

# Customising a Copying-Identifier for Biomedical Science Student Reports: Comparing Simple and Smart Analyses

Julia Medori<sup>1</sup>, Eric Atwell<sup>1</sup>, Paul Gent<sup>2</sup>, and Clive Souter<sup>3</sup>

<sup>1</sup> School of Computing, University of Leeds, Leeds LS2 9JT, England  
{medorij,eric}@comp.leeds.ac.uk  
<http://www.comp.leeds.ac.uk/nlp/>

<sup>2</sup> School of Biomedical Sciences, University of Leeds, Leeds LS2 9JT, England  
J.P.Gent@leeds.ac.uk

<sup>3</sup> Centre for Joint Honours in Science, University of Leeds, Leeds LS2 9JT, England  
D.C.Souter@leeds.ac.uk

**Abstract.** The aim of our project is to develop a system for detecting student copying in Biomedical Science laboratory practical reports. We compare contrasting approaches: “simple” methods Zipping, based on a standard file-compression tool, and Bigrams, a basic comparison of frequent bigrams; and “smart” methods using commercial-strength plagiarism-checking systems Turnitin, Copycatch, and Copyfind. Both approaches successfully flag examples of copying in our Test Corpus of 218 student courseworks, but Copycatch provides more user-friendly batch-processing mechanism. Human experts go beyond word-based pattern matching, and take account of knowledge specific to our domain: methods and questions can legitimately be copied, whereas originality is more important in the Discussion section.

## 1 Introduction

The aim of our project is to develop a system for detecting student copying in laboratory practical reports, customised to a specific genre/subject, in our case initially Biomedical Science first-year reports. We are aware of many commercial plagiarism detectors, but Biomedical Science teaching staff believe these generic systems are generally too sophisticated or complicated for this specific problem.

We have collated a Test Corpus of example student reports in the specific genre, to identify characteristic features. We have also interviewed Biomedical Science teaching staff to elicit/confirm significant diagnostic features which can identify copying, and identify the level of copying considered unacceptable (as some overlap in lab reports is expected).

Unlike existing surveys of generic plagiarism detection systems, we have tried to evaluate systems against the detailed specific requirements identified above. A number of candidate systems were considered, including:

- CheatChecker [1] – an n-gram based tool
- SIM [2] and YAP [3] based on longest common sub-sequence approach.
- Detecting “unusual” similarities between a closed set of essays (or programs) is called collusion and the basis for Malcolm Coulthard and David Woolls’ Copycatch tool [4]
- Copyfind [5] is an example of a system aimed at scientific rather than humanities documents
- Turnitin is probably the most widely-known system, but the website [6] gives scant details of the underlying methods
- Clough [7] is investigating more sophisticated Natural Language Processing techniques.

## 2 Test Corpus of Biomedical Science Student Laboratory Reports

Evaluation using a test corpus is a well-established methodology in Natural Language Processing research, and is appropriate to this task: in our case, the aim is to test candidate systems and compare detection (and “red-herring”) rates. The initial test corpus used in our study was composed of 103 Biomedical Student reports. As there turned out to be few examples of ‘real’ plagiarism in our initial corpus, we extended it by adding 94 reports written by second year students, and by artificially generating 19 partly-copied documents from the original reports. The first year assignment described the experiment and gave instructions on how to use the computer simulation to generate results; then the students had to produce tables and graphs and answer a few questions to analyse the results. It seemed likely that any plagiarism we would find would be in the text of the answers. The second year students had to report on 3 different experiments. Unlike the first year students reports, these students didn’t have to answer specific questions; the text should be more ‘free’ and show more originality. However, they still follow the same structure: Introduction, Methods, Results, Discussion.

To check the sensitivity of each plagiarism detection software tested, we created 19 new files: each composed of one file from which 5%(then 10%,..., 95%) is removed and replaced by a portion of another file. We also added in our corpus one example of plagiarism that had been detected previously.

A characteristic of this genre is that a lot of the vocabulary (being mostly from the science domains) will be shared between all the reports. The answers being short, not a lot of originality will be noticeable for each student. As most of the plagiarism checkers available at the moment are developed to detect plagiarism either in humanities essays (high level of originality) or in programming (with limited vocabulary), there is no existing copying checker for the specific problem of scientific reports: low level of originality but no limited vocabulary.

Note that the names throughout this report are made-up and are not the real names of the students.

### 3 Zipping, a Simple Compression-Based Approach

As a baseline of our study, we will test a method inspired by the article [8]. This article suggests that using file compressors or zippers, we would be able to measure a ‘distance’ between files according to their similarities.

