USING PLAGIARISM DETECTION TOOLS IN ORDER TO CONTROL GROUP WORK SUBMISSIONS IN LARGE COURSES

R. Matos[1], A. Sousa[2]

[1] Shared Services (SPUP), University of Porto (PORTUGAL)
[2] INESC TEC (formerly INESC Porto) / Faculty of Engineering, University of Porto (PORTUGAL)

Abstract

Peer education and teamwork are looked upon as important educational tools due to the massification of higher education programmes. As such, teams frequently produce several deliverables throughout a given course, and these should be properly organized and assessed.

As the deliverables submission process often occur in large numbers, any attempt to manage the procedure too tightly is bound to make the evaluators spend a lot of time just checking, controlling and correcting each individual submission; assessing these vast numbers of deliverables, is another problem, which is simply too vast to tackle in this article. Nevertheless, we do use plagiarism detection tools which offer invaluable information to evaluators. These mentioned tools are also used in an unsuspecting way: while allowing the students to submit the deliverables in a loose way, these tools can be used to check, and control any wrongdoing with little effort.

This article shows the details of the implementation and the lessons learned. The prototype was tested in a course which has roughly one thousand students enrolled in, and that had effectively produced a course-portfolio of an organized set with all deliverables in an (almost totally) automated manner.

Keywords: Large courses, Deliverable Submissions, Plagiarism, Course Portfolio.

1 INTRODUCTION AND CONTEXT

Teaching large courses presents several challenges that are becoming frequent in current days. One of those, is unquestionably the amount of time spent on managing submissions of deliverables. While it is possible to reduce the amount of deliverables by promoting team works, often the time spent in managing the submissions process is not reduced as one could expect.

Nevertheless, “team works are also currently essential to all Engineering degrees and are thought of as an interesting method to promote soft skills and peer learning. Team works are also essential for coping with large courses because otherwise the (manual) evaluation of submissions would be unthinkable.” [1]

The work presented here is the follow-up of a previous effort [1]. Both works were inspired in a single very large, cross program course named “Projeto FEUP”, at Faculty of Engineering of the University of Porto, Portugal [2]. The test occurred in the first (fall) semester of school year 2014/15 and this time, the course had just over 1000 students, split into 150 groups of 6 or more students.

The prototype implemented in our first approach, that used Moodle Learning Management System (LMS) as infrastructural support, gave us a clearer idea of what was useful to have and what was not at all practical.

Although the prerogatives that we followed in the previous work were still applicable, we started wondering if we could discard the operational costs (due to installation, maintenance and licensing) of having an entirely different piece of software, thus abandoning the idea of giving the users the benefit of familiarity with the LMS interface.

As this idea grew, we realized that the software needed to manage submissions made by teams in order to generate an organized list of all submissions was not at all difficult to implement in a simpler way. Thus making it also easy to transform this list into a course portfolio.

2 PROBLEM STATEMENT

Although LMSs (and in our particular case, Moodle) are essential to support large courses like the ones mentioned we tried to simplify the process as much as we could - both for the submitters and for the professors that will grade submissions.

2.1 Needs

As mentioned, the test course has 150 groups and each team submits 3 deliverables at the end of the course, while some other activities are individually graded.

Likewise, as mentioned before and on the previous work, the aim is to make the production of the course portfolio easier, and less managing time consuming, but still maintaining an adequate listing of the deliverables (turned in or missing), respective on-schedule status and other relevant observations.

2.2 Requirements analysis

The following requirements were established for submission of deliverables:

- Support for large number of students
- Integration with school's information system via an authentication mechanism
- Flexible number of students over teams
- Changing groups over time
- Changing teams along the evaluation period
- Safe, confirmed submissions of (large) files whilst keeping the uploader's identity, timestamp, etc.
- Instead of restricting allowable names to a given list as we did in the previous prototype, we chose to rename any file automatically thus avoiding misspelling errors
- Limit the deliverable file format type to PDF.

