TEXT DOCUMENT PLAGIARISM DETECTOR

Lukáš Kořínek
Master Degree Programme (5), FEEC BUT
E-mail: xkorin12@stud.feec.vutbr.cz

Supervised by: Petr Petyovský
E-mail: petyovsky@feec.vutbr.cz

Abstract: This paper provides an overview of diploma thesis concerned with research on available methods of plagiarism detection and then with design and implementation of such detector. Primary aim is to detect plagiarism within academic works or theses issued at BUT. The detector uses sophisticated preprocessing algorithms to store documents in its own NoSQL corpus. Implemented comparison algorithms are designed for parallel execution on graphical processing units and they compare a single subject document against all other documents within the corpus in the shortest time possible, enabling near real-time detection capabilities.

Keywords: text processing, parallelization, CUDA, NoSQL, C++

1 INTRODUCTION

Writing a paper, thesis, or simply any text available to general public demands that every idea coming from a different author has to be properly cited according to a widely accepted norm (e.g. ISO 690, [2]). Lack of citations deem the text to be a product of process known as plagiarism. Occurrence of plagiarism is considered as a serious ethical issue, especially within grounds of the academia.

Therefore, it is vital to prevent releasing plagiarised texts to avoid any future scandals. A specialized software, plagiarism checker/detector, may help with the prevention, as its main function is to examine various textual bases and to measure if and how much are examined texts similar to each other. In general, plagiarism detectors can be based on one of three main principles. Intracorpal systems compare selected text against a well-known finite set of documents. Extracorpal systems search in open databases, typically utilizing web search engines. Intrinsic systems analyze stylistic features of a single text and try to find differences within its parts. A plagiarism detector may compare original fulltexts or data extracted from considered documents.

The plagiarism detection task itself is not an exact discipline and any of the approaches always results in a compromise between practical feasibility, speed of processing, quality of inference and flexibility of usage on various text sources. The essential part is human interaction with the system, because any of the findings (matches between documents) have to be evaluated by a human to avoid unfair accusations due to occurring false positives.

The diploma thesis designs and then develops a new plagiarism detection system for the BUT, where it should be used to compare submitted theses. Its design uses findings from some of the previous papers (see [4], [5]) to construct an optimal detection algorithm deliverable as a complete out-of-the-box solution, not as just a prototype. Main requirements are high speed comparison, acceptable quality of output and integrability into existing web infrastructure. The ultimate goal is to enable on demand plagiarism checking. Challenge in designing such a system is in the volume of data which has to be processed, because the expected database of documents (corpus) may have up to hundreds of thousands of documents. This scale requires parallelization on both the processor and graphical cards. Available system resources have to be handled as effectively as possible.
2 SOLUTION OVERVIEW

Implemented system is an intracorporal plagiarism detector that always compares a single selected document against all other documents within its own corpus.

The system reads texts in PDF format and BUT theses identified by their ID (available on an internal API). Document indexation (addition into the corpus) requires each text to pass a series of eleven pre-processing steps. Data formed by the preprocessor is then suitable for fast comparison. Preprocessing involves text parsing, removal of cited sections and words without informative value, tokenization (splits the text into a list of words) and lemmatization (replaces words with their uninflected form). Not only document content gets saved, but also other relevant metadata, like timestamps, information about its author and page counts.

The processed document contents are stored as so called n-grams, or groups of N adjacent words forming a sequence based on the original text. These n-grams are represented as numeric indices into n-gram tables persisted within application memory. Viability of the described approach had been previously tested in thesis [4]. Usage of numeric values for the n-grams improves detection performance, because comparison of numbers is less compute-intensive than comparison of strings.

Illustrative text and a corresponding n-gram table record (2-grams).

Detection task (document comparison and matches extraction) is highly parallelized and the selected text gets compared against batches of up to thousands of documents. One after another, batches are loaded from the corpus and searched for matches. Obtained matches from every batch are then joined together. Furthermore, compared documents in batches are divided into multiple smaller chunks to further speed up match searching.