The algorithm of common zippers (like gzip) finds duplicated strings and marks up the matching strings with a pointer; which means that, by appending a questioned text to different corpora (e.g. of different authors or languages) and measuring the difficulty of zipping these files, we can measure the level of similarity between two files. The smaller the zipped file of the concatenation, the more common elements the questioned text shares with the corpus.

### 4 Turnitin: A Commercial System

The first commercial system we used is Turnitin.com, a web-based plagiarism detection service first designed to detect material cut and pasted from the Internet but also detects similarities between submitted papers; Leeds University has subscribed to the service [10]. The submission of papers is done by means of laborious cut-and-paste of the text from each document. After about 24 hours, an Originality Report TM is sent onto the user’s account for each submitted paper.

We found that TurnItIn gives quite good results. However, the fact that it gives an index of similarity between the file and all papers found on the web and in the database leads it to miss a few cases of plagiarism, especially in the case of the pair ‘Colin-David’. Turnitin gives as a result the level of similarity between the submitted paper and all the papers in the database and the data found on the web; it seems a waste of time checking on the web and other papers than the ones in the class. A more significant shortcoming is the laborious input method. On the positive side, Turnitin underlines the similarities, making them easier to visualise; and it makes it easier to detect partial plagiarism.

### 5 CopyCatch

These conclusions lead to the next step of our study, which would be to test CopyCatch, a software developed by Woolls from CFL Software Development. After reading the JISC report [9], it seems that it may be more appropriate to our specific needs as it gives instant results, sorting them by percentage of similarity between 2 files.

The main characteristic of this software is that it is easy to use, especially the submission process: it allows the user to browse and select the files to check. The results are almost immediate. It outputs a list of pairs of files sorted by percentage of match between them. It is then possible to have a list of the vocabulary or phrases shared between the 2 files and it can also mark up the files highlighting the similarities between them. It allows the user to check Word documents as well as text, rtf and HTML files: no previous conversion is needed.

CopyCatch finds without difficulty all our ‘real’ cases of plagiarism. It even finds the pair ‘Colin-David’, which was not an obvious case of plagiarism and which we would have missed using TurnItIn. The selection of our files in a few clicks is a very important practical advantage.

## 6 CopyFind

As another alternative, we decided to test another CopyFind [5], available for Linux as well as for Windows. This software requires that the Copyfind executable and all the files to be checked are in the same folder. The user has to add a list of all the files that have to be checked. These can be Word documents. The user can define a threshold of similarity. The system will then output a file containing all the pairs of files and the number of matched words for each of them. An HTML report will then be created for every pair which are above the threshold, with underlined similarities.

We ran the program with standard parameter values; the results are quite good, but it doesn’t detect the pair ‘TestFile-TestFile2’. To find this pair in the results, we have to lower the threshold by changing the minimum number of matching words to report from 500 (value suggested) to 300. The modification of this threshold doesn’t change any of our other results.

CopyFind is a program that works quite well and gives a clear output underlining the similarities between the files. The results are immediate, but its submission method is cumbersome.

## 7 Another ‘Simple’ Approach: Bigrams

This method is a ‘home-grown’ method based on the idea that when 2 files are similar, they should share a high number of their bigrams. The programs are all implemented in Perl. We tested this method first with character bigrams and then, word bigrams. In the first place, we create a model for each file in our corpus by counting all the occurrences of each character bigrams in the file. To compare two files, we simply see whether the most frequent bigrams are the same in both files. We compare the results for a model containing 10, 20, or 30 most frequent bigrams. The results even for “top 30” character bigrams were disappointing: this method will flag copying of an entire document, but below a total match, most files share a large proportion of character bigrams.

We tried the same experiment with word-bigrams, and got better results: word-bigrams are better discriminators.