It would also be interesting to have as little technical dependencies as possible, that is, preferably not to use databases thus keeping for example, backup process as easy as possible.

The overall organization of the system has to make it simple to access files for the graders or to do bulk operations on all files, for example, it has to be easy to run plagiarism detection on all files.

2.3 Views for stakeholders (actors)

When taking into consideration large courses with several submissions, it is most likely interesting to consider that each interested party (stakeholder) will be interested in a different, specific view of the listing of the submissions.

Although we had that in mind, we opted for granting open access to all submissions to all students. Moreover, a student could overwrite any other student's submission.

This approach proved to be amazingly efficient, mainly due to the social pressure between the peers.

No student overwrote another's work either willingly or by accident (which we could easily confirm by analyzing what was recorded in the log system implemented). Furthermore, students organized themselves, thus saving the evaluators an huge amount of time that would otherwise have been spent in simple managing tasks.

2.4 A tendency to produce low scientific quality information

Nowadays, search engines – such as Google or Wikipedia - and blogs, tweets, wikis etc. provide easy access to information that just a few years ago would have been unthinkable. But there is some pernicious side effects that emerge from the generalization of repeated information (shared, re-tweeted…). First, this kind of information is often unverified thus making it of very poor quality as a source of information. Second, the sense of Internet freedom along with the lack of previous consciencialization leads to lack of citation knowledge. This also may come from the restricted access to high quality scientific journals, that drive common users away. Often, they are not even aware of the lack of ethics involved in a simple copy-paste without providing a correct reference to the original

source. This recurrent plagiarism phenomena results - as one might expect - as ultimate consequence, in the very poor scientific quality of new information produced.

3 STATE OF ART AND BRIEF MARKET REVIEW

It's easy to find plagiarism detection tools and methods, either paid or free. Many can be downloaded and used independently, many others are provided as an online service, thus offering several usage cost models [3] [4]

Some of these tools rely on search engines such as Google subcontract findings - this is indeed a viable strategy but frequently inefficient as currently Google limits search to 32 words.

But plagiarism checking is not of widespread use, neither manually nor in automated ways. There is an astonishing absence of voluntary checking on behalf of the authors for instance before submitting works or dissertation or technical report.

Moreover, there is a widespread technical difficulty amongst students to recognize correct citation, make correct transcription, and defining correct ethics.

4 SUBMISSION PLATFORM WITH PLAGIARISM CHECK

Overall, this work presents strategies which allow for an easier interaction in submission, consultation, management and plagiarism detection in massified courses.

Our previous prototype presented some technical issues that were not completely solved. On top of that, the managing efforts were still needed for some tedious tasks.

For this reason, we opted for a completely loosely coupled submissions systems, that did not need a database (DB), but could use an authentication purposes such as for example LDAP. On top of that, the ability to define students groups, and the name of the deliverables was seen as a priority, alongside with the automated plagiarism detected upon submission.

4.1 Starting point

Our initial prototype based on Moodle showed a lack of versatility that crippled its expansion. It showed several data access limitations (mainly due to security reasons), thus having as main weakness the lack of flexibility of its usage and the impossibility of changes performed in our Moodle infrastructure production environment that is tied to a campus wide scope, where stability and quality of service provided is the major concern.

4.1.1 Background

Plagiarism detection tools have commonly some basic concepts involved in their genesis. [5] To better understand the methods for creating the tools, a non-exhaustive list of relevant terms is presented:

Token: a group of characters (or words), sometimes also named *n-gram*, where n denotes the number of characters per token

Tokenization: is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens [6]

Hash: the result of a function that computes a sequence of characters into a number. Usually, this is done to speed up comparisons.

4.2 Cross File Validation

Nevertheless, the probability that errors when submitting deliverables could go unnoticed was something that worried us. Since we needed to cope with the volatility in courses' team making, we could expect that several students could send the same or similar copies of the same deliverable as members of different groups.

