Using copy-detection and text comparison algorithms for cross-referencing multiple editions of literary works

Arkady Zaslavsky[1], Alejandro Bia[2], and Krisztian Monostori[3]

[1] Monash University, Melbourne, Australia
a.zaslavsky@monash.edu.au
http://www.csse.monash.edu.au/~azaslavs/
[2] Miguel de Cervantes DL, University of Alicante, Alicante, Spain
abia@dlsi.ua.es
http://www.dlsi.ua.es/~abia/
[3] Monash University, Melbourne, Australia
Krisztian.Monostori@csse.monash.edu.au
http://www.ct.monash.edu.au/~kmonosto

Abstract. This article describes a joint research work between Monash University and the University of Alicante, where software originally meant for plagiarism and copy detection in academic works is successfully applied to perform comparative analysis of different editions of literary works. The experiments were performed with Spanish texts from the Miguel de Cervantes digital library. The results have proved useful for literary and linguistic research, automating part of the tedious task of comparative text analysis. Besides, other interesting uses were detected.

1 Copy-detection, plagiarism and comparative literary analysis

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 and Shivakumar, 1995b]. Copy-prevention mechanisms include distributing information on a separate disk, using special hardware or active documents [Garcia-Molina and Shivakumar, 1995a]. We believe that these approaches are very cumbersome for genuine users, therefore copy-detection approaches are more practical. Copy-detection does not try to hinder the distribution of documents but rather tries to identify illegal copies.

One of the most pressing areas of copy-detection applications is detecting plagiarism. With the enormous growth of the information available on the Internet users have a handy tool for writing research papers. With the numerous search engines users can easily find relevant articles and papers for their research. These documents are available in electronic form, which makes plagiarism achievable by cut-and-paste or drag-and-drop operations. Academic organizations as well

as research institutions are looking for a powerful tool for detecting plagiarism. Copy-detection tools aim to find whole and partial copies of documents either on the Internet or in local repositories. These tools can be used by digital libraries to find copies of their copyrighted documents redistributed on the Web or in newsgroups. The tools can also detect possible plagiarised documents, since textual overlap may prove to be plagiarism. Though there are more "sophisticated" ways to plagiarise, we believe that finding textual overlap would identify most cases of plagiarism.

Apart from the original purpose of plagiarism detection, it occurred to us that a very different use can be given to this kind of powerful comparison programs. Not only can we use such a system to detect illegal, or at least immoral, cases of text duplication but it may also help compare different documents for research purposes. In this paper we will present the result of a comparison among different editions of literary works like the famous Spanish masterpiece *El Ingenioso Hidalgo Don Quijote de la Mancha* of which there exists different versions. Librarians may also use this system to cross-reference different publications in some cases. Different problems of text comparison for DL professionals and linguists are discussed in the following section.

2 Applications of text comparison to DLs and humanities research

The idea of using the MDR (Match Detect Reveal) system with the purpose of establishing cross-references between literary works was born on a casual talk between members of Monash University and the Miguel de Cervantes DL project [Bia and Pedreño, 2001] at the DL'2000 conference in San Antonio, Texas, last year.

The Miguel de Cervantes DL, apart from being the biggest repository of digital literary texts in Spanish language, devotes part of its efforts to carry out multidisciplinary research projects where humanists and computer scientists can join their skills and experience on developing new tools and methods for DLs [Bia and Pedreño, 2001]. From this point of view, this initiative of applying advanced text comparison to literary texts promised to be not only an excellent case study and testing environment for MDR, but also the perspective of discovering a promising tool for digital libraries and humanities research.

We soon worked out the requirement specifications to fine tune MDR to this new kind of application. For preliminary tests, we first collected a set of texts making sure beforehand that some of them included overlapings (parts of text existent on other files of the sample). We also took note of the placement and kind of overlapings. With the help of some linguists from the Miguel de Cervantes DL we have chosen cases of cross-referred texts suitable for the tests. The results of the comparisons were amazing, no match was missed.

We have foreseen three potential applications for MDR on digital libraries and linguistic research, plus another we didn't think of. We will now describe them briefly.

