ONLINE ASSESSMENT AND CHECKING OF SQL: DETECTING AND PREVENTING PLAGIARISM

Gordon Russell
Napier University
10 Colinton Road
Edinburgh
g.russell@napier.ac.uk
http://grussell.org

Andrew Cumming
Napier University
10 Colinton Road
Edinburgh
a.cumming@napier.ac.uk
http://www.dcs.napier.ac.uk/~andrew

ABSTRACT

The automatic checking of online assessments and tutorials offers a significant advantage to students. Such students can work out-of-hours, from home or work, managing their own time allocation. This allows formal practical sessions to concentrate on learning, and not on assessments. However, there is a danger that students will abuse such systems, invalidating the assessment process. This paper investigates the plagiarism detected in a learning environment for SQL, and the effectiveness of the different techniques that it has used to eliminate plagiarism.

Keywords

Plagiarism, SQL, Databases, Automatic marking, Automatic checking.

1. INTRODUCTION

The students learning SQL at Napier University make use of a home-grown integrated learning environment [4], called *ActiveSQL*. This provides SQL tutorials, and also supports online incremental assessments. The tutorials and assessments are all marked automatically and immediately by the system, giving a perception of immediate feedback to the students.

One problem with the use of automatic checking, especially when combined with online assessments, is that it is hard to validate that the work has not been plagiarised. Such systems may actually encourage plagiarism, as students may feel they are not cheating the Lecturer but are simply cheating a computer. The detachment of human involvement in the marking process may also give rise to a belief that such actions would be hard to detect, and even if detected would be hard to act on.

Common plagiarism tools often rely on having significant student material to apply statistical analysis algorithms to detect changes in style, or having student material that by its nature has many variations (much of which dictated by the personality of the student in question). In programming exercises for instance, aspects of the code such as variable names, function names, or even the approach to loop constructs (*while, do, for, foreach*, etc) can vary significantly between two students attempting to solve the same problem.

In SQL exercises targeted at solving particular data querying problems, some student solutions will appear to be similar to that of other students. One could increase solution variations by asking “bigger” questions, with design, implementation, and testing components. This approach however is harder to assess automatically, and more importantly is difficult to offer as an incremental assessment without increasing lecturer workload. In addition, the author’s investigations have shown that there are significant benefits to having small, incremental style assessments when learning SQL [3].

This paper gives a brief overview of the automatic grading process used in ActiveSQL. It then proposes and analyses an approach to plagiarism detection for SQL. The results of applying this algorithm is visualised, and plagiarism behavior data spanning the last three years is discussed. Finally conclusions are drawn for those interested in assessing SQL online specifically and on online assessments in general.

2. AUTOMATIC CHECKING

The checking algorithm [3] used by ActiveSQL is in two parts. The main check is on *Accuracy*, and compares the result of executing the student’s SQL against that achieved when executing the sample solution. This gives a percentage accuracy mark which corresponds to how closely the two results match up. This accuracy measure includes a

Teaching, Learning and Assessment in Databases, University of Sunderland

method for detecting "hard coded" solutions that fail to work correctly if the dataset changes.

The second check is based on a number of hand-coded heuristics, which attempt to measure the quality of the actual SQL submitted (and ignoring how well the SQL actually performs). This measures things including query length, use of distinct, and whether LIKE was used where "=" should be. Breaking these rules result in penalty marks. This second check also produces textual feedback to the students (e.g. "Your query is too complicated") and gives the illusion of intelligence. In turn, written student feedback for this system shows that student acceptance of automatic marking was greatly enhanced as a result of using these "human-like" feedback responses.

3. DETECTION

Manual detection of plagiarism in the author's database module involves going through over 300 different student's submissions, each of which may involve 10 different SQL statements. This is both time consuming and error prone. An automatic scheme is attractive, if at least to filter out clearly non-matching submissions.

Initially a number of algorithms were written to compare two SQL statements, and the results of each algorithm was weighted, summed, and then if the number was over a particular threshold the match was considered positive. This approach proved to be unworkable. Not only were the weights arbitrary, but as the number of algorithms used increased, the complexity in handling and justifying the weights grew excessively.

The second approach undertaken was to separate the comparison process from the identification of plagiarists. This allowed us to generate data to show plagiarism "suspicions", which could be analyzed in a second phase and either dismissed or approved. The comparison algorithm stores the comparison results in an XML document. This document could then be viewed manually, or processed automatically, as part of our investigations. The XML for each question copied can have the following information.

```
<equality quality='n' />
<shuffle flips='m' />
<histogram>
  <word word='v'
        left='f1' right='f2' />
  <signature op='w' comma='x'
        trail='y' />
</histogram>
```

3.1 Equality Matches

Here the quality of the match is a number from 4 to 10. This algorithm is based on simple string equality. The idea here is to quickly and efficiently detect students who have "cut and pasted" SQL electronically, and then made only minor cosmetic changes. The quality n corresponds to a range from 10 (a perfect string match) to 4 (a string match ignoring case, non-essential white space, and brackets).

