Plagiarism Detection in SQL Student Assignments

Nikolai Scerbakov[1(✉)], Alexander Schukin[2], and Oleg Sabinin[2]

[1] Institute of Interactive Systems and Data Science, Graz University of Technology, Graz, Austria
nsherbak@iicm.edu

[2] Institute of Computer Science and Technology, Peter the Great Saint-Petersburg Polytechnic University, St. Petersburg, Russia

Abstract. An original method for plagiarism detection in SQL student assignments has been proposed. The method is based on identifying so-called "SQL lexemes" - persistent elements of an SQL statement, and "SQL variables" - easily modifiable elements of SQL statements. Thus, any SQL statements can be replaced with a so-called token - sequence of SQL lexemes and SQL variables. Distance between SQL tokens can be calculated using such a well-known algorithm as Levenshtein Metric. Small values of Levenshtein distance between tokens detect such SQL statements that were built by modifications of others.

We also present first practical results of actual application of the algorithm, and discuss further developments of the method.

Keywords: e-Learning · Automatic evaluation · Automatic grading

1 Introduction

Learning by doing seems to be a common learning paradigm in teaching programming, databases and other computer science topics [2–7]. Normally, students are requested to implement practical assignments that can be seen as a practical application of obtained knowledge in the selected area. If we speak about databases, a student assignment is typically a definition of a database schema and of a number of queries by means of so-called SQL - standard language for defining and accessing databases that is supported by all relational Database Management Systems (DBMS) like Oracle, mySQL, Ingress, etc.

Checking and grading of students' assignments require a substantial amount of tedious work by teachers. Grading a particular assignment, the teacher must answer three main questions:

- was a required database functionality correctly implemented?
- is this implementation optimal?
- is this solution original or was done by cosmetic modifications of another assignment?

All the three issues above are reasonably complex; in this paper we concentrate on the third question - on automatic identification such assignments that were done by means of modification of other assignments.

M. E. Auer et al. (eds.), *Teaching and Learning in a Digital World*,
Advances in Intelligent Systems and Computing 716,
https://doi.org/10.1007/978-3-319-73204-6_14

The problem of such identification deserves an individual investigation since:

– this is rather usual when students take an existing assignment and modify it by replacing original identifiers with other identifier (say, “student”→”pupil”, “student_name” →”pupil_name”, “lecture”→”class”, “teaching_book”→”lecture_notes”, etc.);
– standard methods of plagiarism detection does not work in such cases, since the methods are based on a detection of similarity of textual fragments, while SQL fragments look entirely different after the replacements as above.

2 Learning Management Environment

TU Graz TeachCenter is an innovative Learning Management System (LMS) that is used at Graz University of Technology, Austria for several years. Currently the system supports about 1500 individual courses and more than 20000 users. Normally, about 400 users are concurrently online. The system implements a number of different e-learning scenarios such as uploading individual assignments, uploading group projects, collaborative authoring, etc. One of the most popular components of Teach Center is a called “Programming Assignments”, and allows students to upload executable code as an assignment. The system automatically evaluates such course fragments and provides results of such evaluation.

The programming assignments are supposed to be uploaded into so-called “Group Lockers” (Fig. 1). “Group Locker” is a named memory space protected by a special key (Password). Anyone knowing the password may access and upload files into the group locker. Normally, students are requested to create lockers themselves. Names of the lockers are used to identify content. For example, lockers are often named “Group …”, “Project …” etc. The system supports different programming languages and can be set to work with SQL student assignments (Fig. 2).

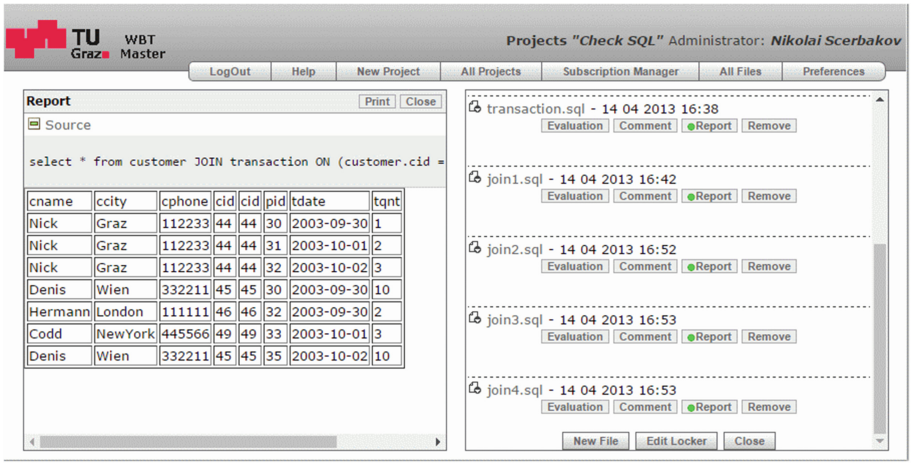

Fig. 1. Programming assignments.