2.1 Detecting cross-references

The first application we thought of, was to automatically detect quotations or cross-references between different texts. In this case the quote could be hyper-linked to the original. We can imagine many research situations where automatic detection of similar sections of text can be useful. It could be a valuable aid in history research for instance.

2.2 Organizing collections of small pieces of text

Another use is to detect repeated poems in different poem collections (may also be tales, letters, etc.), where some of these textual units are repeated in different editions. Most often, when collections of poems are developed, it is difficult to keep track of which poems have been included and where, and which not. MDR proved to be a helpful tool for detecting and locating the corresponding pieces of text for verification purposes. One of our first experiments was to compare a couple of collections of poems from *Ramón de Campoamor y Campoosorio*. We could easily keep track of the poems that appeared in both collections. Figure 1 shows the cross-reference of one isolated poem that appears in two different poem selections.

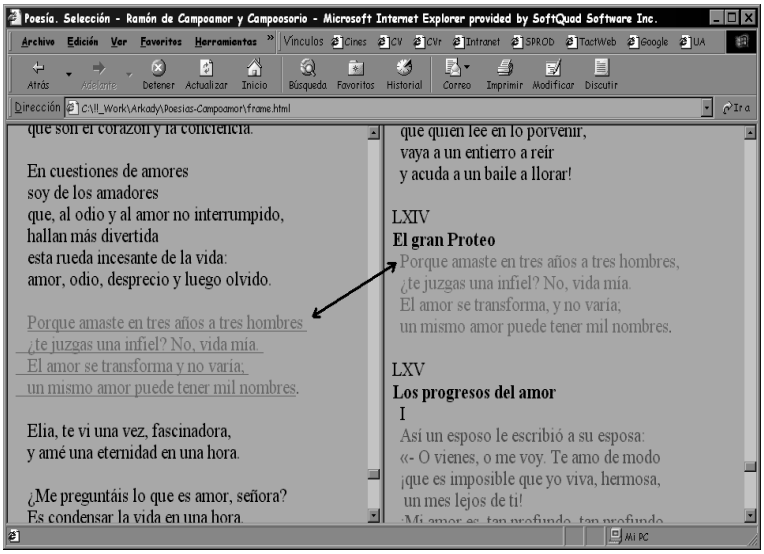

Fig. 1. A repeated poem in two different poem collections.

2.3 Comparative analysis of texts

The objective is to compare different editions of the same literary work. Philologists are usually interested in this kind of comparisons for research purposes.

In ancient literature, there usually are different editions of the same work, with modifications performed some times by the author, some others by the editor. This comparative analysis is in itself an interesting but tedious field of study where any kind of automation is welcome. To cite an example, there are various ancient editions of the *Quijote de la Mancha*, all originals from Cervantes, but still different. This application differs from the previous ones, in the sense that here we have to find differences instead of matches. Although in this case the matching strings are usually in the same order in both sources, synchronization is not always possible since the sources may have long additions inserted in different places, apart from small differences in the matching zones. A conventional sequential comparison may also fail in this case.

In the example of figure 2, we can see clear differences in two versions of Don Quijote: on the left frame we can read *"Eran cuatro, y venían con sus quitasoles, con otros cuatro criados a caballo y dos mozos de mulas a pie"*, while on the right one we see *"Eran seis, y venían con sus quitasoles, con otros cuatro criados a caballo y tres mozos de mulas a pie"*. The merchants from Toledo, in one case are six, in the other four, and their servants on foot are two on one side and three on the other. A little bit above there is another difference in meaning. On the left it says "muy pensado" (thoughtfully) and on the right "muy bien pensado" (very well thought). These are examples of changes introduced by editors through the centuries.

