

재귀적 지역정렬을 이용한 프로그램 표절 탐색

전명재[○], 이평준, 조환규
 {mjjun[○], pilee, adagio}@pearl.cs.pusan.ac.kr

Source code Plagiarism Detection with Recursive Local Alignments

Myung-Jea Jun[○], Pyung-Jun Lee, Hwan-Gue Cho
 Dept. of Computer Science and Engineering, PUSAN National University

요 약

지역정렬(local alignment)과 전체정렬(global alignment)로 대표되는 정렬 문제는 전산학 분야의 전형적인 문제로, 두 서열의 전체적인 또는 부분적인 유사성(similarity)을 찾아 주기 위한 방법이다. 특히 정렬은 두 문자열에서 유사하게 나타나는 유사 서브스트링을 찾아내는 문제라든가 근래의 생물정보학에서 두 DNA시퀀스간의 유사도를 판별하는 문제 등에서 매우 중요한 기법이다. 본 논문에서는 두 서열들을 유사하게 매칭시켜 주는 기존의 정렬 방법을 응용, 변형하여 C, C++, JAVA등으로 짜여진 프로그램 소스들의 유사도를 측정하는 방법을 제시하였다. 실제로 이런 프로그램 소스의 표절은 대학교육 수업과정 등에서 빈번하게 발생하는 문제점으로서 본 논문에서는 프로그램 소스표절을 검사, 탐지할 수 있는 방법론 및 구체적인 프로그램과 그 결과를 제시하고 있다. 아울러 두 프로그램간의 유사성을 비교하기 위해 기존의 지역정렬 방법을 보다 효율적으로 적절히 변형시키는 방법을 제시하고 있다.

1. 서 론

두 서열을 유사하게 매칭시켜 주는 서열정렬에 관한 문제는 전산학의 고전적인 문제라 할 수 있다. 우리가 흔히 접할 수 있는 유사 서브스트링 매칭 문제는 전형적인 서열 정렬 문제이다. 그 밖에 근래에 각광받는 생물정보학의 생물정보 분석이나 텍스트 문서의 유사성을 비교하는 데에도 서열 매칭기법이 적용된다.

정렬은 이와 같이 두 개체간의 유사성을 비교하는 데에 쓰이는 전산학적인 방법이며, 이와 같은 개념은 다양하게 확장, 적용될 수 있다. 본 논문에서는 이런 개념을 확장하여 프로그램 소스간의 유사도를 측정하는 데에 정렬의 방법을 확장, 이용하였다. 프로그램 소스의 표절은 대학교육 수업과정 등에서 빈번하게 발생하는 문제점으로 그것을 발견하고 제재하는 것은 아주 중요한 문제이다. 본 논문에서는 프로그램 소스들을 그 특성에 맞게 서열화하고, 서열화된 소스 개체들에 정렬 기법을 적용하여 유사도를 측정함으로써 학내에서 유발되는 프로그램 과제 표절에 관한 문제를 해결할 수 있는 방법을 제시하였다.

한편 본 논문에서는 보다 정확한 유사도를 측정하기 위하여 기존의 정렬방법을 변형하여 적용해 보았다. 기존의 정렬방법에서 간과할 수 있는 문제점은 무엇이고, 이를 보완할 수 있는 새로운 정렬방법에 대해서 먼저 알아보겠다.

2. 기존 정렬방법의 소개

정렬 기법은 정렬할 대상을 동시에 몇 개로 지정하는지에 따라 다중정렬과 단일정렬로 나눌 수 있다. 본 논문에서 중점을 두고 있는 프로그램 소스 개체간의 비교는 특정한 두 개체를 대상으로 하므로 단일정렬의 방법을 이용한다. 단일정렬은 다시 크게 전체 정렬과 지역 정렬로 분류할 수 있다.

전체 정렬은 두 개체를 전체적인 관점에서 가장 유사해 지도록 정렬을 행하는데, 이것은 두 개체의 길이가 유사할 때에는 유용하지만 두 개체의 길이 차가 심할 때에는 좋은 결과를 얻지 못한다. 이에 반해 지역 정렬은 한 개체의 일부분이 다른 개체의 일부분과 가장 잘 들어맞도록 배열시켜 주는 방법으로, 유사한 서브스트링을 찾는 문제라든가 본 논문에서 관심을 두고 있는 프로그램 소스간의 유사한 부분을 찾아내는 데에 더 적합한 정렬 방법이다.