This algorithm turns out to catch most students. Normal disguise attempts of the students include changing "select" to "SELECT", or inserting returns into the middle of a statement.

3.2 Shuffle Flips

This is based on the idea of Heckles [2] essay plagiarism algorithm applied to SQL. This algorithm is used a number of programming language plagiarism detection systems [1] [6]. The detection issue is picking up on lines moved around. This problem is significant in SQL, as some parts of the text which makes up a query can be reorganized without affecting the performance of the query (e.g. AND lines in a WHERE clause can be reordered). The attribute *flips* gives a measure of how many lines had to be moved around.

In *Shuffle*, the concept of a line is not the same as text with a return character at the end. Instead *Shuffle* reformats the user's SQL into a regular structure. For instance, operators like "AND" and "WHERE" start the lines off. In addition *Shuffle* can be coded to ignore brackets, aliases, case changes, and *equal flips*. Equal flips make the comparison tolerant to changes where the operands on each side of a comparison are switched over, such as changing from "`WHERE a = b`" to "`WHERE b = a`".

Shuffle was written analyze attempts to disguise SQL plagiarism, reporting a "distance" between one SQL query and another. Currently *Equality* is usually sufficient to detect plagiarism in our current data sets. However, it is assumed that as the students become more aware of the plagiarism detection system, they may begin disguising the plagiarism. *Equality* is much faster than *Shuffle*, so is executed first, and *Shuffle* is used only if *Equality* fails to find a match.

3.3 Histogram

In the Histogram tag, data is given to provide evidence that can be presented to the students involved to help counter claims that the plagiarism could have happened by coincidence. It is after all possible for two students to work independently and still produce identical SQL statements.

Histogram parses all the SQL ever submitted and hunts out highly unusual words. Everyone uses "SELECT" but perhaps only two students use "myview1" as a view name… Two students with identical SQL and who are the only ones to use "myview1" would therefore have no "just a co-incidence" argument. Where the two SQL statements have an unusual word, this is shown

```
<shuffle score='0.888889' restarts="0" />
<signature op='0' comma='1' trail='1' />
<histogram mode='all'>
  <word str='labour_cost' left='2' right='2' />
  <word str='TRAIL:1' left='2' right='2' />
  <word str='LABOUR_COST' left='1' right='1' />
 </histogram>
```

This Student (Left)	Another Student (Right)

`select a.description, b.fabric, \_`
`b.colour, b.pattern`
`from garment a, material b,`
`quantities c,order_line d`
`where a.style_no = c.style_q`
`and c.style_q = d.ol_style`
`and d.ol_material = b.material_no`
`and (a.labour_cost+c.quantity*b.cost)`
`> (select g.labour_cost from`
`garment g where g.LABOUR_COST > 80);`
	`select a.description, b.fabric, \_`
	`b.colour, b.pattern`
	`from garment a, material b,`
	`quantities c,order_line d`
	`where a.style_no = c.style_q`
	
	`and c.style_q = d.ol_style`
	`and d.ol_material = b.material_no`
	`and (a.labour_cost+c.quantity*b.cost)`
	`> (select g.labour_cost from`
	`garment g where g.LABOUR_COST >= 80);`

Figure 1: An example of reported plagiarism

with the *word* tag, with attributes showing the word in question, and how many times it appears in SQL statement 1 and how many times it appears in SQL statement 2.

The *word* tag also reports on inconsistent usage of words within each SQL, and how these are used consistently between the two statements being compared. Thus if both suspect queries used "AND" and "And" exactly once each, this would be flagged. This has proved to be particularly useful in plagiarism procedures.

Finally, *signature* looks for unusual patterns of the *space* character in the SQL statements. *op* looks for how spaces are used on each side of an operator (e.g. < and !=), *comma* does a similar thing for spaces around a comma, and *trail* looks for any spaces which appear right at the end of a line. Analysis shows that *trail* is often the crucial factor in confirming that an SQL statement has been electronically distributed; trailing spaces are completely invisible to the user, can happen by accident, and are easily copied between users.

