오픈 소스코드 표절 탐지 기법

한소정, 용환승
이화여자대학교 컴퓨터정보통신학과
e-mail : sjhan@ewhain.net, hsyong@ewha.ac.kr

Detecting Open Source-Code Plagiarism

Sojeong Han, Hwan-Seung Yong
Dept. of Computer Science & Engineering, Ewha Womans University

요 약

오픈 소스코드의 확대는 상용 프로그램에서의 활용 증대로 이어지고 있다. 컴퓨터 프로그램의 유사도 및 완성도 감정에서 오픈 소스코드의 비중이 증대됨에 따라서 오픈 소스코드의 탐지 방법이 요구된다. 본 논문에서는 프로그램 소스코드 검색 기법을 조사하고 평가하여 효과적인 탐지 기법을 제안한다.

1. 서론

소스코드 표절은 원본자료의 출처를 분명히 밝히지 않고 자신의 것처럼 사용하는 행위를 말하는데, 학생이 제출하는 과제에서부터 산업 현장의 프로그램 일부 또는 전체에 이르기까지 소스코드 표절 범위는 매우 다양하다. 이러한 표절 문제는 인터넷 매체의 발달로 인해 더욱더 만연해지고 있고, 최근 국내외 많은 기업들이 오픈 소스를 활용하여 비즈니스를 수행하고 있다. 그러나 그 오픈 소스들이 가지고 있는 라이센스를 무시한 채, 아니면 본의 아니게 라이센스들의 요구조건들을 어기고 오픈 소스를 상업적으로 이용하는 사례들이 점점 많아지고 있다. 하지만 이를 탐지하거나 방지하기 위한 대책 마련은 미흡한 실정이기 때문에 오픈 소스코드 표절을 탐지하는 연구가 필요하다.

본 논문에서는 이러한 오픈 소스코드 표절 탐지 기법에 관하여 제안한다. 논문은 다음과 같이 구성되어 있다. 2장에서는 기존의 소스코드 표절 탐지 기법에 대해 분석하고, 3장에서는 효과적인 탐지 기법을 제안, 응용 프로그램을 구현하고 실험 및 평가를 한 후 4장에서는 결론 및 추후 연구 발전 방향에 대하여 설명한다.

2. 관련연구

2.1. 소스코드 표절 탐지

2.1.1. 소스코드 표절 탐지 기법의 방법론 분석 및 비교

소스코드 표절 탐지를 위한 기법은 크게 Attribute Counting 방법과 Structure Metric 방법, 두 가지로 나눌 수 있다.[1]

1) Attribute Counting 방법

초기 소스코드 표절 탐지 기법에 사용하였고. 사용된 단어나 키워드의 유사성 및 사용 빈도를 비교한다. 이 방법은 문서의 길이에 영향을 받지 않고, 문서의 단락의 순서를 변경하여도 영향을 받지 않는다. 또한, 부분적 표절 탐지에 어려움이 있고, 문서의 검사에 유용하다.

2) Structure Metric 방법

정확한 매치가 아닌 토큰 스트링의 유사성을 계산하는 방법으로 크게 두 가지 단계로 수행된다. 첫 번째 단계에서 코드를 분석하고 토큰 스트링으로 변환한 후, 두 번째 단계에서 변환된 토큰 스트링을 비교 검사한다. 이 방법은 문서의 길이에 따라 비례하며 문서의 단락의 순서를 변경하면 영향을 받는다. 또한 부분적인 표절 탐지가 쉽고, 소스 코드 검사에 유용하다. 이러한 장점으로 Attribute Counting 방법보다 효율적인 방법으로 평가된다.

2.1.2. 소스코드 표절 탐지 툴의 종류 및 특징

기존의 소스코드 표절 탐지 툴들 중, JPlag[2], Moss[3], CodeMatch[4][5],[6]는 다음과 같다.