3. 분할 정렬 방식을 이용한 지역정렬

앞에서 설명한 기존의 지역정렬 방법은 각 서열간의 유사한 영역 중에서 오직 한 부분만을 매칭시켜 주게 된다. 이와 같은 특성은 두 서열간 유사한 부분을 찾아내는 데에는 도움이 되지만 두 서열이 얼마나 유사한지 수치로 정량화시키는 데에는 약간의 문제가 있다. 즉 제일 유사한 부분을 제외한 영역에서의 유사도는 고려되지 않는다는 것이다. 이와 같은 성질은 두 서열간의 유사도를 수치화시키고자 할 때 가장 큰 유사영역만 고려된다는 문제점을 야기한다. 다음과 같은 예를 보자.

서열 1	A	█	E	A	B	█	E
서열 2	Z	█	█	█	█	█	█
지역 정렬	B C D 만을 찾아냄.						

표-1. 기존의 지역정렬

이 경우 공백에 의한 유사도 감소로 인해 각 서열의 후반에 있는 'CD' 라는 공통 서열은 찾아내지 못하게 된다. 이와 같은 기존 지역 정렬의 성질은 두 서열의 전체 유사도를 수치화하는데 있어서 부적합한 것이다. 즉 가장 긴 매칭 지역을 제외한 유사도는 무시된다. 이와 같은 문제점을 개선하기 위해 본 연구에서는 기존의 지역 정렬 방법을 개선하였다. 즉 분할정렬 기법을 정렬 방법에 도입한 것이다. 분할정렬 기법을 도입한 개선된 지역정렬의 원리는 다음과 같다.

	A	B	C	D	E	A	B	C	D	E
Z	█	█	█	█	█	█	█	█	█	█
B		0								
C										
D				Max						
C								0		
D									Max	

표-2. 분할정렬 방식을 이용한 지역정렬

지역 정렬을 한 후 0부터 가장 큰 값을 가지는 값까지를 결과로 가지는 것은 기존의 지역정렬과 같다. 하지만 여기서 끝나는 것이 아니라 매칭된 지역의 좌상단 영역과 우하단 영역에서 지역 정렬을 재귀 호출한다. 이와 같이 함으로써 이전에 발견된 매칭영역과 중복되지 않고, 또한 서열의 순서를 고려한

전체적인 범위에서의 유사도를 구할 수 있다. 이와 같은 분할 정렬 방식의 지역 정렬은 국소 범위의 매칭만 찾아주는 지역 정렬의 단점과, 두 서열의 길이 차이가 클 때 유사도를 정확히 계산할 수 없고 최소 매칭 길이를 제한할 수 없는 전체 정렬의 단점을 보완한다.

4. 이전의 표절 검사기법

표절검사를 위한 시스템은 처음엔 일반적인 문서에 대한 표절탐지를 위해 개발되었으며, 프로그램에 대한 표절 검사는 초기엔 같은 방법을 사용하였다가 점차 프로그램의 특징을 고려한 다양한 기법들이 소개되었다.[1]

4.1 지문법(fingerprint method)

지문법은 본문 내에 나타나는 특징들을 지문과 같이 고유한 것으로 취급하고 그런 고유한 특징들이 서로 다른 개체간 얼마나 같은 빈도로 나타나는가를 표절 여부의 기준으로 쓰는 것이다. 특정한 단어의 빈도수나 관용 어휘, 특정 문장 등을 지문으로 삼을 수 있으며, 현재 구현되어 있는 각 시스템들은 여러 가지 방법으로 문서들의 특징을 상호 비교, 평가한다.

4.2 구조기반(Structure related method) 검사

프로그램 소스의 표절은 일반 문서와는 다른 특징이 있다. 몇가지 예를 들어보자면 첫째, 단어인 변수 이름을 통째로 바꾸고 표절을 하는 것이 가능하다. 둘째, 키워드를 비슷한 것으로 대체하고 표절하는 것이 가능하다. 셋째, 문단의 순서 즉 함수의 위치를 바꾸어 표절하는 것이 가능하다. 즉 일반문서와는 달리 사람의 눈에는 달리 보일 수 있게 편집을 하여 표절하는 것이 쉽게 가능하다는 것이다. 이와 같은 이유로 프로그램의 표절 검사는 보통문서에 일반적으로 적용되는 지문법보다는 프로그램의 구조에 기반한 검사기법을 적용하는 것이 더 신뢰성이 높다.