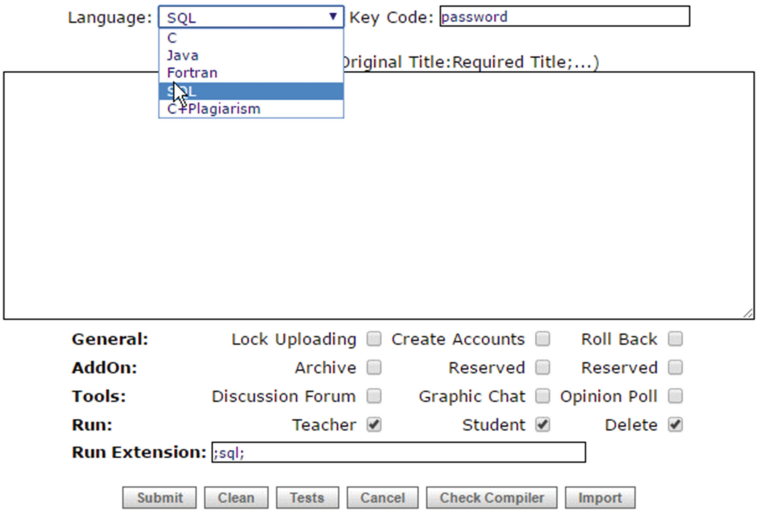

Fig. 2. Adjusting the system to evaluate SQL assignments

Typically, assignment requirements look as follows:

Please, develop a database application that consists of:

- Database schema of 4 relations;
- Two queries in terms of SQL. Each query should operate with 2-3 relations;
- Two SQL queries to illustrate the Join and set operations;
- Two SQL queries illustrating Group By and Having statements.

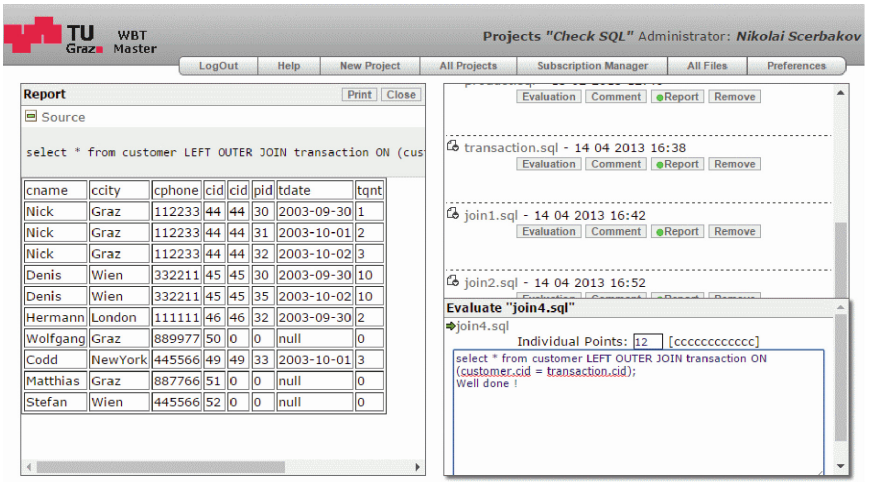

Fig. 3. Grading a student assignment

Students are supposed to upload local files having *.sql extension, and the system automatically evaluates the sources. The results are shown in the form of so-called "reports" (Fig. 3). Each report includes the source text and results of the automatic evaluation.

Grading of such programming assignments is a very tedious work. The teacher must look onto a source text and report that was produced as the file was evaluated, and answer the following questions:

– was the required database functionality correctly implemented?
– is this implementation optimal?
– is this solution original or was done by cosmetic modifications of another assignment?

Automatic evaluation of student files greatly facilitates checking correctness of the source texts. At the same time, standard methods of plagiarism detection does not work in the case of slightly modified SQL statements that may look entirely different after the simple replacements of titles for tables and attributes.

3 Plagiarism Detection

The method is based on identifying so-called "SQL lexemes" - persistent elements of an SQL statement, and "SQL variables" and "SQL constants" - easily modifiable elements of SQL statements. SQL variables and constants can be normalized, i.e. replaced with automatically generated titles in such a way that a certain normalized title replaces all occurrences of a particular SQL variable or constant. Thus, any SQL statements can be replaced with a so-called token - sequence of SQL lexemes and normalized titles.