## 8 Result Comparison between the 5 Experiments

Throughout our study, we discovered several occurrences of plagiarism in our corpus; These are listed in Table 1. In this table we show which system was able to point out each case of plagiarism. It is considered as positive (marked with

**Table 1.** Comparison of results (key: ‘+’ = detected, ‘-’ = not detected)

Known cases of copying	pointed out by				
	Zippping	TurnItIn	CopyCatch	CopyFind	Bigrams
Albert 1-2	+	+	+	+	+
Barbara 1-2	+	+	+	+	+
TestFile - TestFile2	+	+	+	+	+
TestFile - SourceA	-	+	-	-	-
TestFile - SourceB	-	+	-	-	-
TestFile2 - SourceA	-	+	-	-	-
TestFile2 - SourceB	-	+	-	-	-
Geraldine - Fred(ass2)	+	+	+	+	+
Jane - Naomi(ass2)	+	+	+	+	+
Colin - David(ass2)	+	-	+	+	+/-
Jane - Naomi(ass3)	+	+	+	+	+
Constructed plagiarisms	up to 20%	up to 30%	up to 30%	up to 20%	
ScanFile1 - ScanFile2	+	+	+	+	+

**Table 2.** Usability of the five systems tested: submission and output

systems	Usability	
	submission	output
Zippping	hard-coded into the program	sorted list of pairs by value of ‘distance’
TurnItIn	Copy+Paste documents individually	Report with similarities highlighted
CopyCatch	select from file manager window	Statistics+ sorted list of pairs by % of match + files side by side with similarities highlighted
CopyFind	make a list of files	HTML reports for each pairs above threshold with underlined similarities
Bigrams	give directory name	Sorted list of pairs

a ‘+’ in Table 1) if the paper was found at the top of the list of results. In the case of the pair ‘Colin-David (Assignment2)’ in the word-Bigrams experiment, the pair was only shown within the first seven results after three false-positives. Therefore, it was marked with a ‘+/-’ as it was detected, however, whether this pair of assignments would be manually verified would depend on the amount of time given to the task by the user. If the plagiarism is detected considerably down the list of results (where there is little chance it would be checked by a user manually checking every script in list order) it is marked with a ‘-’ in Table 1.

## 9 Discussion

In the tests reported above, both approaches successfully flagged examples of “blatant” copying, but results were less clear-cut when reports had small amounts of overlapping text. Results based on our Test Corpus indicate that we need to

take account of knowledge specific to our domain. In Biomedical Science laboratory reports, students must always write four sections: Introduction, Method, Results, Discussion. In fact this is standard practice even for research journal papers; and this format is a mandatory part of the course work specification. Biomedical Science lecturers know that certain parts of the text can legitimately be copied (the given specification of Method, and specific questions set by the lecturer), whereas originality is more important in other parts (notably the Discussion section). Ideally our copying-checker should include some “intelligence”, to make it ignore overlap in unimportant sections, and focus on the Discussion section; overlap in Discussion is a much surer indicator of copying.

We have analysed the requirements of our specific problem, through discussion with Biomedical Science teaching staff. We have surveyed a range of candidate systems, and evaluated five systems (Turnitin, Copycatch, Copyfind, and two “home-grown” systems, Zipping and Bigrams) in experiments with a Test Corpus of Biomedical Science student laboratory reports. We concluded that none of the systems stood clearly above the rest in their basic detection ability, but that Copycatch seemed marginally more successful, and more significantly it had the most appropriate user interface for our needs as it ran on a local PC and simplified the analysis of a whole directory of lab reports. Following personal contact with Copycatch developers, we agreed to be the first beta-test site for the enhanced Java version. We installed Copycatch on Biomedical Science computing network for future use by staff. A side-effect of availability and use copy-checking software is the need to migrate from paper-based report submission and processing to electronic submission, probably via student pigeon-holes in the Nathan Bodington virtual building. The School of Biomedical Sciences has decided to require electronic submission of all laboratory reports in future, to facilitate use of copying-detection software. During tests, we found evidence to support suspected cases of copying, and discovered at least one new (previously unknown) case of student copying, which was reported to the appropriate authority.

## References

1. CheatChecker: <http://www.cse.ucsc.edu/~elm/Software/CheatChecker/>
2. SIM: <http://www.few.vu.nl/~dick/sim.html>
3. YAP: <http://www.cs.su.oz.au/~michaelw/YAP.html>
4. CopyCatch: <http://www.copycatch.freemove.co.uk>
5. CopyFind: <http://plagiarism.phys.virginia.edu/software.html>
6. Turnitin: <http://www.turnitin.com/>
7. Clough, P.: <http://www.dcs.shef.ac.uk/~cloughie/>
8. Benedetto, D., Caglioti, E., Loreto, V.: Language Trees and Zipping. *Physical Review Letters*, Vol. 88,4 (2002)
9. Bull, J., Collins, C., Coughlin, E., Sharp, D.: Technical Review of Plagiarism Detection Software Report. JISC (2001)
10. Wassall, Terry: On-line Plagiarism Detection Projects 2001-2002. FLDU Technical Report, University of Leeds (2002)