WCES-2010

Support for checking plagiarism in e-learning

Daniela Chuda*, Pavol Navrat

*IISE FIIT Slovak University of Technology, Ilkovicova 3, 812 19 Bratislava, Slovakia

Received October 26, 2009; revised December 2, 2009; accepted January 13, 2010

Abstract

The paper attempts to analyze current situation in plagiarism detection and to analyze existing methods and tools for checking the plagiarized programming code and natural language (particularly Slovak) text. Then, we describe our approach to plagiarism detection for two particular but important cases: texts in formal programming language and texts in natural language (Slovak). We describe method and tools that evaluate plagiarized programming code and plagiarized Slovak text. Our method and tool proposal has an ambition to improve some of the known weaknesses. We aim at implementing an effective, widely usable tool with more precise results. We are in the process of setting it in our Faculty environment.

Keywords: Plagiarism; plagiarized programming code; plagiarized text; evaluating of assignment.

1. Plagiarism

Plagiarism is a global problem, which occurs in many different areas of our life. There are many different forms of plagiarism. Plagiarism at schools can be a highly de-motivating factor for teachers and also for students (Chudá, 2009a). If plagiarism is not addressed sufficiently, plagiarists could gain undeserved advantage, e.g. more marks for their assignments with less effort.

In e-learning systems students upload their assignments as a written text in natural language and, in certain study programs, as a program code. Students use various approaches to create plagiarized assignments, copy and paste being the most elementary one. There are various types of plagiarism (What is plagiarism, 2009) involved: using sources without properly citing them, paraphrasing text, reusing ideas with/without citing references, and others.

2. Related work

We reviewed several existing methods and freely accessible tools for the plagiarism detection. We are focused on tools from an academic background. For testing purposes we used several real-life assignments written in Slovak language and in programming language. They were plagiarized manually by us in five different ways and levels. The tools use a number of different methods, or their combinations for plagiarism detection: tokenization, greedy
string tiling, Karp-Rabin algorithm, Heckel’s algorithm, k-grams, string matching algorithm (Clough, 2000). Results of a comprehensive comparative analysis are in Tab. 1.

Besides many strong points, the methods and tools have some weaker points, too. Most of them share disadvantages such as

- a weaker graphical user interface, and in particular poor support for setting configuration options,
- ignoring comments and text strings,
- a weak robustness against some slight changes, such as rewriting switch statement into a series of if statements, rewriting one print statement into several ones, rewriting one kind of loop statement into another one,
- too general tokens are a disadvantage for smaller programs, it may result in many false positives,
- inability to recognize reordering of operands in expressions.

None of the tools that we reviewed process texts in Slovak, neither is able to process a text with diacritics.

<table>
<thead>
<tr>
<th>Tools/Name</th>
<th>SIM</th>
<th>JPlag</th>
<th>PMD’sCPD</th>
<th>YAP3</th>
<th>Sherlock</th>
<th>MOSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of application</td>
<td>console</td>
<td>Web application, java</td>
<td>Web application., java application.</td>
<td>-</td>
<td>Java application</td>
<td>Web servise, Perl script</td>
</tr>
<tr>
<td>Methods, algorithms</td>
<td>Tokenization</td>
<td>Tokenization + Greedy String Tiling</td>
<td>Tokenization + Greedy String Tiling</td>
<td>incremental comparison of two files</td>
<td>WINnowing algorithm (Schleimer, 2003)</td>
<td></td>
</tr>
<tr>
<td>GUI</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Result check</td>
<td>visual similarity of two parts</td>
<td>visual similarity of the parts, color</td>
<td>Only one similar parts</td>
<td>visual similarity of two parts</td>
<td>visual similarity of two parts, color</td>
<td></td>
</tr>
<tr>
<td>Way of measuring similarity</td>
<td>Fraction in percent, numbers of similar rows</td>
<td>Fraction in percent, histogram, group of similar files</td>
<td>Number of similar rows, tokens</td>
<td>Sum of percent, similarity graph</td>
<td>Fraction in percent</td>
<td></td>
</tr>
</tbody>
</table>

3. Our approach to plagiarism detection method and tools

We identified several possibilities of improvement in the analysed tools. This has been a solid basis for devising our tools that attempt to detect similarity between student works. One of the ideas we brought in is to combine such a tool with our portal for student work submission. For student works written in natural language, we attempt to develop processing of a complete Slovak alphabet, i.e. including letters with diacritics. Also, existence of several forms of a single word in Slovak language creates a specific problem that we attempt to tackle.