For this reason, we came up with the idea of using plagiarism tools to catch similar submissions of deliverables, stressing that it worked as a managing helper, and not as an ethical enforcer.

In an initial stage we consider using a fingerprint or Winnowing algorithms, which are usually very good candidates for document-matching. [7]

Instead, because of the sheer amount of documents to compare, we opted for a much more crude approach, but faster method, using only a Normalized Compression Distance function

$$NCD_Z(x,y) = \frac{Z(xy) - minZ(x), Z(y)}{maxZ(x), Z(y)}$$

being Z(x) the binary length of the file x compressed with compressor Z (for example “gzip”, bzips2”) [8], ignoring completely the improvements that could arise from more complex algorithms. Simply put, one way of interpreting the results of this function, is if two texts possess a NCD value less than 30% they can be considered as work of plagiarism.

This function can be easily implemented in PHP like this:

```
<?php
function ncd($x, $y) {
   $cx = strlen(gzcompress($x));
   $cy = strlen(gzcompress($y));
   $result=(strlen(gzcompress($x . $y)) - min($cx, $cy)) / max($cx, $cy);
   return $result;
}
?>
```

Although this method proved to be quite fast, and more than enough for our requirements, because of the large size of the submitted deliverables (and since we were using PHP’s file_get_contents which fetches the entire string of data into a variable and stores it in the hosts memory, thus creating some problems) we opted to use instead a plagiarism detection command line tool called sherlock, thus making our system similar in functionalities to the BOSS Online Submission System [9],[10].

All considered we only had to:

1. convert all submitted PDF’s to plain text and store all converted files in temp folder
2. make some sanity checks:
    1. check if original file is really an PDF (by confirming if the file has the initial characters “%PDF-”)
    2. check if the converted plain text file is larger than 0 bytes (that could occour if the conversion from PDF to TXT had failed.)
3. run: sherlock *.txt in temp folder
4. print the result of 3.

since we could easily use sherlock via an PHP shell_exec call. Although we had started by using a PHP class for extracting the text content out of the PDF deliverables submission files, in the end we opted by using a similar solution that was used for sherlock, that is:

```
<?php
function pdf2text( $pdf ) {
   $output=shell_exec("/usr/bin/pdftotext ".$pdf." - 2>&1");
   return $output;
}
?>
```

All the methods above, provided us with a very efficient method to assemble a similarity matrix [11] between all submissions deliverables.

The proposed approaches allow us to verify intentional or not “similar copying” of files among groups but the actual identification of the reason needs more information (or manual work).

4.3 Automated or Bulk Sending to plagiarism tool

Several approaches are possible to take advantage of the organization described.

The approach taken is to bulk submit to the “Turnitin” tool that, in turn, identifies and highlights plagiarism issues.

After completion of the previous step, it is possible to recover global similarity data in each file and such data may be publishable or may be sent to submitters and graders.

Algorithm for sending feedback to authors / submitters:

1. Upon each submission of the “Report”
2. Recover submitter email list and grader email from the 1st page of the report (this is imposed in the mandatory template of the course) - this is done by the pdf2text tool and regular expression matching
3. Use web scraping to get each URI from the Turnitin visualizer / highlighter
4. Use curl tool to get indicators for the submitted file
5. “Prepend” information to a private text file (URI, plagiarism indicators, etc)
6. Generate automated email to submitters with the previous file

When submission deadline is crossed, automated email for the graders may be generated with relevant information such as URI for the file(s) of the group, numbers regarding plagiarism indicators, URI for Turnitin visualizer and file describing history of submissions (etc.).

5 STUDY STRATEGY USING TURNITIN AND “PROJETO FEUP”

While we were working on this project, some questions arouse in our minds.

- Would it be possible to better manage large number of submissions from large courses?
- Would it be possible to assist an author to detect and improve plagiarism and citation practices via quasi-real-time feedback?

