은 문서나 프로그램 소스 코드의 구조 파악이 어렵다는 단점을 가진다. 예를 들어 문서에서 일부분을 추가로 삽입한 경우에는 그 부분의 표절 여부 파악이 어렵다. 그리고 채점자가 전체 데이터 집합의 유사 구조를 볼 수 없기 때문에 사용자 제어가 어렵다.

2.1.2 구조 기반 검사 방법

문서와는 달리 프로그램의 소스는 제어 흐름의 변경이 어려우며, 프로그래밍 언어의 문법이 정해져

있기 때문에 구조적인 특성이 잘 나타난다. 따라서, 구조적인 분석 방법은 문서의 표절 검사보다 제어흐름을 가지고 있는 프로그램 소스 코드의 표절 검사에 많이 사용된다. 소스 코드에서 토큰을 추출하여 나열하면 선형 구조를 이루는 서열로 만들고 이들 서열들 간의 비교를 스트링 매칭 기법으로 접근한다. 이 방법은 부분적인 표절 탐지가 가능하나 길이에 따라 얻어내는 정보가 비례한다는 단점이 있다.

3. SLINK를 이용한 표절 검사

3.1 유사도

복잡한 자료 집합에서 비슷한 정도에 따른 군집 구조를 만들어 내기 위해서는 “근접성(Closeness)” 또는 “유사성” 계수가 필요하다. 유사도는 군집화 결과의 정확도에 직접적으로 큰 영향을 미친다. 따라서, 좋은 군집화 결과를 얻기 위하여 잘 정의된 유사도 계수(Similarity Coefficient)를 선택하는 것이 중요하다.

프로그램 소스 코드의 표절 유형을 분석하면 다음과 같다.

1) 원본 소스 그대로 제출하는 경우
2) 주석문을 삽입, 삭제, 편집하는 경우
3) 변수 또는 상수 이름이나 형식을 바꾸는 경우
4) 정의된 함수의 위치를 바꾸는 경우
5) 부분적으로 소스 코드를 고쳐 쓰는 경우
6) 실제로 사용되지 않는 코드를 끼워 넣는 경우

위의 6가지 방법 중 앞의 5가지 방법이 일반적인 표절 과정에서 사용되는 경우이고, 방법 6)은 다소 지능적인 방법으로 소스 코드의 흐름 분석(Flow analysis)을 통해 “죽은 코드”(dead code)를 찾아내는 복잡한 과정이 필요하다. 본 연구에서는 소스 코드 표절 정도를 계층적 군집화를 통해 시각화하고 채점자가 표절 정도를 판별할 수 있는 환경을 제공하는데 주목적이 있으므로 6)의 경우는 제외하였다.

주석은 프로그램의 실행에 영향을 미치지 않으므로 학생이 임의로 주석을 추가할 수 있다. 따라서 소스 코드에서 주석을 제거한다. 주석을 제거한 다음 소스 코드로부터 아래와 같이 특징 추출(Feature extraction)을 한다.

f_1 = 문자 개수
f_2 = 단어 개수
f_3 = 키워드 개수
f_4 = 식별자 개수

특징 f_1, f_2, f_3, f_4는 위의 1)부터 5)까지의 표절 방법을 유사한 것으로 판별한다. 각 소스 코드 객체 F는 4차원 공간상의 점 (f_1, f_2, f_3, f_4)으로 추상화된다. 두 소스 코드 객체 사이의 거리는 소스 코드에 해당하는 점들 사이의 유클리드 거리를 사용하였다. 일반적으로 정보검색에서 사용되는 코사인 계수는 두 점 벡터의 코사인 값을 측정한 것으로 문서의 크기에 상관없이 벡터의 방향성이 일치하면 유사한 문서로 판단한다. 표절 검사의 경우, 비교하는 집단인 프로그램 소스 코드의 길이는 비슷하다. 따라서 유클리드 거리가 일반적으로 문서 정보 검색의 유사도 판별을 위해 사용되는 코사인 계수보다 더 정확한 성능을 나타낼 것으로 판단하였다.

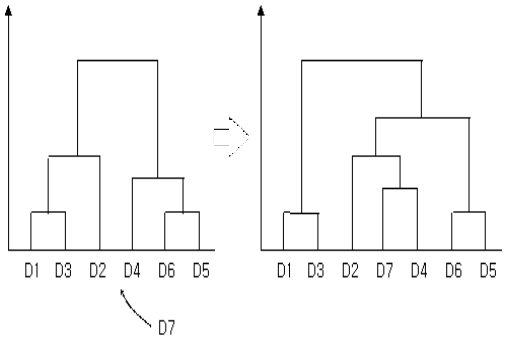
유클리드 거리 $r_{euclidean}$은 두 점 X, Y 사이의 기하학적 거리를 나타내며, 거리 값이 작게 나올수록 유사한 정도가 높다. 유클리드 거리 공식은 다음과 같다.

