자바소스코드 유사도 측정 시스템

김은혜[0], 이송아, 허준, 한경숙, 오용철
한국산업기술대학교 컴퓨터공학과
{wnsrmsu[0], olive425, juncjswo, khan, oh}@kpu.ac.kr

Java source code Similarity Measurement System

Eun-Hye Kim[0], Song-A Lee, Jun Heo, Kyung-Sook Han, Yong-Chul Oh
Department of Computer Engineering, Korea Polytechnic University

요 약

JSMS(Java source code Similarity Measurement System)는 자바 소스 코드의 유사도를 측정하고 이와 관련한 소스 코드의 정보를 시각적으로 표시하는 시스템이다. 기존의 표절 검사 시스템은 소스코드의 구조적 특징을 반영하지 못해 유사도 결과의 신뢰성이 낮고 대부분 편리성과 가독성이 좋지 않아 사용하기 불편하였다. 본 논문에서 제안하는 JSMS는 이러한 단점을 보완하기 위해 함수 선형화를 사용하여 소스코드의 구조적 특징을 반영하였다. 또한 쉽고 간단한 조작으로 편리성을 제공하며, 관련 정보와 유사 구간을 시각적으로 표시하여 가독성을 높였다. 향후 다양한 언어 지원과 폭넓은 시각적 정보 제공을 보완하여 사용자의 학습 자료로 사용할 수 있으며, 소스코드 표절의 객관적 기준이 되는 도구로 활용 가능하다.

1. 서 론

IT산업의 발전과 소프트웨어의 발달로 사회는 빠르게 지식정보기반으로 이행하고 있고, 이와 함께 정보와 지식에 대한 지적 가치의 중요성은 증대되고 있다. 컴퓨터 프로그램 저작권은 지적재산권의 한 종류로서 마땅히 보호하고 관리되어야 하지만 그 침해와 피해 사례는 늘어만 가고 있다[1]. 이에 반해 이를 해결하기 위한 근본적인 기준과 도구는 현재 매우 미흡한 실정이며, 그 인식 또한 부족한 편이다. 현재 사용할 수 있는 소수의 표절 감지 시스템은 소스코드의 구조와 특징을 고려하지 않아 높은 신뢰성을 갖기 어렵다. 또한 대부분의 시스템이 사용법이 어렵고 복잡하여 효율성과 가독성이 매우 낮다. 본 논문에서 제안하는 JSMS는 소스코드 유사도 측정 시스템으로 이와 같은 문제 해결의 토대를 제공하고 문제에 대한 인식을 제고하도록 하였다. JSMS는 소스코드의 특징을 이용한 구조적 분석으로 유사도 측정의 신뢰성을 높이고, 관련 정보를 보여줌으로써 사용자가 이해할 수 있도록 하였다. 또한 쉽고 간단한 조작과 시각적 표시를 통해 사용자의 편리성 및 가독성을 높일 수 있도록 구현하였다.