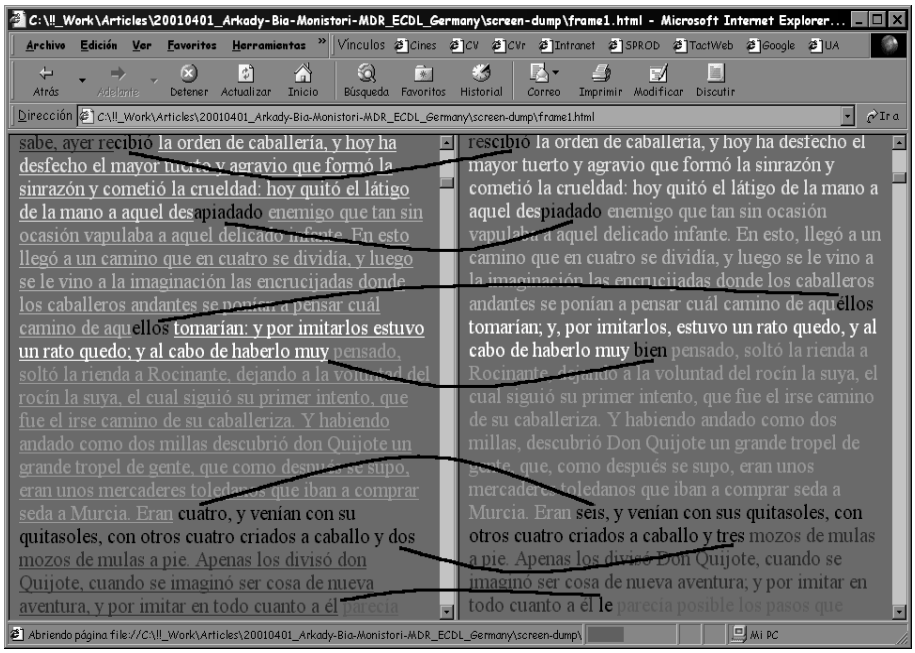

Fig. 2. Comparing two editions of *Don Quijote* (the real screen makes full use of colour to highlight the matches)

2.4 Detecting variations and mistypings

There are variations in spelling owed to ancient editor changes, which are not strictly errors since the Spanish language was not normalized until 1713, where the *Real Academia de la Lengua Española* was created and language spelling started to settle. Those variations, though legal, produce differences between editions. An example of ancient spelling variations can be seen in figure 2, where we can see *"recibió"* and *"rescibió"*, *"desapiadado"* and *"despiadado"*, and *"aquellos"* and *"aquéllos"*. None of these are errors, but ancient variations.

On the other hand, no matter how much care correctors put on freeing texts from digitization errors, there are always some errors that remain. So an unforeseen result of the application of MDR to comparative analysis was the detection of spelling errors or variations in DL texts, out of the comparison of different editions of the same work.

3 Existing Tools and Algorithms

Copy-detection schemes have two fundamental approaches: digital watermarking, and string comparison. The target of digital watermarking is to find illegal copies and track down the user who purchased the electronic document and redistributed it. This method cannot be used by librarians for the tasks discussed in the previous section. In watermarking methods, undetectable codewords are placed in documents similar to the method of placing watermarks in bank notes. These codewords can represent a unique document identifier and this identifier can be assigned to a given customer who purchased the document. Codewords can be hidden in the document by slightly altering the layout of the text. These alterations must be reliably decodable yet not perceptible to the user. In string-comparison based algorithms we can define the basic problem as with given two strings T and P with the length of m and n respectively, we have to divide T and P into substrings $T = t_1 s_1 t_2 s_2 t_3 s_3 ... t_k s_k t_{k+1}$ and $P = p_1 q_1 p_2 q_2 p_3 q_3 ... p_r q_r p_{r+1}$ where for each s_y there is an x so that $s_y = q_x$ and $S(|s_y|)$ is maximal. As an example compare $T = abcdbcadca$ and $P = aabaca$. One possible partition is $T = ()(ab)(cdb)(c)(adc)(a)()$ and $P = ()(a)()(ab)(a)(c)(a)$. T is partitioned into $t_1 = ()$, $s_1 = (ab)$, $t_2 = (cdb)$, $s_2 = (c)$, $t_3 = (adc)$, $s_3 = (a)$, $t_4 = ()$ and P is partitioned into $p_1 = ()$, $q_1 = (a)$, $p_2 = ()$, $q_2 = (ab)$, $p_3 = (a)$, $q_3 = (c)$, $p_4 = (a)$. $s_1 = q_2$, $s_2 = q_3$, $s_3 = q_1$. $S(|sx|) = 3$, which is not maximal in this case because the partition giving the maximum for $S(|sx|)$ is $T = ()(ab)()(c)()(b)()(ca)(d)(ca)()$ and $P = ()(a)()(ab)(a)(ca)()$. The maximum overlap is a single value, but different partitions can result in the same maximal overlap value, that is, the partition is not unique. We analysed four research prototypes that aim to solve the above-defined maximal overlap problem. These research prototypes include the SCAM (Stanford Copy Analysis Method) system [Garcia-Molina and Shivakumar, 1995a], the Koala system [Heintze, 1996], the "shingling approach" of [Broder et al.,], and the file system clustering method of the sif tool [Manber, 1994]. The general approach to define the maximal overlap is to split up the text into chunks, calculate some hash function on those