3.1. Detecting plagiarized programming code

We decided to use sophisticated tokenization method for program pre-processing and the greedy string tiling for tokens comparison.

The common tokenization method will be improved and all types of cycles will be replaced with the same token. The synonyms will be converted to one form. The architecture of SID Plag method is in Fig 1.

Our method will work as follows:

- if a program consists of more files, these files will be integrated into one according to their size,
- renaming variables – all variables of type <type> will be renamed to <type>_ID,
- transformation to lower case,
extracting strings and comments – strings and comments will be compared separately from the rest of the program,
- tokenization – transformation of the program into tokens, synonyms will be transformed into the same token (Kováčová, 2009b),
- comparison of tokens – the greedy string tiling algorithm will be used because it is resistant to the reordering of statements,
- presentation of the results to the user.

In Tab. 2 we give comparison of testing seven pairs of student works (Kováčová, 2009a). It summarizes results of testing them by our tool SID Plag and by tools JPlag, SIM and Sherlock. The first five pairs of works are free of any plagiarism. The sixth pair of works is contaminated with plagiarism. The seventh pair of works is formed by an original text and a plagiarized version of it, created artificially by ourselves by changing variable names, comments, while cycle to do-while cycle, ordering of operands, ordering of commands, types of variables and so on.

<table>
<thead>
<tr>
<th>pair of student works</th>
<th>plagiarized?</th>
<th>SID Plag</th>
<th>JPlag</th>
<th>SIM</th>
<th>Sherlock</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. no</td>
<td>4/5%</td>
<td>less than 10%</td>
<td>0</td>
<td>5%</td>
<td></td>
</tr>
<tr>
<td>2. no</td>
<td>2/3%</td>
<td>less than 10%</td>
<td>0</td>
<td>13%</td>
<td></td>
</tr>
<tr>
<td>3. no</td>
<td>21/7%</td>
<td>less than 10%</td>
<td>0</td>
<td>15%</td>
<td></td>
</tr>
<tr>
<td>4. no</td>
<td>9/11%</td>
<td>less than 10%</td>
<td>0</td>
<td>13%</td>
<td></td>
</tr>
<tr>
<td>5. no</td>
<td>9/16%</td>
<td>less than 10%</td>
<td>0</td>
<td>14%</td>
<td></td>
</tr>
<tr>
<td>6. yes, by ourselves</td>
<td>82/78%</td>
<td>61/64%</td>
<td>67/64%</td>
<td>55%</td>
<td></td>
</tr>
<tr>
<td>7. yes, by ourselves</td>
<td>95/92%</td>
<td>85/89%</td>
<td>68/66%</td>
<td>75%</td>
<td></td>
</tr>
</tbody>
</table>

Our tool evaluates possible plagiarism in both directions. We realized the relation of being “similar due to plagiarism” is not necessarily symmetrical. Imagine a pair of two student works, first of 20 lines and the second of 30 lines, formed by copying the 20 lines and adding 10 other lines of program code. Clearly, fraction of copied work differs if we consider how much is copied from the latter in the former work or vice versa. To have a homogenous comparison, we suggest taking the higher of the two values in the column SID Plag.

Our method and tool for detecting plagiarism in programming text is based on several particular improvements. We process and compare separately program codes, comments and strings; we transform all kinds of loops into a single token. We process synonyms because they are very likely candidates for plagiarizing. Another special case we process is verbatim copies.

3.2. Detecting plagiarized Slovak text

There are a few special problems to be addressed when processing works written in Slovak language. The thing is that unless we address them, a general (or English language) text tool performs quite ineffectively. Not only Slovak uses a partly different (i.e. enhanced) alphabet, Slovak grammar is more complicated. We offer an option of removing diacritics to address the problem of the difference in alphabets. Our tool PlaDeS (Chudá, 2009b) preprocesses submitted texts, which includes removing stop words and stemming.

Our tool is able to process various types of files, including pdf and doc files, and convert them into text files. Before converting, we propose to check metadata. They are source of possibly valuable information on authorship, processing date, used templates, etc. In one version, we experiment with utilizing readability index (Návrat, 2007). The architecture of PlaDeS tool is in Fig 2.
4. Process of evaluation of student works

Plagiarism detection at our institution is based on various checking approaches and procedures. We decided to combine it with a standard electronic student work submission portal that has been devised within (Learning Management System) Moodle. We define exactly conditions for submitting a program code and related documentation. In the Moodle environment, every student is identified and authenticated uniquely. Usually, after having uploaded their homework, students present their works orally during classes.