For example, the four SQL statements can be converted into tokens as below:

```
CREATE TABLE `customer` (`cname` varchar(50),`ccity`
varchar(50),`cphone` int(11),`cid` int(11),
PRIMARY KEY  (`cid`));
```

►CREATE TABLE #01 (#02 varchar(#N),#03 varchar(#N), #04 int(#N),#05 int(#N), PRIMARY KEY (#05));

```
CREATE TABLE `product` (`pname` varchar(50),`pprice`
int(11) `pid` int(11) PRIMARY KEY  (`pid`));
```

►CREATE TABLE #11 (#12 varchar(#N),#13 int(#N) #14 int(#N) PRIMARY KEY (#14));

```
CREATE TABLE `transaction`(`cid` int(11), `pid` int(11),
`tdate` date,`tqnt` int(11),PRIMARY KEY  (`cid`
`pid`,`tdate`));
```

►CREATE TABLE #21 (#05 int(#N),#14 int(#N),#22 date, #23 int(#N), PRIMARY KEY (#05, #14, #22));

```
SELECT cname, ccity, cphone FROM customer WHERE cid IN
 (SELECT cid FROM transaction Where pid IN
   (SELECT pid FROM product Where pname = 'VDU'));
```

►SELECT #02,#03,#04 FROM #01 WHERE #05 IN (SELECT #05 FROM #21 WHERE #14 IN (SELECT #14 FROM #11 WHERE #12 = #S));

Distance between SQL tokens can be calculated using such a well-known algorithm as Levenshtein Metric [1]. We calculate a Levenshtein distance [1] between two tokens as a minimum number of edit operations with normalized titles (insert, delete or replace) required to change one token into the other. Note, we consider any difference in SQL lexemes as a sign of absence of plagiarism. Small values of Levenshtein distance [1] between tokens detect such SQL statements that were built by modifications of others.

For example, the student assignment below is easily identified as an assignment suspicious for a plagiarism since tokens are simply identical, Levenshtein distance is equal to 0.

```
CREATE TABLE `patient` (`pat_name` varchar(30),`address`
varchar(60),`insurance` int(7), `pat_id` int(7)) PRIMARY
KEY  (`pat_id`));
```

►CREATE TABLE #01 (#02 varchar(#N),#03 varchar(#N), #04 int(#N),#05 int(#N), PRIMARY KEY (#05));

```
CREATE TABLE `medicine` (`med_name`
varchar(50),`med_length` int(4) `med_id` int(7) PRIMARY
KEY  (`med_id`));
```

►CREATE TABLE #11 (#12 varchar(#N), #13 int(#N), #14 int(#N) PRIMARY KEY (#14));

```
CREATE TABLE `prescription`(`pat_id` int(7), `med_id`
int(7),  `pdate` date,`amount` int(11)),PRIMARY KEY
(`cid` `pid`,`tdate`));
```

►CREATE TABLE #21 (#02 int(#N), #14 int(#N), #22 date, #23 int(#N)), PRIMARY KEY (#02, #14, #22));

```
SELECT pat_name, address, insurance FROM patient WHERE
pat_id IN
  (SELECT pat_id FROM prescription Where med_id IN
  (SELECT med_id FROM medicine Where med_name ='Dysport'));
```

►SELECT #02,#03,#04 FROM #01 WHERE #05 IN (SELECT #05 FROM #21 WHERE #14 IN (SELECT #14 FROM #11 WHERE #12 = #S));

In more complex cases, the system can be adjusted by setting an upper limit for the value of the Levenshtein distance to identify suspicious assignments.

4 Conclusion

In this paper, we proposed a rather simple method for plagiarism detection in SQL student assignments. The method is based on converting students' assignments into so-called tokens, and calculating a Levenshtein distance between such tokens.

The system demonstrated rather good functionality. Thus, manually we could find out just 2–3 cases of plagiarism while grading 400 user assignments, the system identified 44 cases for the same amount of assignments. Of course, all the cases were checked manually, and students were asked to come for an additional interview. As a results, 18 cases we found out as really cases of plagiarism. Such a big number of plagiarisms can

be explained by the fact that this was a first time we applied the system, and students were sure that that cases of plagiarism will not be detected manually. We also found that the system could be used as "early warning of plagiarism" for students, since substantial number of user assignments were identified as a plagiarism but students provided acceptable explanations of such similarity.

References

1. Black, P.E. (ed.): Levenshtein distance. In: Dictionary of Algorithms and Data Structures [https://xlinux.nist.gov/dads/]. U.S. National Institute of Standards and Technology (2008). Accessed 4 May 2017
2. Macfadyen, L.P., Dawson, S.: Mining LMS data to develop an "early warning system" for educators: a proof of concept. Comput. Educ. **54**(2), 588–599 (2010)
3. Wu, J.H., Tennyson, R.D., Hsia, T.L.: A study of student satisfaction in a blended e-learning system environment. Comput. Educ. **55**(1), 155–164 (2010)
4. Dietinger, T., Maurer, H.: GENTLE – General Network Training and Learning Environment. In: Proceedings of ED-MEDIA98/ED-TELECOM 1998, Freiburg, pp. 274–280 (1998)
5. Ebner, M., Scerbakov, N., Maurer, H.: New features for e-learning in higher education for civil engineering. J. Univ. Sci. Technol. Learn. **1**(1), 93–106 (2016)
6. Scerbakov, A., Ebner, M., Scerbakov, N.: Using cloud services in a modern learning management system. J. Comput. Inf. Technol. **23**(1), 75–86 (2015)
7. Scerbakov, N.: TU Graz Teach-Center (2001). http://coronet-iicm.tugraz.at/wbtmaster/welcome.html. Accessed 13 Apr 2017