chunks, select certain chunks from all the chunks to be stored in an index. When a document has to be compared to documents in the repository, the suspicious document, that we want to compare to others, has to be chunked using the same strategy as used in registration. The overlapping chunks then must be identified. Given these chunks, a decision function decides whether a given document is plagiarised, is a partial or exact copy of other documents depending on the objective of the system. If we want to use these methods for identifying the differences between e.g. two different versions of *Don Quijote* then this method is constrained by the chunking method and the selection strategy used. If we are using non-overlapping chunks of text of e.g. ten words then any difference between the texts will be at least ten words. As we will see in Section 5 sometimes there are only slight differences, such as different accents used, which are not easily picked up by the output of these systems. Our approach, which is discussed in the following section, uses a suffix tree structure, which is able to identify exact overlapping chunks by using the matching statitistics algorithm. The main advantage of this algorithm that it is able to pick up smaller differences, such as the above-mentioned differences in accents.

4 MDR Architecture

The procedure of processing documents is the following. Documents must undergo preprocessing in order to be indexed in the repository. This preprocessing converts documents in different formats into pure ASCII text and then they are further converted into a format that can be used by the matching-engine. This format also makes sure that simple modifications, i.e. using different punctuations, adding extra spaces, will not hinder detection.

Documents are then chunked and hashed and then recorded in the index. When a suspicious document is submitted to the system the search-engine identifies candidate documents in the index and candidate documents are compared using the matching-engine. The matching-engine builds a suffix tree for the suspicious document and compares the candidate documents to the suffix tree. A suffix tree is an index that represents all suffixes of a string, which means that all substrings are also represented. Suffix trees can be built in linear time, and our implementation uses Ukkonen's [Ukkonen, 1995] algorithm for the construction. Chang's matching statistics algorithm [Chang and Lawler, 1994] can be used to create an array of numbers that represents the longest overlapping chunk starting in each position in the suspicious document. Once this array is given it is easy to calculate overlap-percentage, the only thing we have to consider is that a matching statistic value of k at position i represents the same chunk as a matching statistic value of k-1 at position i+1.

Matching statistics can be calculated in linear time. The running time of the algorithm is proportional to the size of the string to be compared to the tree and independent from the size of the tree. Unfortunately suffix trees are too big to be used as an index, though the matching statistics algorithm runs fast (app. 1MB/sec) making it possible to be used in the second stage. The

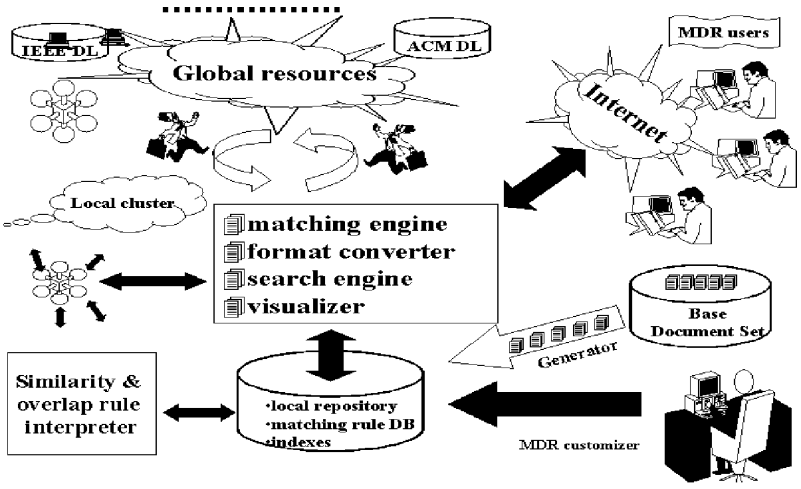

Fig. 3. This figure depicts the architecture of MDR. It is a general architecture that shows all potentials of the MDR system with different components.