Thus at an elementary level, teacher to whom student hands in her homework, lets the student present the work orally or present how the program works. An experienced teacher is able to recognize quite quickly if the student presents her own work. Sometimes, however, the case is not so self evident. Teacher asks clarifying questions. Of course, the student realizes their purpose is to confirm or refute a suspicion of plagiarism. Tension grows. Whatever the outcome, the interview leaves an uncomfortable taste at both sides. Student, if originality of her work was challenged without reason, feels offended. Teacher, having experienced 20, 40 or even 60 such presentations on a single day, feels first and foremost exhausted both physically and mentally.

Besides such subjective complications, there are some objective circumstances that make plagiarism detection difficult. Within one subject, there can be several groups of students, their labs being taught by several assistants. However, home assignments may partly overlap. Students may, for various reasons, submit plagiarized works. Their assistant can hardly detect the plagiarism if the source work is authored by a student from the class but from a different group, taught by a different assistant.

Teacher is also challenged emotionally, since many students try to avoid the worst by coming with superficial explanations that are hard to refute, try to bargain or even lie, and all this creates a pressure on him. However, deep down the teachers know that this approach has a fundamental weakness, if not a flaw. They know that in our culture, it is the duty of the pursuant to produce a proof of guilt. Even if the student does not demand: Prove that I plagiarized!, as the students frequently do, teachers know they should but they cannot solely by themselves. That is why there are calls that all the submitted works are to be checked by a plagiarism detection tool. However, a tool does not decide about existence of plagiarism, nor does it determine which work is source and which is plagiarized. Resulting suspicious cases are reviewed by the teacher. If manual review substantiates the suspicion, student fails and is referred to the disciplinary committee.

5. Our experience with detection of plagiarism

A more sophisticated level of checking plagiarism is to use some detection tool. This seems to be an obvious concept that must lead to improvement. Nothing is farther from truth. All such tools are based on some kind of comparison between a given student work and other work or works which are candidates of plagiarized sources. But how does the tool know which are the candidates of plagiarized sources? Usually, there are only few clues. The tool needs a definition of a set of student works that are to be considered. Some tools have a database of student works, which has been formed gradually. For many fields, such a closed world may prove effective. In computing, if the set of candidates of plagiarized sources is not open to anything available on the web, any tool will find only a subset of possible cases of plagiarism. But if a student work is to be checked against anything available on the web, besides the usual databases of theses, homework etc., it would require a dramatically different kind of tool. So far, research
in the area focuses mostly on the former kind of tool. Within this kind, a special class of tools is concerned with detecting plagiarism between computer programs. This requires special algorithms for finding similarities.

We created at our institution our own detection tool and started to use it a few years ago first in one subject, then in a few others. Our experience so far can be summarized in a few points. Detecting similarity between two programs requires much more sophisticated rules than those used for a plain text in natural language. The notion of similarity is much finer than just fraction of copied text. Non dedicated algorithms do not take the syntax and semantics of the programs into account. Even if not open world is attempted, success of a tool dramatically depends on the data it has available, i.e. the database of previous works.

A detection tool, or any tool, can never be the ultimate cure. Changing attitudes of all the stakeholders is much harder, but a real progress is unimaginable without it. When students realized that we work actively on the problem, take measures and mean it seriously, number of attempts of plagiarism decreased. There have been various measures tried, of a varying degree of severity: from reduction of marks achieved in a course, to failing the course and referral to disciplinary committee. Note the dual track measure that is emerging recently. Failing a course is not a punishment, but a simple consequence of the fact that marks can be earned for an independent work only. At the same time, plagiarism is a gross breach of the rules, so a disciplinary action, usually an expulsion from the university on probation is proper.

6. Conclusion

We described our method and experience with the whole process of checking plagiarism in practical classes. Our tool incorporates some rules that reflect syntax or semantics of the language. Despite their incompleteness and simplicity, they contribute to improving effectiveness of the tool. Worth noting is also the tool that recognizes some specific features of Slovak language.

We wish to stress that any tool alone can never be the cure. Changing attitudes of all the stakeholders is much harder, but a real progress is unimaginable without it. It is important to understand that a tool solely cannot be the solution, since the problem is the people. We cannot pass the responsibility to some tool. There must be someone responsible for detecting, gathering evidence and making judgment. Ethical problems will never be resolved by a software tool.

References


JPlag tool, Retrieved February 22th, 2009 from <https://www.ipd.uni karlsruhe.de/jplag/>


Sherlock, BOSS tool, Retrieved February 22th, 2009 from <http://www.boss.org.uk/)