Matches extraction involves comparison of documents’ n-grams and a subsequent clustering based on a min-max distance metric. Matched clusters (typically on a scale of paragraphs) are then filtered by thresholding both their size and equality of their contents. Line and page markers are used to locate the match within original document fulltexts. Resulting clusters are grouped by compared documents and presented to the user. A web-based graphical user interface is also included.

Figure 1: Graphical user interface of the detection system.
3 SOFTWARE ARCHITECTURE

The plagiarism detector solution includes following components:

**Application service**  The service makes a key element of the solution. Written in C++, it provides an interface for all implemented functions of the detector, like document indexation and match searching. It is also responsible for management of available resources. Output of the service can be obtained in either line-delimited JSON objects or a human readable structured text. Parameters of the service are configurable in a standalone file.

**API client**  The client is a slim web application written in NodeJS. It provides a REST API and a web interface for tasks run by the service. Communication between the client and the service is based on TCP sockets. The expectation is that a user or supersystem will communicate with the service solely throughout this client.

**Document corpus**  NoSQL MongoDB database containing indexed documents and their metadata.

**Additional data memory**  Also handled by MongoDB, persists data critical for indexation tasks.

All the functions of the application are available in form of callable Tasks. Default thread of the application performs only task scheduling, while every called task gets executed on its own worker processor thread. According to their needs, specific tasks may then utilize creation of new subtasks or graphical processing unit threads based on Nvidia CUDA platform. The graphical threads are used not only for the match searching, but also for some of the preprocessor steps.

In order to standardize hosting environment and installation of the solution, application’s components have been containerized facilitating *docker* and *podman* services. The application has been designed to run on a UNIX-based operating system with an Nvidia graphics card and a multicore processor.

4 TESTING

A set of 11 documents has been created for qualitative testing of the detector (forming a so called annotated corpus). These documents contain both copy&paste plagiarisms (identical copies) and paraphrases. Comments attached to the documents annotate source and placement of plagiarised sections.

Apart from this set, a corpus of real BUT theses from years 2018 and 2020 has been indexed. In total, for testing there were 4,498 documents consisting of almost 9 million unique n-grams (2-grams; preliminary tests on longer n-gram variants seemed to perform worse in detection of paraphrases). Length of the documents varied between 1 and 300 pages. The comparison was performed in 10 batches, each of them utilizing 32x256 graphical threads (optimal combination found for testing hardware Nvidia GTX 1660S).

<table>
<thead>
<tr>
<th>document</th>
<th>page count</th>
<th>time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>samples/sample_2.pdf</td>
<td>37</td>
<td>18</td>
</tr>
<tr>
<td>samples/sample_1.pdf</td>
<td>56</td>
<td>29</td>
</tr>
<tr>
<td>internal://2018:112346</td>
<td>158</td>
<td>65</td>
</tr>
</tbody>
</table>

**Table 1:**  Time-oriented test results.

Found matches fully cover copy&paste plagiarism in the annotated database, most of the paraphrases and a noticeable number of false positives. The table above doesn’t show found match counts, because their inclusion would be meaningless without more details. Improvements in indexation and optimalization of threshold parameters would help to eliminate some of the false positives found.
5 CONCLUSION

A new software solution for plagiarism detection has been designed and implemented. The solution is suitable for integration into BUT systems. Deployed system may replace or accompany an existing solution used at the moment of writing this paper. A possible arrangement might be, that a detection on newly turned-in theses would execute two plagiarism detection softwares in series, where one of them is faster and the other one has better result quality.

However, the solution is independent on university infrastructure and can be applied to any textual match searching task in situations where a finite corpus is known and the languages concerned use the latin script. The solution is delivered in form of a container cluster, enabling it to be deployed in a cloud environment.

As for the next steps, further speed up of the match searching algorithm would make the system usable in an on-demand basis. To achieve that, brute force approach in bringing multiple and more powerful graphical processing units, or memory access optimalizations (enabling maximalization of threads run on every graphics card) may both be used. Improvements in preprocessing steps’ selectivity and match searching process parameters would help eliminate most of the false positives.

In the future, an extension of the application searching for keywords in online sources is possible. Inclusion of web pages in the corpus would increase the likelihood of a successful match for plagiarisms which do not come from other theses, but from texts available on the web.

REFERENCES


