

Identifying Overlapping Documents in Semi-Structured Text Collections

Krisztián Monostori, Arkady Zaslavsky, Heinz Schmidt

School of Computer Science and Software Engineering
Monash University, Melbourne
AUSTRALIA

E-mail: {krisztian.monostori, arkady.zaslavsky, heinz.schmidt}@infotech.monash.edu.au

Digital libraries provide vast amounts of digitised information on-line. Preventing these documents from unauthorised copying and redistribution is a hard and challenging task, which often results in not putting valuable documents on-line [Garcia-Molina et al., 1995a]. Copy-prevention mechanisms include distributing information on a separate disk, using special hardware or active documents [Garcia-Molina et al., 1995b]. One of the most current areas of copy-detection applications is detecting plagiarism. With the enormous growth of the information available on the Internet students have a handy tool for writing research papers. With the numerous search engines students can easily find relevant articles and papers for their research. But this tool is a two-edge sword. These documents are available in electronic form, which makes plagiarism feasible by cut-and-paste or drag-and-drop operations.

Systems for detecting plagiarism and illegal copies are almost non-existent. This poster presents the core component (Matching Engine) of the MatchDetectReveal (MDR) system currently under development. The matching-engine uses string matching algorithms based on suffix trees. The aim of this algorithm is to identify overlapping chunks between the suspicious document, the one we suspect to have overlap with other documents, and the candidate documents, those that have probably been used for creating the suspicious document. Identifying candidate documents is out of the scope of this poster.

We considered the implementation issues of Ukkonen's algorithm [Ukkonen, 1995] for building the suffix tree and Chang's matching statistics algorithm [Chang et al., 1994] for finding overlapping texts. We first considered the space-requirement of a suffix tree and we found that a suffix tree for a 1M pure text, which is approximately 600 pages, takes up 138MB of memory. We tailored the algorithm for our purposes by only inserting suffixes of the document starting at the beginning of words since we are not interested in overlaps that start in the middle of a word. For example we are not interested in an overlap like 'sign of' when we have the following two texts: 'Mr. Harper was responsible for the design of the system' and 'It was a sign of sportsmanship'. By this modification we can reduce the space requirement to approximately 20%. The modified Ukkonen's algorithm has to take into consideration the different nature of the tree because suffix links have different meaning in this representation, which is presented in this poster. Also performance issues are addressed: both time and space requirement of the original and modified algorithms are presented.

Chang's matching statistics algorithm is also modified to utilise the modified suffix tree structure. The poster presents two different approaches to use Chang's algorithm for finding overlap. One approach is the direct application of Chang's matching statistics algorithm when the suspicious document is kept as a string in memory and we build a suffix tree for each candidate document one by one. The drawback of this approach is the time penalty of building a suffix tree for each document. We call this approach the STCD approach (Suffix Tree for Candidate Documents).

We analysed the possibility of building only one suffix tree, that is for the suspicious document, and comparing candidate documents as strings to that suffix tree. The obvious advantage of this

approach is that we only have to build one suffix tree, which can be kept permanently in memory. In this poster we present this 'reverse' application of Chang's algorithm. We discuss the necessary modification, ie. extra information to be stored in the tree, in order to be able to use Chang's algorithm reversely. The main problem of this approach is to find the positions in the suspicious document because these positions are stored as leaf indices in the tree. We have to find all the indices because one chunk of a candidate document might have been copied to different parts of the suspicious document. This approach is called STOD approach (Suffix Tree for the Original Document).

We compare the performance of the two algorithms in large document collections and also analyse them in 3 hypothetical scenarios:

- **Scenario 1:** the suspicious document is genuine. There can be accidental overlaps but they do not add up above a given percentage.
- **Scenario 2:** the suspicious document is heavily plagiarized, that is 60-70% percent of the document is copied from 8 different documents and the order of the chunks is mixed up.
- **Scenario 3:** the most part of the suspicious document is genuine, but there is a huge chunk (1 page), which is copied from another document.

After applying either of the approaches we have to define the overall overlap content of the suspicious document, which is not a simple summation of the defined overlaps because candidate documents might have overlap among themselves. The poster also presents the algorithm for defining the overall overlap content of the suspicious document.

References

- Chang W.I., Lawler E.L. (1994). Sublinear Approximate String Matching and Biological Applications. *Algorithmica* 12. pp. 327-344.
- Garcia-Molina H., Shivakumar N. (1995a). The SCAM Approach To Copy Detection in Digital Libraries. *D-lib Magazine*, November.
- Garcia-Molina H., Shivakumar N. (1995b). 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.*
- Garcia-Molina H., Shivakumar N. (1996). Building a Scalable and Accurate Copy Detection Mechanism. *Proceedings of 1st ACM International Conference on Digital Libraries (DL'96) March, Bethesda Maryland.*
- Plagiarism.org, the Internet plagiarism detection service for authors & education (1999). URL <http://www.plagiarism.org>
- Ukkonen E. (1995). On-Line Construction of Suffix Trees. *Algorithmica* 14. pp 249-260.