1) JPlag

웹 기반 시스템으로써, Java, C, C++ 와 Scheme으로 작성된 소스코드 사이의 유사성을 측정하는데 소스코드를 토큰 스트링으로 변환 후 유사성을 측정한다.

2) Moss

웹 기반 시스템으로써, 8가지 언어(Java, C, C++, Pascal, Ada, ML, Lisp, Scheme)를 지원하며 알고리즘은 다음과 같다.

첫째, 각 소스파일로부터 공백과 문단기호를 제거하고, 모든 문자들을 소문자로 변환한다. 둘째, k길이의 k-gram으로 코드를 나눈다. 셋째, 각 k-gram을 hash한다. 넷째, 소

*본 연구는 한국과학재단 목적기초연구(R-01-2006-000-10609-0) 지원으로 수행되었음

스코드의 fingerprint를 만들기 위해 hash들의 subset을 선택한다. 마지막으로 fingerprints를 비교한다.

3) CodeMatch

상용 소프트웨어로 다양한 프로그래밍 언어를 지원, 프로그램 언어에 대한 정의를 한 후 어떠한 언어라도 지원가능하다. 이 소프트웨어는 비교하고자 하는 두 폴더를 선택 후 그 폴더 안의 소스파일들을 비교한다. 표절 탐지결과는 다른 툴보다 정확하지만 실행속도가 느리다는 단점이 있고, Statement Matching, Comment Matching, Identifier Matching, Instruction sequence matching, Correlation Score와 같은 5가지 매칭 알고리즘을 사용하여 비교를 한다.

2.1.3 정리

기존의 소스코드 표절 감지 툴들은 앞에서 알아본 JPlag와 Moss와 같은 툴들은 학생들의 과제 표절 검사를 위한 단순한 툴이기 때문에 오픈소스코드 표절을 감지하는 툴로 사용하기에는 실행속도가 한없이 길거나 많은 오픈소스들 중에서 표절을 감지하기에는 부족한 면이 많다.

2.2. Google 코드 검색

2.2.1. Google 코드 검색

Google 코드 검색[8]을 이용하면 검색속도가 빠르고 인터넷에 호스팅 되는 공개 액세스 소스 코드를 한 곳에서 검색할 수 있어 기능에 대한 정의나 샘플코드를 찾을 수 있다. Google 코드 검색으로 할 수 있는 일은 좀 더 자세한 검색을 위해 정규식 사용, 언어별 라이선스 별 또는 파일 이름 별로 검색을 제한, 코드가 속한 전체 패키지 및 웹 페이지로 돌아가는 링크를 통해 소스파일 보기 등이다.

2.2.2. Google 코드 검색 데이터 API

Google 코드 검색 데이터 API는 클라이언트와 웹 어플리케이션들이 함수정의와 샘플코드를 위한 공개 소스코드를 찾는 것을 허용해준다. Google 코드 검색 API 질의 방법은 다음과 같다.

1) 질의를 매뉴얼 하게 보낸다.

코드 검색은 특정화된 정규식과 일치되는 모든 공개 코드 샘플들을 요청하게 하고, 코드 검색으로부터 결과를 얻기 위해 http://www.google.com/codesearch/feeds/search에 적절한 질의 파라미터들을 함께하여 HTTP GET 요청을 보낸다. 예를 들어, malloc 문자열을 요청하기 위해서는 다음의 HTTP 요청을 사용할 수 있다.

http://www.google.com/codesearch/feeds/search?q=malloc

다음과 같이 요청을 할 때, 코드 검색은 HTTP 200 OK 상태 코드와 검색 전체 결과들을 포함하는 결과물을 나타낸다.

2) 클라이언트 라이브러리를 이용하여 질의를 보낸다.