5. 표절검사를 위한 프로그램 소스의 서열화

본 연구에서는 현재 알려진 프로그램 표절 검사 기법중에서 프로그램 실행 순서에 따라 나타나는 키워드로 프로그램의 구조를 서열화하여 정렬을 행하는 방법[1]을 채택하였고, 이 때의 정렬 방법으로 앞에서 언급한 분할정렬 방식의 지역 정렬 방법을 적용하였다. 이와 같은 프로그램 소스의 서열화는 프로그램간 유사성의 문제를 단순히 1차원 서열의 유사도를 측정하는 문제로 바꿀 수 있다. 또한 프로그램의 실행순서를 고려함으로써 함수의 기술 위치를 바꾸는 등의 표절 방법을 다른 표절 기법보다 효율적으로 탐지해낼 수 있게 된다.

6. 실제 프로그램에서의 유사도 측정 실험

본 연구에서는 앞에서 언급한 프로그램 소스의 서열화 방법 및 정렬 방법을 바탕으로 프로그램의 유사성을 체크, 표절 검사를 수행할 수 있는 프로그램을 제작하였으며, 이를 바탕으로 실제 작성된 프로그램에 대해 표절 검사를 수행해 보았다. 보다 신뢰성있고 객관적인 평가를 위해 ICPC 아시아지역 예선을 겸하고 있는 2003년도 대학생 프로그래밍 경시대회를 예선, 본선 프로그램에 대해 표절 여부를 검사하였다.[7]

대학생 프로그래밍 경시대회에서도 프로그램 표절의 발생은 예외일 수 없다. 실제로 주최측에서는 사람이 직접 표절 여부를 검사하고 있으며, 표절로 인해 예선에서 실격처리된 팀이 여럿 있다. 그러나 사람의 검사에는 한계가 있어서 고작 같은 학교의 참가팀에 대해서만 표절 검사 여부를 측정하고 있으며, 이 또한 능동적인 표절의 경우에는 발견하지 못할 수 있다. 사람이 발견한 표절과 본 연구에서 개발한 시스템으로 개발한 표절 검사 결과는 표-4와 같다.

예선문제 번호	정답 갯수	추출된 키워드의 평균 길이	본선문제 번호	정답 갯수	추출된 키워드의 평균 길이
1	100	63	1	58	47
2	22	127	2	55	71
3	55	67	3	34	108
4	27	205	4	19	96
5	12	136	5	23	74
6	24	55	6	17	105
			7	10	254
			8	30	43
			9	5	193

표-3 : 2003 대학생 프로그래밍 경시대회 데이터 내용

문제 번호	팀 1	팀 2	학교	사람의 검사	본 시스템의 검사
예선 1번	92	107	같음	O	O
예선 1번	52	105	같음	O	O
예선 1번	19	100	같음	O	O
예선 1번	16	58	다름	X	O
예선 1번	66	82	다름	X	O
예선 3번	44	57	다름	X	O
예선 3번	78	108	같음	O	O

표-4 : 대학생 프로그래밍 경시대회의 표절 검사 결과

실제로 본 시스템에서 프로그램간의 유사성을 체크했을 때 프로그램의 구조가 비슷하여 표절로 볼 수 있을 정도의 유사한 프로그램을 더 볼 수 있었다. 표-4는 그 중에서도 확실한 것만 뽑아낸 것이며, 본 시스템으로는 표절이 확실한 이와 같은 프로그램 쌍 뿐만 아니라 프로그램 흐름이 비슷한 프로그램 쌍 또한 다수 검출해낼 수 있었다.

6. 프로그램 간 유사성의 통계적 성질

프로그래밍 경시대회 프로그램의 유사성을 분석한 결과 몇가지 흥미있는 통계적 특징들을 발견할 수 있었다. 첫째로 같은 목적으로 제작된 프로그램인 경우 그렇지 않은 경우보다 표준편차가 확연히 컸다. 이것은 같은 문제에 대한 프로그램인 경우 비슷한 구조를 가지거나 매우 관계가 없는 두 극단으로 나뉘는 데 반해 다른 문제에 대한 프로그램인 경우는 전반적으로 관계가 없는 구조를 띠기 때문인 것으로 해석된다.

