Document Overlap Detection System for Distributed Digital Libraries

Krisztián Monostori, Arkady Zaslavsky, Heinz Schmidt
School of Computer Science and Software Engineering
Monash University, Melbourne
900 Dandenong Road, Caulfield East, 3145
Australia
Tel:+61-3-9903-{1410,2479,2332}
E-mail: {krisztian.monostori, arkady.zaslavsky, heinz.schmidt}@infotech.monash.edu.au

ABSTRACT
In this paper we introduce the MatchDetectReveal(MDR) system, which is capable of identifying overlapping and plagiarised documents. Each component of the system is briefly described. The matching-engine component uses a modified suffix tree representation, which is able to identify the exact overlapping chunks and its performance is also presented.

KEYWORDS: overlap detection, string-matching, suffix tree, distributed system

INTRODUCTION

Digital libraries and semi-structured text collections provide vast amounts of digitised information on-line. Information is duplicated for several different reasons. Documents may appear in different file formats (for example MS Word, PostScript, HTML, etc.), may be stored on mirror sites for easier access, or users can copy documents for illegal purposes, i.e. creating plagiarised assignments, redistributing an electronic document that they do not have copyright for.

There are several scenarios when the elimination of duplicates can be useful. Search-engines often retrieve identical or nearly identical documents. Copy-detection mechanisms may be used as a filter on search results, which excludes documents that overlap with documents above a certain threshold. Another application could be used to show the differences between these documents. If we have a set of related documents that might overlap to a certain extent we are probably interested in the union of these documents where duplicate parts are eliminated.

Finding illegal copies of documents is another application of copy-detection methods. Publishing documents electronically bears a risk of easy illegal redistribution. A user can purchase a document and then redistribute it in Usenet groups or Web pages. With the help of search-engines users can easily find relevant documents of interest. It is also very easy and tempting to create assignments or research papers by cut-and-paste or drag-and-drop procedures. Different styles within the paper often reveal plagiarisers but without presenting the actual source it is hard to prove plagiarism. A copy-detection system can find documents that overlap with the submitted document either in a local repository or on the Internet. It is another question whether a legal action would be successful even if plagiarism is obvious.

There are a few systems built for plagiarism detection including SCAM[1] and plagiarism.org[4]. They build an index on a collection of registered documents by using hashing algorithms and comparing these hashed values. They report plagiarism if overlap is above a certain threshold.

THE MATCHDETECTREVEAL (MDR) SYSTEM

The MDR prototype system is built for copy-detection. It compares a given suspicious document to the set of candidate documents. We briefly introduce the *Matching Engine* component of the system below and then give a brief description of other components.

Matching Engine

The core component of the system is the Matching Engine. It uses string-matching algorithms based on suffix trees to identify the overlap between the suspicious document and candidate documents.

The algorithm builds a suffix tree for the suspicious document and compares candidate documents to the suffix tree. The suffix tree is built using a modified Ukkonen's algorithm [2]. The positions and lengths of chunks that overlap are identified by the matching statistics algorithm [2]. Running time of the algorithm is depicted in *figure 1*. The running time is compared to the total size of candidate documents to be compared to the suspicious document. It shows that the running time is linear and it is around 1 sec for 822,600 bytes of candidate documents, which is approximately 550 pages.

Not only can the matching-engine run on a single computer but we can also schedule detection jobs on a cluster of PCs [3]. Jobs then can be executed in parallel utilising either a special-purpose cluster, which may also hold a distributed repository, or idle workstations. Clusters may also be used in a distributed fashion by identifying idle resources on the Internet. We can assign jobs to resources that are actually in the close proximity of the document to be analysed and downloading time can be reduced in this way. Results can be collected when all nodes have finished processing. Results must be carefully merged because different nodes may find the same matching chunk.