accuracy of the system is the highest possible because it identifies exact chunks and does not include any uncertainty, such as hashing. We consider this algorithm a method that could complement those methods used in other prototype systems and incorporated into our search-engine.

The MDR system also contains a submission system, whose scope is wider than the submission component of other prototype systems. It contains a complete system for submitting assignments for different courses. The similarity and rule interpreter component is under development. We would like to set up a set of rules for handling obvious modifications: changing the name of geographical names, changing numbers, adding or deleting extra whitespaces, etc. The document generator is a complementary component that can be used to generate documents that overlap with documents in the base document set. These documents then can be used to test our algorithms. Using the local cluster for the comparison is discussed in [Monostori et al., 1999]. Comparison jobs can be generated when a suspicious document is submitted and the local cluster can be used to compare documents in parallel.

This architecture of the MDR system is tailored for plagiarism-detection and copy-detection applications but core components may be used for other purposes including cross-referencing and comparison of different versions of the same document as mentioned in Section 2. If we have text versions of those documents we simply use the converter component to eliminate non-significant differences, i.e. multiple whitespaces, different punctuations. As soon as we have a canonical form of the document we can submit it to the matching engine, which provides us with positions and lengths of overlapping chunks. The Visualiser

component is responsible for generating HTML files from these positions. Figure 2 depicts the output of the comparison of two versions of *Don Quijote*.

Identical colours depict identical chunks in both documents and black chunks are the differences between the documents. We could use a different colouring scheme that would use black for identical chunks and different colours for differences. The colouring scheme used stems from the colouring scheme of visualising plagiarised documents where similarity is more important than difference. The following subsection describes briefly how such comparison is done by the matching engine.

4.1 The matching engine

The matching engine component uses a suffix tree structure to compare documents. A suffix tree is a data structure that includes every suffix of a given string, thus every substring. Suffix trees can be built in linear time using different algorithms [Ukkonen, 1995,Gusfield, 1997]. We used Ukkonen's algorithm not only because of its relative simplicity but because it also stores suffix link information, which is heavily utilised by the matching statistics algorithm. The suffix tree of the string *abcdabca*\$ is shown in figure 4. Chang et al. [Chang and Lawler, 1994]

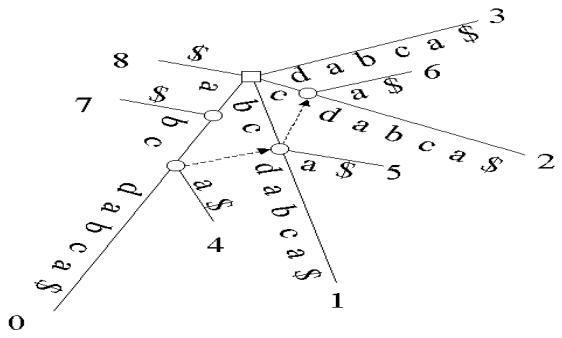

Fig. 4. Suffix tree of the string *abcdabca*\$

describes an algorithm for finding so called matching statistics - $ms(i)$, which is the longest substring of T starting at position i that matches a substring somewhere in P. Having built a suffix tree for P, matching statistics of T can be found in $O(n)$ time where n is the length of T. The main idea of the algorithm is that starting from the first position in T we calculate the matching statistics $ms(0)$ by traversing the suffix tree of P. Then, in order to calculate $ms(i)$ while having calculated $ms(i-1)$ we have to back up to the node above our current position, follow the suffix link of that node and then traverse down from that node.