2. 관련연구

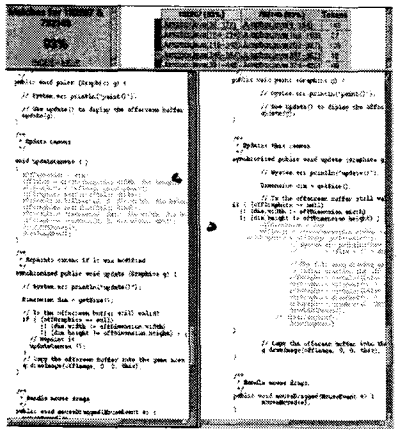
2.1 소스코드 표절의 기법과 유형

표절이란 원저자의 참조 사항 없이 무단으로 재사용하거나 기존의 것을 간단한 편집과정을 거쳐 자신의 것으로 쓰는 행위를 말한다[2]. 소스코드는 일반 문서와 다르게 구조적 변경에 큰 영향을 받으므로 소스코드의 표절은 일반적인 표절의 정의만 가지고 판단할 수 없다. 그래서 소스코드의 표절을 검사하는 방법에는 유사한 단어들이나 키워드들의 사용 횟수를 바탕으로 문서의 표절 여부를 판단하는 지문법(fingerprint) 방법과 호출과 분기의 흐름을 고려하는 구조적(structure-related) 방법이 있다[3]. 프로그램 소스코드의 표절 방법과 유형에는 다음과 같은 것이 있다. 주석이나 변수를 수정, 삭제하거나 실제로 사용하지 않는 코드를 삽입할 수 있다. 정의된 함수의 위치를 바꾸고 부분적으로 수정 할 수 있으며, 라이브러리 코드를 사용할 수 있다. 또한 함수를 분할하거나 합치는 방법을 사용할 수 있다. 소스코드의 표절 여부를 검사할 때 가장 고려되어야 할 부분은 구조적인 변형에 대한 것인데, 대부분의 기존 시스템은 이것을 적절히 반영하지 못하고 있는 문제점이 있다.

2.2 JPlag

JPlag는 기존의 소스코드 표절 검사 시스템으로 독일의 한 대학에서 연구하여 만들어진 것이다[4].Java, C, C++ 등의 프로그램 소스코드의 유사도를 검사 할 수 있으며, 자바언어가 실행되는 환경이면 어디서든 웹을 통해 무료로 이용할 수 있다. [그림 1]은 JPlag의 실행화면이다. 좌측 상단에 두 파일에 대한 유사도의 결과가 나타나 있다. 오른쪽에는 라인별로 비교하였을 때 유사한 정도에

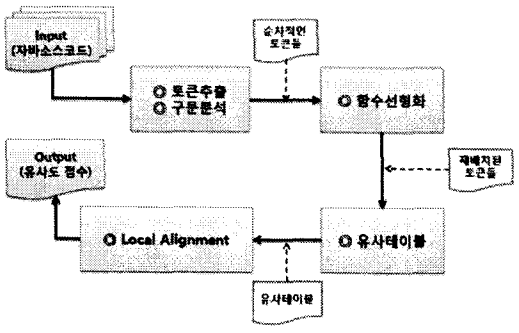
따라 색깔별로 그 수치를 보여주고, 그 아래 두 소스파일을 보여주는 View에서는 유사한 부분을 같은 색으로 표시하고 있다. JPlag는 소스코드를 토큰으로 분리해 선형 구조의 서열로 만들고 서브 스트링 매칭 방법으로 비교, 접근하는 방식을 사용한다[4]. 이러한 비교를 통한 결과는 [그림 1]처럼 시각적으로 표시되며 기존에 있는 여러 표절 검사 시스템과 비교해 볼 때 성능, 인터페이스 면에서 가장 신뢰 할 수 있는 시스템이다. 그러나 JPlag의 분석 원리는 일반적인 문서 표절을 검사하는데 더 적합한 방법으로 소스코드의 구조적 특징을 완벽히 반영하지 못한다. JPlag를 이용하여 유사도를 검사하는 실험에서 소스코드의 수정과 편집의 정도를 고려하였을 때 유사도 결과가 들쑥날쑥하고 규칙적이지 않다는 점에서도 그 사실을 알 수 있다.



[그림 1] JPlag의 실행 화면

3. 설계 및 구현

3.1 시스템 구성도



[그림 2] JSMS 시스템 구성도

[그림 2]는 논문에서 제안하는 JSMS 시스템 내부 구성도이며, 역할과 구성에 따라 크게 4 부분으로 이루어진다. 먼저 자바소스코드를 입력으로 파싱을 수행하게 되는데 토큰추출과 구문분석을 통해 순차적인 토큰들이 저장된다. 이 과정에서 클래스와 메소드, 식별자에 대한 정보가 심벌테이블로 저장되고, 이는 앞서 추출한 토큰들과 함께 함수 선형화의 과정에서 사용하게 된다. 함수 선형화는 소스코드의 구조적인 특징을 반영하기 위해 사용하는 것으로, 프로그램의 호출과 분기 흐름에 따라 토큰들을 절차적으로 재배치하는 작업이다. 함수 선형화 과정을 통해 원본 소스코드와 비교할 소스코드의 토큰들이 최종적으로 재배치되어 추출되고, 각 토큰들의 가중치 값을 이용한 계산으로 유사테이블을 작성한다. 유사테이블의 값은 Local Alignment방법을 사용하여 최종 유사도 점수를 결과로 나타내는 데에 적용된다.