직접 질의를 보내는 것 대신에 클라이언트 라이브러리를 사용해 질의를 보낼 수 있다. 예를 들어, malloc 문자열을 검색할 때 (그림 1)과 같은 자바코드를 사용할 수 있다.

```
CodeSearchService myService = new CodeSearchService("exampleCo-exampleApp-1");
URL feedUrl = new URL("http://www.google.com/codesearch/feeds/search?q=malloc");
CodeSearchFeed resultFeed = myService.getFeed(feedUrl, CodeSearchFeed.class);
System.out.println("Number of Entries Received: " + resultFeed.getEntries().size());
```

(그림 1) 자바코드 사용 예

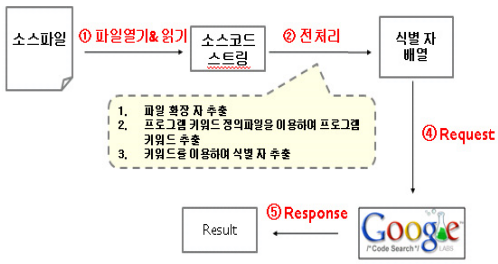
CodeSearchService 클래스는 GData서비스로 연결되는 클라이언트를 나타낸다. CodeSearchService.getFeed 메서드로 완전한 URL 질의를 넘겼을 때, 클라이언트 라이브러리는 질의와 응답과 반환된 XML을 분석하여 보낸다.

하지만 Google 코드 검색은 한 줄 단위의 Request를 원칙으로 하고 있기에 전체적인 소스코드를 검색하기에는 적합하지 않다.

3. 오픈 소스코드 표절 탐지

3.1 오픈 소스코드 표절 탐지 기법

앞 절에서 알아보았듯이, Google 코드 검색 API를 사용하면 오픈소스를 쉽고 빠르게 찾을 수 있기 때문에 본 연구에서는 Google 코드 검색 API를 이용한 효과적인 오픈 소스코드 표절 탐지 기법을 제안한다. 탐지 기법의 구조는 (그림 2)와 같다.

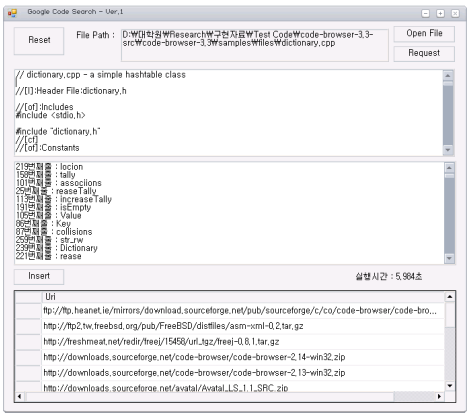


(그림 2) 오픈 소스코드 표절 탐지 기법 구조

먼저, 표절을 탐지하고 하는 소스를 선택하여 스트링형태로 저장한다, 전처리 과정에서는 각 프로그래밍 언어별로 미리 정의해놓은 프로그램 키워드 파일을 통해 키워드들을 제거하여 프로그래머가 정의한 특별 식별자들을 추출해낸다. 다음은 전처리 과정에서 추출해놓은 식별자들을 Google 코드 검색 데이터 API를 사용하여 연속으로 Request 한다. 이때 코드 검색 feed를 검색하기 위해 사용되는 URL은 "http://www.google.com/codesearch/feeds/search?q=식별자"이다. Google 코드 검색은 가장 일치하는 결과 순서대로 반환하기 때문에 Response결과의 상위 값을 실제로 비교하여 소스코드 동일여부를 검증한다.

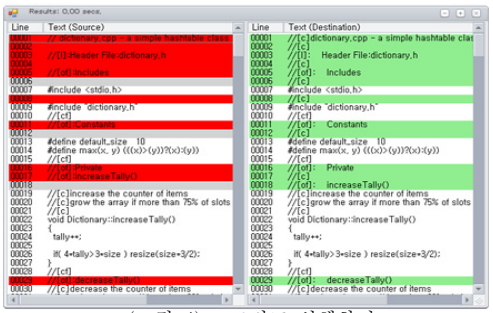
3.2. 응용 프로그램

앞 절에서 제안한 오픈 소스코드 표절 탐지 기법을 바탕으로 응용 프로그램을 구현하였다. 실행화면은 (그림 3)과 같다. 원본 소스에서 식별자를 추출하여 Request버튼을 누르면 Google 코드 검색 데이터 API를 통해 식별자가 포함된 오픈소스들의 Uri 값을 가져온다.