We have tailored both Ukkonen's building algorithm and Chang's matching statistics algorithm by taking into account that we only want to find overlaps

starting at beginning of words. With this modification the space requirement can be reduced to approximately 20% of the original suffix tree and the matching statistics algorithm also benefits from this modification. This modification is similar to the one in Baeza-Yates et al. [Navarro et al., 1999] but they only suggest that substrings starting at the beginning of words are to be included and they do not consider how to build the tree in linear time. We analysed and modified Ukkonen's and Chang's algorithm to reflect these changes. We will also analyse the possibility of inserting only those suffixes that start at certain positions, e.g. every 30th, for further reducing the size of the suffix tree. Part of a modified suffix tree for the converted string "-they-were-the-last-to-arrive-but-they-were-not-late-" is depicted in figure 5. The character [-] depicts a space. If during the matching statistics algorithm we traverse down the route labelled

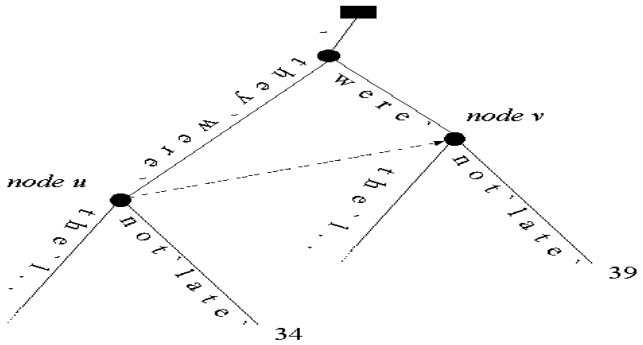

Fig. 5. Part of a Modified Suffix Tree

by "-they-were-not-" in the next step we do not have to start matching at the root rather we follow the suffix link between node u and node v (depicted by the dashed arrow) and continue traversing the tree from node v. This interpretation of the suffix link is different from the one in the original Ukkonen's algorithm and since we only insert suffixes starting at the beginning of words a suffix link points to the node with the label of the original node without the first word. In our example the label of node u is "-they-were-" and the label of node v is "-were-". Not only does this modified suffix tree require less space but the matching statistics algorithm can also benefit from it calculating the matching values only for positions where words start.

5 Results of comparing different editions of Cervantes's Quijote with MDR

We compared three different editions of *El ingenioso hidalgo Don Quijote de la Mancha* by Miguel de Cervantes Saavedra. The first one, catalogued as #001270,

is a digital edition based on the Madrid edition, published by Espasa-Calpe, 1911-1913. The second, catalogue #002035, is a digital edition based on the Barcelona edition, typography from F. Giró, 1888, 4th. ed., donated from the private library of Emma María Guijarro Hortelano, with a great number of engravings and reproduction of contemporary paintings. The third one, catalogue #002693, is a digital edition based on the Barcelona edition, A. López Robert, 1905, 3rd ed., with illustrations by the famous draftsman Don José Passos.

In spite of being both from the Barcelona branch of the Quijote family tree, editions #002035 and #002693 show phrases with opposite meanings:

- #002035: algunos que, no conteniéndose en los límites de su ignorancia
- #002693: algunos que, conteniéndose en los límites de su ignorancia

More surprising is to see that the number of silk merchants from Toledo that Don Quijote see in their way to Murcia were six, or just four?, and were accompanied by three or just two "mozos de mulas"?

- #001270: Eran seis, y venían con sus quitasoles, con otros cuatro criados a caballo y tres mozos de mulas a pie. (see figure 2)
- #002035: Eran cuatro, y venían con su quitasoles, con otros cuatro criados a caballo y dos mozos de mulas a pie.
- #002693: Eran seis, y venían con sus quitasoles, con otros cuatro criados a caballo y tres mozos de mulas a pie.

Below we read "Florimorte de Hircania", or is it "Florismarte de Hircania"?

- #001270: -Éste que se sigue es Florimorte de Hircania -dijo el Barbero. -Ahí está el señor Florimorte? -replicó el Cura-.
- #002035: Éste que se sigue es Florismarte de Hircania, dijo el barbero. Ahí está el señor Florismarte? replicó el cura;
- #002693: Éste que se sigue es Florismarte de Hircania dijo el barbero. Ahí está el señor Florismarte? replicó el cura.