$X = (f_1, f_2, f_3, f_4)$, $Y = (f'_1, f'_2, f'_3, f'_4)$

$$r_{euclidean} = \sqrt{\sum (X - Y)^2}$$

3.2 SLINK를 이용한 계층적 군집화

SLINK의 가장 큰 특성은 점진성이다. 점진성은 <그림 1>에서와 같이 하나의 데이터가 새로 추가되었을 때 기존의 군집 구조에서 필요한 부분만을 갱신하여 새로운 군집 구조를 만든다[2][6].



<그림 1> SLINK의 점진적 특성

SLINK의 또 한 가지 특성은 군집 구조를 표현하는데 있어서 포인터 표현의 인덱스를 이용한다. 군집화 과정에서 데이터들을 서로 비교하고 재귀적인 방법으로 갱신하는 과정이 모두 두 개의 배열로 된 인덱스에서 처리한다. 따라서 문서의 공간적인 성능은 O(N)으로 메모리상에서의 부담을 줄일 수 있다. 또한, 수형도의 군집 결과를 저장하고 출력하고 관리하는 데 있어서 편리함을 얻을 수 있다.

SLINK 알고리즘은 세 개의 배열을 사용하는데 Π는 클러스터에 포함된 문서의 가장 큰 Index를 갖고, Λ는 연결된 문서 또는 군집간의 최소 유사도 계수를 나타낸다. N개의 문서로 구성된 군집에서 Π, Λ는 λ_N과 π_N을 포함하며 하나의 새로운 문서, n+1이 추가 된다.

이때 SLINK 알고리즘은 Π, Λ와 n+1번째의 데이터 객체와 n 개의 데이터 객체간의 유사도 계수를 갖는 일차원 배열 M을 재귀 연산을 통해 새로운 군집 구조로 갱신한다.

아래의 SLINK 알고리즘에서 첫 번째, 두 번째 과정은 Π, Λ와 M을 초기화 한다. 세 번째 과정은 새로운 데이터와 가장 가까운 데이터를 찾은 후 재귀적인 방법으로 Λ을 갱신한다. 네 번째 과정에서 전체 Π을 갱신한다. SLINK 알고리즘의 공간적인 성능은 유사도 계수를 얻는데 필요한 O(n)이며, 계층적 군집화 문제가 갖는 최소 시간인 $O(n^2)$을 만족한다.

```
1. Set Π(n + 1) to n + 1, Λ(n + 1) to ∞
2. Set M(i) to d(i, n + 1) for i = 1, … , n
3. For i increasing from 1 to n
     if Λ(i) ≥ M(i)
       set M(Π(i)) to min {M(Π(i)), Λ(i)}
       set Λ(i) to M(i)
       set Π(i) to n + 1
     if Λ(i) < M(i)
       set M(Π(i)) to min {M(Π(i)), M(i)}
4. For i increasing from 1 to n
     if Λ(i) ≥ Λ(M(i))
       set Π(i) to n + 1
```