3.2 토큰추출 및 구문분석의 구현

본 시스템은 특정 토큰 및 문법을 인식하여 자바 언어로 된 파서를 생성하는 JavaCC를 사용한다[5]. JavaCC는 스캐너와 파서의 역할을 동시에 할 수 있는 파서 생성기로 시스템에서 요구하는 토큰을 추출하고, 파서에서 필요한 정보를 생성하는데 매우 편리하고 적합한 도구이다. 이 과정에서는 다음의 [표 1]과 같은 키워드를 토큰으로 추출한다. 각 키워드는 구조적으로 미치는 영향에 따라 3 가지로 분류되며 각각 다른 가중치를 갖게 되는데, 이것은 유사테이블을 만들어 최종 유사도를 계산할 때 쓰이게 되는 자료이다[6]. 또한 구문 분석을 통해 클래스와 메소드, 식별자 등에 대한 정보를 심벌테이블로 만들어 저장하고, 이는 함수 선형화를 위해 참고할 정보로 활용된다.

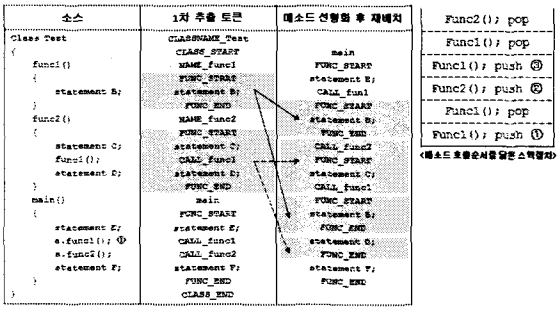
[표 1] 소스에서 추출되는 키워드의 분류

구조적 키워드	if, else, while, do, for	
비구조적 키워드	Type	int, float, double, unsigned, signed, String, char, long, boolean
	Operator	=, ==, !=, +=, -=, *=, /=, <, >, <=, >=
	Etc	switch, break, return, true, false
확장된 토큰	블록키워드	CLASS_START, CLASS_END, METHOD_START, METHOD_END
	이름키워드	NAME_METHOD_1, CALL_METHOD_1

3.3 함수 선형화의 구현

본 시스템은 소스코드의 구조를 고려하는 함수 선형화 방법을 사용하였다. 이것은 프로그램의 구조적 특징을 반영하는 기법으로, 소스코드의 호출과 분기의 흐름에 따라 재구성하여 비교하게 된다. 토큰 추출 과정을 통해 토큰이 추출되고 심벌테이블의 정보가 저장되면, 시스템은 다시 한 번 소스코드를 읽어서 분석하게 된다. 이 때 소스코드에서 함수를 색출하고, 심벌테이블에서 검색한 함수의 정보를 이용해 재배치 작업을 수행하게 된다. 함수 선형화 방법을 사용하면 소스 코드의 실제적인 수행 절차와 같이 토큰들이 재배치되기 때문에 더미코드의 삽입이나 메소드의 위치 변경, 실제 사용하지 않는 코드의 삽입 등에 효과적으로 대처 할 수 있다[3]. 이러한 유형은 소스코드의 표절을 감시하고 분석하는데 크게 작용하는 변수이지만, 기존의 시스템은 이를 잘 반영하지 못하고 있다. 본 시스템에서는 함수 선형화 기법을 사용함으로써 이러한 유형에 대처하고 해결하도록 하였다.

[그림 3]에서는 함수 호출 순서를 스택을 이용하여 함수 선형화를 하는 예를 보여주고 있다. 1차 추출 토큰에서 함수를 색출해서 재배치하는 과정을 색깔로 표시하여 설명한 그림이다. 우측에 있는 그림은 메소드가 색출되고 재배치되는 시점과 종료를 스택으로 표시하여 나타낸 것이다. 스택은 함수의 재귀호출, 함수 호출 내의 호출 등을 구분하는데 쓰는 정보이다.