3.4 Example

Consider Figure 1, where LEFT student has been detected plagiarizing with student RIGHT. The difference between LEFT and RIGHT is that RIGHT has a blank line in the middle, and the range on the last line is ">=" rather than ">". This cannot be detected using the *Equality* test. *Shuffle* does detect the copies, and gives it a similarity of 89%.

To strengthen the case, consider the histogram output. Here "labour_cost" is used twice in each query, and "LABOUR_COST" used once in each query. In addition invisible trailing spaces appear on the end of two of the lines in LEFT, and also appear in identical places in RIGHT. Certainly this is an electronic copy that has been disguised slightly by the students.

3.5 Validation

As part of the investigation, it was decided to attempt to validate the comparison process using a number of techniques. Firstly, cross-cohort plagiarism was identified (e.g. plagiarists who seemed to copy of each other yet studied in different groups or different years). Secondly, cases of plagiarism in the current cohort where a student appeared to copy more than a single SQL statement were taken through Napier's plagiarism procedures. All students who were taken through these formal plagiarism procedures were confirmed through that process to have plagiarized.

3.5.1 Cross-Cohort Plagiarism

Out of over 1000 student checked, only three groups of 2 students per group came back as breaching a cohort boundary. Two of these groups were traced to incorrect registration details (i.e. they actually were in the same cohort) and the last group was confirmed as two students who knew and helped each other.

4. EVOLUTION OF THE SYSTEM

The ActiveSQL site and its underlying systems have evolved over a number of years. Some of these changes were specifically implemented in an attempt to remove the need for students to plagiarise. Previously simple student-to-student plagiarism analysis code was used, and this appeared to show a decrease in plagiarism. However, this year it was considered that perhaps rather than reduce plagiarism, it had simply changed in nature, and was now in a form which was difficult to detect.

In this paper three cohorts are analysed in depth. These cohorts relate to year 2 students undertaking a degree in computing, with the module starting in 2002, 2003, and 2004. The following lists the assessment approach in each year:

- 2002: Students work on a minimum of 5 questions from a list of 15. They are awarded the highest 5 of the marks. Questions 1-5 were worth 12, 6-10 were worth 15, and 11-15 were worth 20 marks each.

  A significant degree of plagiarism was detected at the end of the module. In an attempt to discourage plagiarism the decision was made to switch to question banks. An incremental approach to assessments was also adopted in an attempt to improve student motivation.

- 2003: Assessment split into 4 stages. At each stage the student was presented with 2 questions randomly selected from a bank of 5 related questions. Each stage was worth 25%. The stages grow progressively harder.

  Plagiarism has decreased. However student feedback indicated that the jump between stages 2 and 3 was too great. The majority of students only completed stages 1 and 2. In an attempt to encourage progression into stage 3 the decision was made to give a mix of difficulty level questions in each stage.

- 2004: The stages were refined so that each stage consisted of 1 question from that stage's bank and 1 question from the previous bank.

5. Plagiarism Group Analysis

The initial step in this analysis was to visualise how students copied from each other. A visualizer was written, based of code available from [5]. By introducing question banks in 2002 it was assumed that students would find it difficult to identify friends with the same question as themselves.

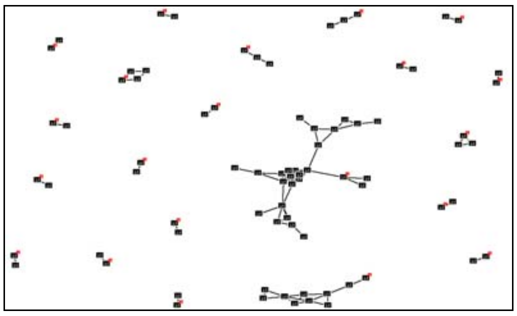

Figure 2: CO22001 Cohort 2002

Figure 2 shows a visualization for cohort 2002. The nodes are students who have some degree of plagiarism with another student, and the lines between nodes indicate who was involved in the plagiarism. The length of each line indicates the degree of plagiarism involved (shorter is more). This seems to indicate some significant groups of students who heavily copy from each other.

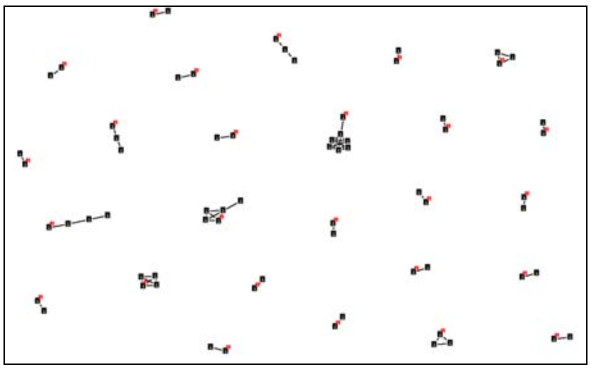

Figure 3: CO22001 Cohort 2003

Figure 3 shows the change to random question banks. Now large groups have been eliminated, but there are still some groups who copy heavily from each other. The plagiarism groups are tightly packed, indicated heavy copying has occurred. It is desirable to break these tightly packed groups.