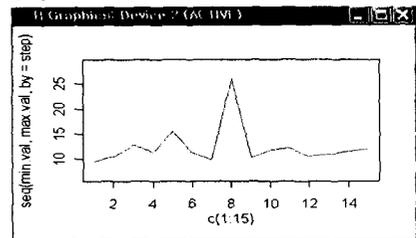


그림-1. 예선 본선 모든 문제 상호간 계산한 표준편차 값

그림-1은 통계 툴인 R로 본선 2번과 나머지 14문제와의 유사도의 편차를 나타낸 것으로, 같은 문제간의 편차가 다른 것과 확연히 구분됨을 알 수 있다. 실험 결과 15개의 문제 중에서 14개가 자기 자신과의 유사도 편차가 제일 큰 값을 가졌다. 두번째 특징은 일반적인 프로그램의 유사도는 프로그램의 키워드 길이가 늘어나더라도 서서히 증가한다는 것이다. 그리고 이 때의 유사도는 같은 프로그램일 경우가 그렇지 않은 경우보

다 전반적으로 큰 값을 가진다. 그림-2는 이것을 보여주는 그림이다. 또한 첫 번째 사실에서도 언급했듯이 같은 프로그램의 유사도 간에는 편차가 커서 변화가 심하다는 것 또한 관측할 수 있다.

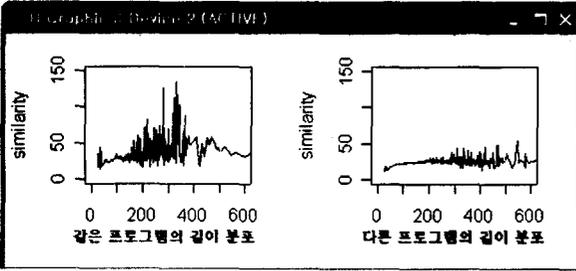


그림-2. 같은 프로그램(좌)과 다른 프로그램(우)의 키워드 길이에 따른 유사도 분포. 좌측의 값이 우측의 값에 비해 전반적으로 크다. 한편 표절된 프로그램은 이와는 다른 양상을 띤다. 그림-3에서 보듯 서로 관계가 없는 프로그램간 유사도는 추출된 키워드 수가 늘어나더라도 아주 천천히 증가하고 있다. 그에 반해 표절된 프로그램은 추출된 키워드가 증가했을 때 그만큼 유사도가 많이 증가하므로 비교적 가파르게 증가함을 알 수 있다.

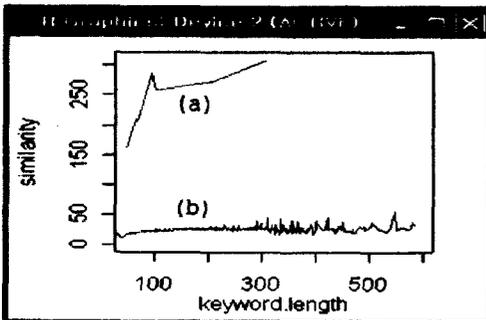


그림-3. (a)표절된 프로그램과, (b)정상적인 프로그램간의 추출된 키워드 길이에 따른 유사도 그래프

세 번째 특징은 각 문제에 대해서 프로그램 간의 유사도는 근사적으로 정규분포의 형태를 띤다는 것이다. 이것은 매우 중요한 성질로, 특정 문제에 대한 일반적인 유사도의 분포를 확립하면 이 정규분포의 곡선을 기준으로 하여 특정 프로그램간의 유사도가 어디에 위치하는지를 분석하는 것만으로 표절인지 아닌지의 여부를 판단할 수 있게 된다.