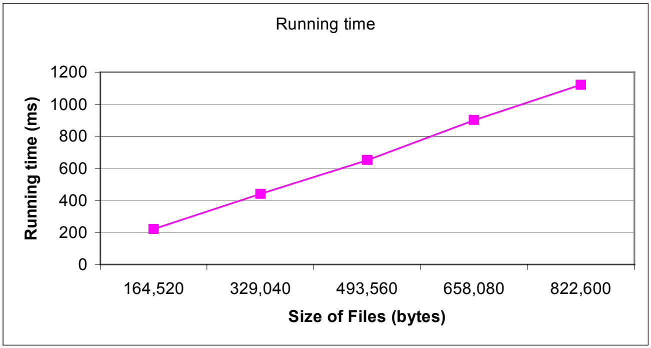

Figure 1. Running time of the algorithm

Figure 2 shows how the number of processors effects the overall execution time. In this experiment the suspicious file was a file with 30% genuine work and the rest was taken from 8 different documents. 350 files (total size 84MB app. 50,000 pages) were compared to the suspicious document and they included those files that were used for creating the suspicious document.

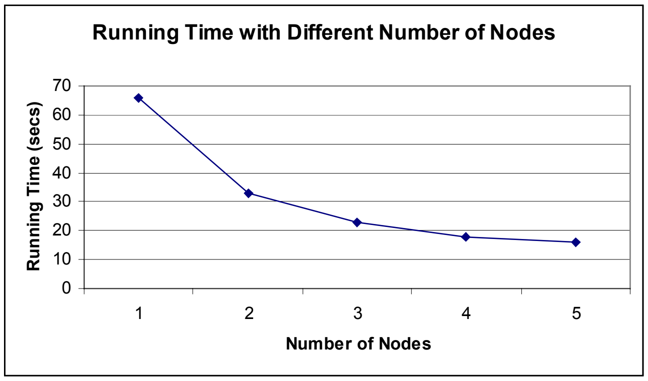

Figure 2. Running time in the parallel case

The experiment shows that the speed-up is not linear. It is because these jobs are very data-intensive and the network gets more congested when we use more nodes. In the future we will also experiment with distributed approaches when documents are distributed among the nodes before the execution starts. Different job-distribution schemes must be applied in this distributed repository case because the cost of comparing a local document is much cheaper than comparing documents on other nodes.

Other components

Matching-engine processes documents converted by the *Converter* component. Not only are documents converted from for example PostScript to plain ASCII but plain ASCII formats are further converted to a unified format, which ignores multiple whitespaces, case-insensitive, and is ready to be processed by the string-matching algorithm.

If overlap is detected the user must be presented with a clear view of detected overlapping chunks. Our priority in developing the system was the matching engine, which currently reports positions and chunk sizes when plagiarism is detected. In the future the *Visualiser* component will be responsible for delivering an easy-to-use user interface.

The *Search-engine* component is currently under development and it is responsible for identifying candidate documents to be compared to the suspicious document. At the moment we are experimenting with different indexing technics.

Agent technology will also be used to find candidate documents in digital collections. Agents are sent to different sites carrying the suspicious document or its index and the suspicious document can be analysed against documents stored in the given digital library.

Document Generator is a supplementary component, which generates "plagiarised documents". The "genuine" part of the document is randomly created and different chunks are randomly extracted and inserted from a base document set. Document Generator can also be configured to use MS Word's thesaurus to substitute a certain number of words in each chunk with their synonyms. Document Generator was used to produce a test document set for performance analysis.

Exact copies of chunks are only the simplest way of plagiarising. To address more sophisticated ways the *Similarity and Rule Interpreter* component defines how to handle inexact matches, i.e. using synonyms.

ACKNOWLEDGMENT

Support from Distributed Systems Technology Centre (DSTC Pty Ltd) for this project is thankfully acknowledged.

REFERENCES

1. Garcia-Molina H., Shivakumar N. SCAM: A Copy Detection Mechanism for Digital Documents. *Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries (DL'95), June 11 - 13, Austin, Texas*, 1995.
2. Gusfield D. *Algorithms on Strings, Trees, and Sequences. Computer Science and Computational Biology*. (Cambridge University Press), 1997.
3. Monostori K., Schmidt H., Zaslavsky A. Parallel Overlap and Similarity Detection in Semi-Structured Document Collections. *Proceedings of 6th Annual Australasian Conference on Parallel And Real-Time Systems (PART '99),* Melbourne, Australia, 1999.
4. Plagiarism.org, the Internet plagiarism detection service URL http://www.plagiarism.org, 1999.