[그림 3] 함수 호출 스택을 이용한 함수 선형화 방법

3.4 지역 정렬의 구현

함수 선형화 과정 후, 절차적으로 재배치된 원본 소스코드와 비교 소스코드의 토큰들은 2차원 배열의 x, y축을 구성한다. 각 토큰들의 가중치 값을 이용해 [표 2]와 같은 동적 프로그래밍 공식에 따라 일치 여부를 검사하고 유사테이블을 작성한다[7]. 이 작업이 끝나면 지역 정렬(local alignment)을 이용해 최종 유사도 결과를 산출하게 된다. 지역 정렬은 전체 정렬(global alignment)과 함께 유전공학에서 DNA의 유사성을 검사하기 위해 쓰는 방식이다. 전체 정렬은 말 그대로 전체 토큰의 유사성을, 지역 정렬은 전체에서 최대의 부분 정렬의 유사성을 비교하는 방법이다[8]. 소스코드는 부분적인 구조에 따라 유사성을 분석하고 파악해야 한다. 따라서 지역 정렬 방법을 시스템에 적용하는 것이 더 효과적이다. [표 2]는 두 소스코드에서 추출된 토큰들이 가지는 가중치에 따라 계산되는 방법을 나타낸 것이다. [표 1]에서 분류되는 키워드의 가중치를 이용해 일치, 불일치 여부를 계산하여 유사테이블을 작성하고, 이 유사테이블은 최종적으로 유사도 결과를 산출하는데 사용한다.

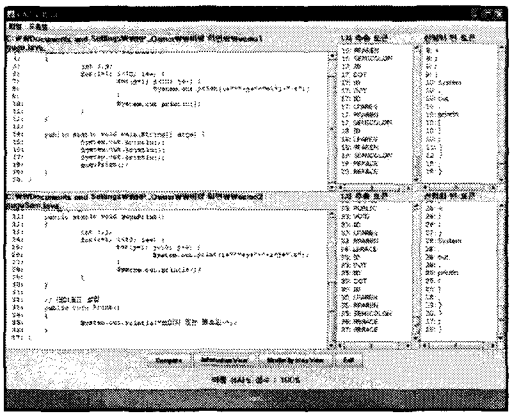
[표 2] 유사테이블을 채우기 위한 동적프로그래밍

Left_top m(x-1, y-1)	Top m(x, y-1)
Left m(x-1, y)	Maximal value m(x, y)

$$m[x,y] = \max \begin{cases} m[x-1,y-1] + score(x,y) \\ m[x,y-1] + gap \\ m[x-1,y] + gap \\ 0 \end{cases}$$

* score : x번째 키워드와 y번째 키워드의 일치여부에 따른 점수
* gap : 공백 삽입에 대한 감점점수

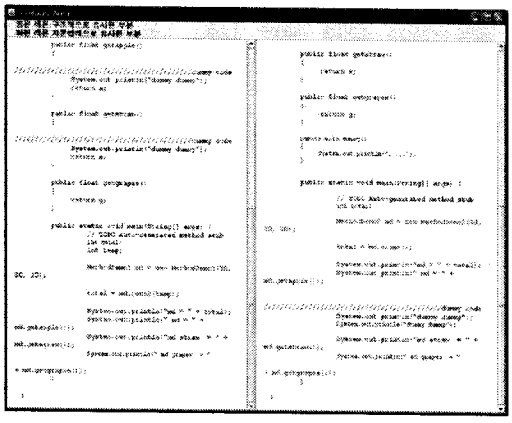
3.5 구현 결과



[그림 4] JSMS 실행 결과 화면

[그림 4, 5]는 본 시스템 JSMS로 자바소스코드 유사도를 측정한 결과이다. [그림 4]는 두 소스코드를 입력하여 최종 유사도 점수와 그와 관련된 정보를 결과로 보여주는 화면이다. 시스템 하단의 버튼은 파일을 분석하여 최종 유사도 결과를 보여주고, 관련 정보를 표시하여 사용자의 이해를 돕는다. 소스코드를 분석하고 유사도를 측정하는 과정의 일부분인 토큰 정보와 유사테이블을 볼