(그림 3) 소스코드 검색 실행화면

Google 코드 검색은 가장 일치하는 결과 순서대로 반환하기 때문에 Response결과의 최상위 Package주소 값을 통해 소스 Package를 저장하고 압축을 푼다. 검증 과정에서는 (그림 4)와 같이 원본소스와 비교한다.

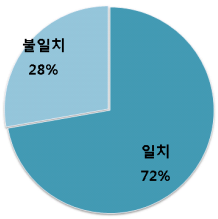


(그림 4) 코드비교 실행화면

3.3. 성능 측정 및 평가

오픈 소스 코드 프로젝트 포털 사이트인 sourceforge.net[10]에서 랜덤으로 추출한 소스코드 100개를 테스트 하였다. 본 연구에서 구현한 코드 검색 프로그램은 Google 코드 검색 데이터 API를 통해 소스 표절 여부를 탐지하기 때문에 기존의 소스코드표절 프로그램들과 성능비교는 불가능하여 본 프로그램의 성능만을 측정하였다.

그 결과는 (그림 5)과 같이 72%의 정확성을 나타냈고, 속도는 각 소스코드 당 1000줄의 기준으로 평균 10초가 걸렸다. 28%의 불일치하는 코드들은 전처리과정에서 추출한 식별자들이 특별하지 않는 식별자들로 코드를 검색하였기 때문에 다른 코드들을 결과를 나타냈다.



(그림 5) 원본 소스코드 동일여부

4. 결론

컴퓨터 프로그램의 유사도 및 완성도 감정에서 오픈 소스코드의 비중이 증대됨에 따라서 오픈 소스코드의 탐지 방법이 요구된다. 본 논문에서는 그러한 프로그램 소스코드 검색 기법을 조사하고 평가하여 효과적인 탐지 기법을 제안하였다. 테스트 결과로 약 72%의 정확성을 보였기 때문에 오픈 소스코드 표절 여부를 정확하게 탐지 하는 것은 아니지만, 전처리 과정에서 정확성을 높인다면 오픈 소스코드 표절 여부를 좀 더 효율적으로 탐지 할 수 있을 것으로 생각된다.

향후과제로는 전처리 과정에서 사전을 이용하여 특별하지 않는 식별자들을 제거하여 정확성을 높일 예정이다.

참고문헌

[1] Kristina L. Verco and Michael J. Wise, Software for Detecting Suspected Plagiarism : Comparing Structure and Attribute - Counting Systems, St. Australian Conf. on Computer Science Education, Sydney, Australia, July, 1996

[2] Lutz Precheit, Guido Malpohl & Michael Phillippsen, "JPlag : Finding Plagiarisms among a Set of Programs, " Universitat Karlsruche, Germany, Technial Report 2000-1, 2000

[3] Aiken, A, MOSS: a system for detecting software plagiarism. University of Berkeley, CA. Available at http://www.cs.berkeley.edu/~aiken/moss.html. 1998

[4] Saul Schleimer, Daniel Shawcross Wilkerson, Alexander Aiken: Winnowing: Local Algorithms for Document Fingerprinting. SIGMOD 2002, 76-85

[5] Zeidman, R., Software Source Code Correlation, icis-comsar, pp. 383-392, 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR'06), July 2006.

[6] Bob Zeidman, "Detecting Source-Code Plagiarism," Dr. Dobb's Journal, July 2004, 55-60.

[7] S.A.F.E, http://www.safe-corp.biz/products_codesuite.htm

[8] Google Code Search, http://www.google.com/codesearch

[9] Google Code Search Data API, http://code.google.com/apis/codesearch/

[10] Sourceforge.net, http://www.sourceforge.net/