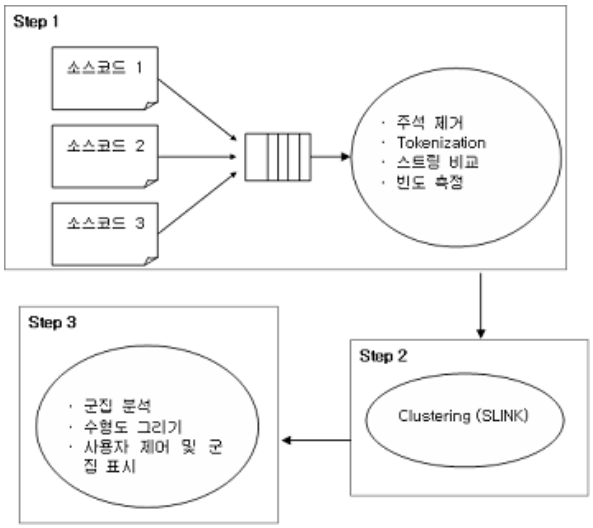
군집화 결과에서 오름차순으로 정렬한 후 조건에 맞도록 객체를 정렬하여 x축에 나열한다. 각 데이터의 λ만큼 수직으로 선을 그린 다음 수직선과 만날 때까지 수평으로 선을 그린다. 이런 과정으로 얻은 <그림 3>과 같은 수형도에서 임의의 수직선(군집화 임계값)을 통해 군집 결과를 조정하여 볼 수 있다. SLINK 알고리즘은 자바로 구현되었으며 핵심적인 코드는 <그림 2>와 같다.

```
public class Slink  {
 public Slink(List q, List l, List p) {
  ObjectList = q;
  lambda = l;
  phi = p;
 }

 public void clustering(DataObject newObj) {
  DataObject oldObj;
  List matrix = new ArrayList();
  int idx = newObj.getID();

  phi.add(idx, new Integer(idx));
  lambda.add(idx, new Double(1000.0));
  if (idx > 0) {
   for (ListIterator i = ObjectList.listIterator() ;
i.hasNext(); ) {
    oldObj = (DataObject) i.next();
    matrix.add(new
Double(DataObject.dissimilarity(oldObj, newObj)));
    }

    for (int i = 0 ; i < ObjectList.size(); i++) {
     Integer pi = (Integer) phi.get(i);
     Double li = (Double) lambda.get(i);
     Double mi = (Double) matrix.get(i);
     Double mpi = (Double) matrix.get(pi.intValue());
     if (li.compareTo(mi) >= 0) {
      matrix.set(pi.intValue(), mpi.compareTo(li) > 0 ? li
: mpi );
      lambda.set(i, mi);
      phi.set(i, new Integer(idx));
     } else
       matrix.set(pi.intValue(), mpi.compareTo(mi) > 0 ?
mi : mpi );
     }
     for (int i = 0 ; i < ObjectList.size(); i++) {
      Integer pi = (Integer) phi.get(i);
      Double li = (Double) lambda.get(i);

      Double lpi = (Double) lambda.get(pi.intValue());
      if (li.compareTo(lpi) >= 0)
       phi.set(i, new Integer(idx));
      }
    } else return;
   }

 private List ObjectList;
 private List lambda;
 private List phi;
}
```

<그림 2> SLINK 알고리즘의 자바 구현

3.3 소스 코드 표절 검사 구현

전체 시스템은 <그림 3>과 같이 세 단계에 걸쳐 수행되며, Java Platform에서 구현되었다. 각 단계에서의 수행하는 작업들은 각각 병렬적으로 수행된다.



<그림 3> 소스 코드 군집화 시스템 구조

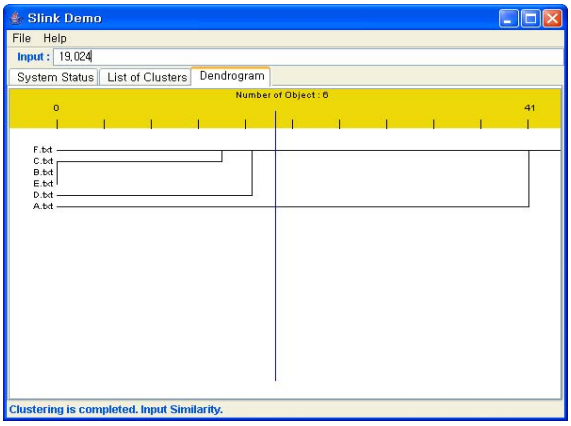
3.3.1 Step 1

첫 번째 단계는 전처리 단계로서 각 문서로부터 군집화에 필요한 과정들을 수행한다. 소스 코드 표절 검사를 위한 자료는 텍스트 파일이다. 따라서, 첫 단계에서는 주석을 제거하고 필요한 데이터 부분을 추출한다. 추출한 소스 코드로부터 StringTokenizer를 이용하여 String을 비교, 소스 코드간 유사도를 결정하는 문자, 단어, 키워드, 식별자 개수를 산출한다.

3.3.2 Step 2

Step 2 에서는 유클리드 거리를 토대로 유사도 계수를 측정하여 해쉬 테이블에 저장하고 SLINK 알고리즘을 이용하여 새로운 군집화 구조를 만든다.

군집화의 결과는 앞에서 기술한 포인터 표현인 Π, Λ 두 개의 객체이다. 두 개의 객체는 수형도의 정보를 가지고 있으며, 군집화가 완료된 후 Step



<그림 4> 수형도

3에서 결과의 출력을 위한 작업들을 수행한다.

3.3.3 Step 3

문서의 전처리 과정과 군집화가 완료된 후, 군집화 결과로부터 <그림 4>와 같은 수형도를 그린다. <그림 4>는 소스 코드 파일 6개를 군집화한 후 결과를 수형도로 그린 것이다.