수 있게 하였다. [그림 5]는 소스코드의 유사 정도를 색깔로 표시한 화면이다. JSMS는 소스코드의 구조적 특징을 함수 선형화를 사용하여 반영하였는데, 그 결과로 표시된 빨간색 부분이 구조적으로 유사한 부분이다. 일반적인 문서 표절을 검사하는데 더 적합한 지문법의 검사 결과는 파란색으로 유사한 부분이 표시 된다. 모의실험에서 소스코드 표절 기법과 유형을 적용하여, 기존 시스템 중 하나인 JPlag와 본 논문에서 구현한 JSMS의 결과를 비교해 보았다. JPlag의 경우 키워드의 편집, 삭제의 수와 유사도 변화의 폭에 대해 연관성이 없고, 불규칙적인 부분이 많았다. 또한 단순히 소스 코드의 순서와 위치를 바꾸는 것에 대해서도 유사도의 변화가 큰 것을 볼 수 있었다. 이에 반해 본 논문에서 구현한 JSMS 시스템에서는 키워드의 편집, 삭제의 수에 대해 비교적 융통성 있는 유사도의 차이를 보였다. 또한 구조적으로 영향력이 없는 소스코드의 삭제나 단순히 위치와 순서를 바꾸는 수정에 대해 잘 대처하여 좋은 결과를 나타내었다. 이는 소스 코드의 구조적 특징을 적절히 반영하고, 더불어 지문법도 적용하여 보다 높은 신뢰성을 가지는 시스템을 구현한 결과이다.



[그림 5] 유사 구간의 표시

4. 결론 및 향후 연구

오늘날 프로그램 저작권의 가치는 매우 중요하게 인식되고 있지만, 그에 대한 관리와 보호의 체계는 자리 잡혀 있지 않은 실정이다. 이러한 현실을 제고할 수 있는 노력의 일환으로 본 논문에서는 소스코드의 표절을 감시할 수 있는데 적합한 시스템을 구현하였다. 기존의 시스템이 대체로 지문법만을 사용한 것과 달리 본 시스템 JSMS는 소스코드의 구조적 흐름과 구조에 미치는 영향에 따른 가중치를 반영하여 더 많은 표절 유형에 대처할 수 있도록 하였다. 특히 함수의 위치적인 구조를 조작하거나 실제로 사용하지 않는 함수를 추가하는 경우, 구조적으로 미치는 영향이 적은 코드를 수정, 삭제하는 유형에 대해 좋은 결과를 나타내었다[3]. 이는 함수 선형화와 구조적 가중치를 이용한 접근으로 소스코드의 특징을 적절히 반영하고, 지문법 방식의 검사를 중용하여 시스템 성능의 신뢰도를 증가 시킨 결과라고 할 수 있다. 또한 유사 구간을 색깔로 표시하고 관련 정보를 사용자가 볼 수 있게 하여 사용성과 가독성을 개선시킬 수 있었다.

본 시스템은 사용자가 프로그램의 원리를 이해하고 흐름을 알 수 있도록 정보를 제공하는데, 사용자가 효율적으로 사용할 수 있도록 보다 시각적으로 제공할 수 있는 방법이 필요하다. 앞으로 Java 언어 외에 C, C++ 언어 등을 지원하고, 학습 자료로 활용 할 수 있는 정보 제공의 기능을 개선한다면 좀 더 실용적이고 객관적인 시스템으로 거듭날 수 있을 것이다.

5. 참고 문헌

[1] 김우정, "프로그램 표절 감정 기법에 관한 연구", 홍익대학교 정보대학원 학위논문(석사), 2003. 02

[2] http://www.calstatela.edu/centers/write_cn/plagiarism.htm

[3] 황미녕, 강은미, 한기덕, "유전체 서열의 정렬 기법을 이용한 소스 코드 표절 검사", 한국정보과학회 논문지 C, Vol, 09NO, 03pp, 0352~0367, 2003. 06

[4] JPlag, http://www.ipd.uni-karlsruhe.de/jplag/

[5] 전명재, "JavaCC를 이용한 구문 분석", 부산대학교 그래픽스 응용 연구실, Technical Report, 2003

[6] 전명재, "표절 검사를 위한 키워들 간 상호 유사 테이블 구성", 부산대학교 그래픽스 응용 연구실, Technical Report, 2003

[7] 전명재, "소스 표절 검사를 위한 Divide And Conquer 방식의 Local Alignment", 부산대학교 그래픽스 응용 연구실, Technical Report, 2003

[8] 조환규, http://pearl.cs.pusan.ac.kr/publication/Genomic Alignment.ppt, 부산대학교 그래픽스 응용 연구실