Given the previous experience in organization of large courses, this article used a case study of the “Projeto FEUP” course, which is a cross program course named “Projeto FEUP” for the 1st year students, at Faculty of Engineering of the University of Porto, during 2 editions.

After analysing a comparison between sherlock and Turnitin [12], we decided to use it as well, to see if our perception of the data was consistent with the results provided by sherlock.

The results differed a little, as one might expect.

5.1 Turnitin experience

Some of the informations provided by Turnitin are more useful for large amounts of documents analysis, if we could extract the data (e.g. to a spreadsheet or statistical analysis package). For that, we needed to rely on web scraping [13] techniques (with PhantomJS [14] or similar tool, since Turnitin relies heavily on JavaScript), to extract data programatically, since there is no export method builtin that fully satisfied our needs. However, this method makes it possible to combine Turnitin's results with other tools, opening the way to easily identify duplicate submissions, identify excessive copy and paste from the Internet and plagiarism from submissions of previous editions of, for example, same course in bulk mode.

6 CONCLUSIONS AND FUTURE WORK

The results shown in the previous section demonstrate that even if plagiarism and citation ethics are explicitly addressed in a course, facing short deadlines, many students turn to fast apparent completion methods (regarding citation ethics as a luxury they can not afford).

This practice can only be reduced by having students see adequate plagiarism feedback that further proves the concerns of the institutions involved. This should be as easy as possible and as clear as possible for authors, graders and all of the scientific community.

A test sample of 147 submissions made in the 'Projeto FEUP' was sent to Turnitin. This test case course is an initial course at FEUP, in Portugal and explicitly addresses plagiarism issues. The results without plagiarism feedback hint concerns that many students fail to comply plagiarism ethics - about 20% of the submissions are likely to have 25% to 49% of material similar to what was found in the internet and in the search databases of the used tool.

The article also suggests an adequate system for submissions that can easily lead to quasi real time automated feedback to the authors, in this case, by using the Turnitin tool. The student / author can use this information to improve the submission.

Upon the final delivery date, plagiarism information is also to be sent to the graders.

The authors expect to implement such strategy in the next edition of the course.

REFERENCES

[1] A. Sousa, R. Matos, "Managing Team Work Submissions in Large Moodle Courses in Order to Generate Course Portfolios", in Information Systems and Technologies (CISTI), 2013 8th Iberian Conference on , vol., no., pp.1,6, 19-22 June 2013 © IEEE, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6615797&isnumber=6615699

[2] Institutional page of "Projeto FEUP" course at university's information system: https://sigarra.up.pt/feup/pt/UCURR_GERAL.FICHA_UC_VIEW?pv_ocorrencia_id=353321 (Jan. 2015)

[3] http://elearningindustry.com/top-10-free-plagiarism-detection-tools-for-teachers (Jan. 2015)

[4] http://www.grammarcheck.net/review-10-sites-that-check-for-plagiarism/ (Jan. 2015)

[5] V. T. Martins, D. Fonte, P. R. Henriques, D. Cruz, "Plagiarism Detection: A Tool Survey and Comparison", http://drops.dagstuhl.de/opus/volltexte/2014/4566/pdf/14.pdf

[6] http://en.wikipedia.org/wiki/Tokenization_%28lexical_analysis%29 (Jan.2015)

[7] http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6574560 (Jan.2015)

[8] http://en.wikipedia.org/wiki/Normalized_compression_distance (Jan.2015)

[9] http://www.dcs.warwick.ac.uk/boss/ (Jan.2015)

[10] http://sydney.edu.au/engineering/it/~scilect/sherlock/ (Jan. 2015)

[11] http://en.wikipedia.org/wiki/Similarity_matrix (Jan. 2015)

[12] http://www.cscjournals.org/manuscript/Journals/IJCL/volume3/Issue1/IJCL-33.pdf (Jan. 2015)

[13] http://en.wikipedia.org/wiki/Web_scraping (Jan. 2015)

[14] http://phantomjs.org/ (Jan. 2015)