<그림 4>에서 수직으로 내린 선으로 군집의 개수 및 유사도를 조절하여 채점자가 원하는 수준의 군집을 보여준다. 이때, 채점자가 원하는 수준에서 군집 구조를 볼 수 있다. <그림 4>에서는 수직선의 지점에서 수형도가 두 개의 군집으로 분리되는 것을 볼 수 있다. B, C, D, E, F를 하나의 표절 군집으로 간주된다. 군집의 크기가 1인 군집은 표절이 아닌 것으로 간주된다. 임계값을 10으로 낮추면 B, C, E를 표절 군집으로 간주하게 된다. 따라서 채점자가 프로그램 과제의 유형 또는 난이도에 따라 수형도에서 임계치를 제어할 수 있다.

3.4 소스 코드 표절 검사 실험

실험 데이터는 프로그래밍 시간에 제어문을 공부할 때 많이 보게 되는 합계를 구하는 자바 프로그램으로 SLINK 알고리즘을 이용한 소스 코드 표절 검사 실험을 하였다. 프로그램 소스 파일 6개의 파일로 실험을 하였는데, 원본 소스 파일 두 개와 그 중 하나의 소스 파일의 주석 변경, 변수 추가, 위치 변경, 변수명 수정을 해서 표절해서 만든 소스 파일 4개이다. 각 소스 코드에 대한 표절 방법은 <표 1>에 제시되어 있다.

<표 1> 소스 코드 표절 실험 예제 데이터

	A	B	C	D	E	F
표절 방법	원본 1	원본 2	원본 2의 주석 변경	원본 2의 변수 추가	원본 2의 위치 변경	원본 2의 변수 수정

<그림 4>는 프로그램 소스 예제 파일 6개를 가지고 소스 코드 표절에 관한 실험의 군집화의 결과를 시각적으로 그린 수형도이다. <그림 3>을 보면 단순한 기법의 표절인 주석 변경, 위치 변경을 한 소스 파일 같은 경우는 거의 같은 유사도를 보였고, 변수 수정 한 소스 코드 역시 높은 유사도를 보여 표절한 것임을 알 수 있다.

프로그램 과제 검사 시에 모두 다른 소스 코드가 나오는 것은 아니다. 예를 들어 '구구단 프로그램'과 같이 간단한 프로그램 과제의 경우 거의 대부분 유사한 소스 코드가 나오게 된다. 복잡하고 어려운 프로그램의 경우에는 프로그램 결과물을 도출하기 위해 사람마다 생각하는 방향이 다르기 때문에 표절이 아니고서는 같은 소스 코드가 거의 만들어지지 않기 때문에 유사도가 확연히 다르게 나타난다. 따라서 표절한 소스 코드와 표절하지 않은 소스 코드를 수형도를 통해 시각적으로 확인하는 것이 중요하다.

4. 소스 코드 표절 검사 평가

4.1 평가 데이터

소스 코드 표절 검사 프로그램의 성능을 평가하기 위하여, 실험 데이터로 학부 1학년생 컴퓨터 개론 및 실습 과목 강의 중에 학생들이 제출한 '두 원을 입력하도록 하는 그래픽 프로그램'과제를 수집하였다. 제출자는 46명이고, 프로그램 작성 언어는 C++언어이다.

[표 2] 소스 코드 표절 실험 실제 데이터 그룹

	표절그룹	비표절그룹
실험 데이터 수	10	10

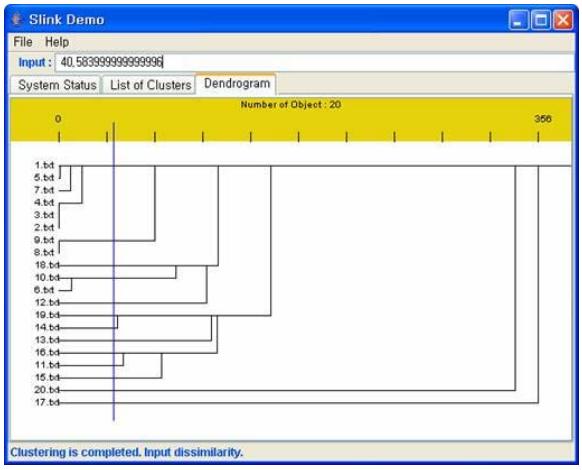
학생 46명의 과제 중 프로그램 표절이 적절히 이루어진 과제 20개를 선정하여, 본 논문에서 제시한 방법인 SLINK를 이용한 스트링 비교 방법을 적용하여 유사성을 검사하였다.