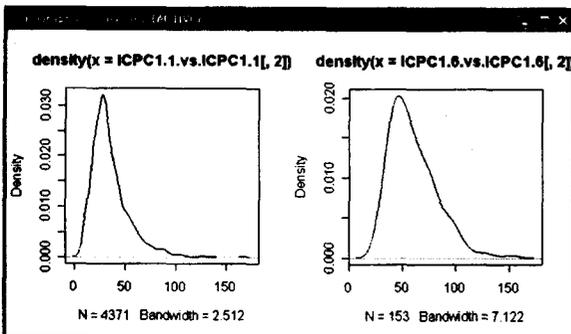


그림-4. 특정 문제에 대한 유사도의 분포

그림-4는 예선 두 문제의 모든 프로그램간에 존재하는 유사도에 대한 히스토그램이다. 예상할 수 있듯이 가운데를 기준으로 정규분포와 유사한 형태를 띠고 있다. 이와 같은 형태의 규칙적인 분포는 이것을 모집단으로 구한 평균과 편차를 바탕으로 특정한 similarity 값 이상을 가지는 프로그램 쌍을 표절로 판단하게 할 수 있는 객관적인 근거를 제시한다. 즉 분포도의 오른쪽에 나타나는 높은 유사도 값들의 percentile은 정규분포의 분석에서 쓰이는 표준 편차의 상수배로서 표시할 수 있고, 이를 이용하여 특정 분포의 상위 몇퍼센트를 표절로 볼 수 있을지 하는 판단에 대한 객관적인 기준을 마련할 수 있다.

7. 결론 및 개선사항

본 논문에서는 실험 순서에 따라 프로그램의 특성을 서열화 하는 방법[1]을 적용하여 생성된 서열을, 기존의 지역 정렬방법을 전체적인 범위에 적용 가능하도록 개선한 분할정렬 방식의 지역 정렬 방법을 이용하여 정렬함으로써 프로그램간의 표절 여부를 검사하는 시스템을 제시하였다. 그리고 이 시스템을 사용하여 신뢰성 있는 대회인 전국 대학생 프로그래밍 경시대회에 사용되었던 프로그램들의 표절 여부를 검사해 보았다. 자료가 대형화됨에 따라 사람의 수동적인 검사로는 간과되거나 정확히 발견될 수 없는 표절의 문제를 여기서 제시한 시스템으로 효율적이고 정확하게 검사할 수 있음을 보였다.

논문에서도 언급하였고, 또한 쉽게 예측해 볼 수 있듯이 임의의 두 프로그램간의 유사도는 프로그램의 목적과 길이에 따라 특정한 통계적 특성을 띠게 됨을 알 수 있었다. 특히 표절된 프로그램의 통계적 수치는 이와 같은 특성을 벗어나는 특이한 현상으로 나타나고, 따라서 정량화된 통계적 수치를 이용하여 표절된 프로그램을 쉽게 찾아낼 수 있음을 보였다. 다만 실험에 사용된 데이터가 그다지 크지 않고 실험이 많이 이루어지지 않아 프로그램 간의 유사도에 대한 보다 일반적인 통계적 특성을 밝혀 내는 데에는 미진한 점이 있었다. 향후에는 보다 다량의 다양한 프로그램들에 대한 실험을 계속 진행하여 일반적인 프로그램의 통계적 특성을 밝혀내고, 임의의 프로그램간 유사도 크기를 평가할 수 있는 보편적 기준이 될 수 있는 수치를 마련해야 할 것이다. 그래서 프로그램의 표절 여부를 통계적으로 쉽게 검증할 수 있게 해야 할 것이다.

8. 참고자료

- [1] 강은미, 황미영, 조한규, "유전체 서열의 정렬 기법을 이용한 소스코드 표절 검사", 정보과학회, 2003
- [2] Alexander N. Mikoyan, "Using CBR Techniques to detect plagiarism in computing assignments"
- [3] Lutz Prechelt, Guido Malpohl, Michael Philippsen, "Finding Plagiarisms among a set of programs with JPlag", Journal of Universal Computer Science, 2001
- [4] Michael J. Wise, "YAP3 : Improved detection of similarities in computer program and other texts"
- [5] Mike Joy, Michael Luck, "Plagiarism in Programming Assignments", IEEE Transactions on education, 1999
- [6] Paul Clough, "Old and new challenges in automatic plagiarism detection"
- [7] http://www.kado.or.kr/f2/f2_s6_t5.asp
- [8] Martin Tompa, "Lecture Notes on Biological Sequence Analysis", 2000
- [9] David W. Mount, "Bioinformatics - Sequence and genome analysis", 2001