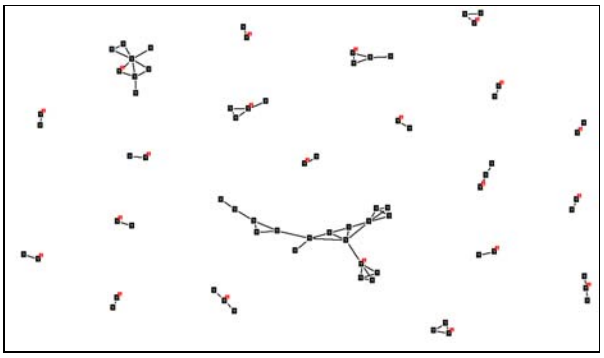

Figure 4: CO22001 Cohort 2004

In Figure 4, which has an adjusted difficulty distribution in the question bank selection mechanism, tight clustering is significantly reduced. Groups still exist, but nodes do not appear so tightly packed (the degree of copying was smaller). The number of groups has also decreased. It is possible that the now stretched-out big groups indicate that the number of questions in each bank is simply too small, and that slightly increasing the possible questions per bank will stretch these links to breaking point. However, adding to a question bank is non-trivial, as difficulty levels have to be maintained in each bank.

6. Analysis

The visualization has highlighted a number of issues worthy of further investigation. Long thin

	Cohort		
	2002	**2003**	**2004**
Total Number of students	312	301	263
Average number of questions attempted	5.7	4.6	5.8
Average Score	60%	51%	64%
Plagiarised questions	208	120	138
Percentage of dishonest answers submitted	12%	9%	9%

Figure 5: Statistics Gathered

associations suggest that the links are weaker than high-density groups. Perhaps these can be broken by better student education, more tutorials, or slightly more questions per bank. This will be looked at for 2005. It is also of interest to confirm the visual impression that random question banks have reduced plagiarism 2002-3, and that in 2004 more questions attempted did not lead to significantly more plagiarized questions.

Figure 5 shows statistics that confirm our initial visual assessment. Most importantly the degree of plagiarism has decreased with the use of question banks, and that peer pressure to attempt more questions did not lead to more plagiarism. Note that the *Average Score* in 2002 was originally calculated using a different marking scheme, but has been remarked here with the scheme used in the 2003 and 2004 cohorts.

7. CONCLUSIONS

Random question banks can be part of the solution to plagiarism. However the economics of having large banks of moderated questions of equivalent difficulty means that the bank size will likely be too small to eliminate plagiarism on their own. Large plagiarism groups can always defeat large question banks.

Visualization tools can assist system designers in identifying how plagiarism has changed, and in suggesting possible ways to improve their plagiarism detection algorithms. By studying such visualizations, it may also be possible to obtain a deeper understanding of plagiarism behavior. This will hopefully identify avenues to make plagiarism less attractive to students who are under pressure to meet their perceived targets.

Procedures to punish plagiarists should only be seen as a last resort. Even draconian punishments are not effective deterrents unless detection rates are high. The administrative costs of an academic conduct hearing are high, and they can be draining and dispiriting for all concerned.

8. REFERENCES

[1] Clough, Paul (2000), Plagiarism in natural and programming languages: an overview of current tools and technologies, Research Memoranda: CS-00-05, Department of Computer Science, University of Sheffield, UK.

[2] Heckel, Paul (1978). A Technique for Isolating Differences Between Files. Commun. ACM 21(4): 264-268.

[3] Russell, Gordon and Cumming, Andrew (2004). Improving the Student Learning Experience for SQL using Automatic Marking. In Demetrios Kinshuk and Pedro Isaias (Eds.), Cognition and Expolaratory Learning in Digital Age (CELDA 2004) pp 281-288. Lisbon: IADIS Press. ISBN 972-98947-7-9.

[4] Russell, Gordon and Cumming, Andrew. ActiveSQL: http://db.grussell.org .

[5] Shapiro, Alexander: Touchgraph visualiser project. http://touchgraph.sourceforge.net .

[6] M. J. Wise. YAP3 (1996). Improved detection of similarities in computer programs and other texts. In Twenty-Seventh SIGCSE Technical Symposium, pages 130--134, Philadelphia, USA.