본 논문에서 제시한 방법을 실제 데이터를 가지고 소스 코드의 표절 검사에 있어서 어느 정도의 측정 능력이 있는지 확인하고 또한, 수형도에서 임의의 수직선을 이용하여 이동하며 결과를 조정하면서 검사자가 원하는 수준에서 정확한 결과를 볼 수 있는지 확인해 보았다.

4.2 평가 결과의 표현

소스 코드 파일 20 개를 받아서 군집화를 완료한 표절 검사 실행 결과 나타난 수형도는 <그림 5>와 같이 나타난다. 적은 양의 실험 데이터로 실험하였지만 계층적 응집 군집화 기법을 이용한 스트링 비교 방법이 유사성 측정에 충분히 의미가 있음은 확인 할 수 있다.

본 실험의 거리 값의 범위는 0 ~ 356이다. 거리값이 356에 가까울수록 표절하지 않은 소스 코드이고, 0에 가까울수록 유사도가 높으며, 거리값이 0인 것은 변수 하나 바꾸지 않고 그대로 제출한 표절 소스 코드임을 알 수 있다. 과제의 난이도와 전체적인 구조를 확인하고, 수형도에서 임의의 수평선을 이용하여 내비게이션하며 결과를 조정하여 보았다. <표 3>과 같이 임계값 40 부근에서 모든 표절 군집들을 찾을 수 있었다.



<그림 5> 임계값을 적용한 수형도

<표 3> 임계값 40에서 찾은 표절 그룹

	실제 표절 그룹	찾은 표절 그룹
표절 그룹 1	1,2,3,4,5,7	1,2,3,4,5,7
표절 그룹 2	8,9	8,9
표절그룹 3	6,10	6,10
비 표절 그룹	11 ~ 20	11 ~ 20

5. 결론 및 향후 과제

본 논문에서는 소스 코드 특성을 고려한 SLINK 알고리즘을 적용하여 군집화한 것을 수형도로 나타냄으로써 채점자가 보다 손쉽게 프로그램간의 표절 여부를 검사하는 시스템을 제안하였다.

그리고 이 시스템을 사용하여 학부 수업 중에 제출된 프로그램들의 표절 여부를 검사해 보았다.

학생이 제출한 과제물의 수가 많거나, 길이가 긴 소스 코드일 때 수동적인 검사로는 어려운 점이 있는 표절 문제를 본 논문에서 제시한 시스템으로 시각화함으로써 채점자의 제어가 용이하였다. 표절이 확정된 소스 코드 외에 표절 소스 코드로 강하게 의심되는 소스 코드에 대해서는 사람의 최종 확인을 거쳐야 할 필요가 있다.

이 시스템을 사용하여 교육 현장에서 프로그램 과제의 표절 여부를 효과적으로 검사하는 것이 가능하며 학생들이 프로그램을 스스로 작성하는 것을 유도하므로 교육적 효과를 얻을 수 있다.

보다 정확한 표절 검사를 하기 위해서 소스 코드의 제어 흐름을 파악하는 방법이 필요하다. 표절자가 프로그램의 실행과 무관한 코드를 삽입하는 경우 본 논문에서 제시한 유사도 계산 방법으로는 표절 여부를 파악하는 것이 어렵다. 따라서 프로그램의 실행 흐름을 파악하여 전처리 과정에서 실행과 무관한 코드를 제거하여야 보다 정확한 표절 검사를 할 수 있다. 향후 이 문제에 대한 연구가 필요하다.

참고문헌

1] 전명재, 이평준, 조환규 (2004), 분할점령 방식을 이용한 지역정렬과 이를 이용한 소스코드 표절 탐색 기법, 한국정보과학회.

2] M. Blosseville, G. Hebrail, M. Monteil, N. Penot. (1992), Automatic document classification: natural language processing, statistical analysis, and expert system techniques used together, SIGIR'92.

3] Edlund, John R., What is plagiarism and why do people do it?, Writing Center of Calfornia State University, 1988.

4] Faidhi, J. A. and S. K. Robinson (1987), An empirical approach for detecting program similarity and plagiarism within a university programming environment, Computing in Education, Vol. 11, 11-19.

5] Lorr M. (1983), Cluster Analysis for social Scientists: Jossey -Bass Social and Behavioral Science Series, Proquest Info & Learning.

6] R. Sibson (1973), SLINK: An optimally efficient algorithm for the single-link cluster method, Computer journal.