These are examples of mismatches, easily detected by MDR. In these cases the differences are in meaning. In others we see that one version has been modernized: ancient forms of Spanish have been converted to modern, normalized Spanish.

The way traditional literature researchers like *Diaz y Diaz* did this job of comparative analysis was by filling tables by hand with the mismatches they found when reading two texts side by side. This method was slow and error prone. One important advantage of working with this kind of computerized tool is time saving. It takes only a few seconds to compare two versions of *El Quijote*, compared to weeks or months it may take without computer aid. Digital librarians and literary researchers also benefit from the friendly interface, that takes advantage of hyperlinks to match the coincidences and in this way revealing the differences with better accuracy and less mistakes than using traditional methods.

6 Conclusions

An interesting application of this technique is the development of critical editions. These are based on the comparative analysis of different editions of the same literary work, usually taking one that is considered the most faithful to the original work, and then annotating the differences between this text and the each of the others. During this process, misprints or editor changes will be detected. The research can focus either on ancient lexical usages or on the style of the author. A tool like MDR would save a considerable amount of time and effort.

Many differences between editions are related to punctuation and capitalization, since editors used to make this kinds of modifications at will. Ignoring them, as MDR does is desirable in most cases.

Conventional file comparison programs usually do sequential string comparison, which is a good approach when comparing similar texts in search of differences. Since sequential matching requires synchronization of the sources, this approach is useless when the matching strings appear in a different order and mixed with other text zones that do not match, i.e. when the texts are too different. MDR proved to be specially suitable for this task, since it does not require the matching zones of text to appear in any order.

MDR outperforms conventional sequential file comparison tools, being able to detect even small matches in texts which are highly different. These matches can appear in any place and any order and still be successfully detected. The format of the output, as cross-linked HTML files displayed on conventional browsers using frames is very convenient and practical.

Concerning digitisation processes, MDR can be used to compare the output of different OCR (optical character recognition) programs applied to the same book, or to compare two versions of the same text that have been corrected in parallel by different persons, or just to compare two versions of the same file to verify the changes performed to one to produce the other.

7 Future work

MDR detects identical chunks of texts between two files, ignoring only punctuation and capitalization. Another interesting approach would be to detect similar but not identical strings. One possible approach could be to ignore a set of given stopwords, i.e. words that are generally meaningless, reducing the matching to compare meaningful words like nouns and verbs. The use of synonyms could be another attempt. This would be useful not only to detect plagiarism in the modern sense, but also to detect *imitatio*, the ancient practice of novel writers imitating the style and sometimes copying entire phrases of acclaimed writers. This is another interesting point in literary research.

Another possible improvement concerning the interface could be to use XML instead of HTML to mark up the output of MDR, indicating the matches and mismatches detected and the cross-linking between the chunks of texts of

the compared files. Then we can use XSL transformations to produce different HTML renderings customized for different types of research: in this way we could highlight the differences, or otherwise the similarities, or produce bidirectional cross-links, or just customize the colors for ease of reading.

References

[Bia and Pedreño, 2001] Bia, A. and Pedreño, A. (2001). The Miguel de Cervantes Digital Library: The Hispanic Voice on the WEB. *LLC (Literary and Linguistic Computing) journal, Oxford University Press*, 16(2):161–177. Presented at ALLC/ACH 2000, The Joint International Conference of the Association for Literary and Linguistic Computing and the Association for Computers and the Humanities, 21/25 July 2000, University of Glasgow.

[Broder et al.,] Broder, A., Glassman, S., and Manasse, M. Syntatic Clustering of the Web. In *Sixth International Web Conference*, Santa Clara, California, USA. URL: http://decweb.ethz.ch/WWW6/Technical/Paper205/paper205.html.

[Chang and Lawler, 1994] Chang, W. and Lawler, E. (1994). Sublinear Approximate String Matching and Biological Applications. *Algorithmica*, 12:327–344.

[Garcia-Molina and Shivakumar, 1995a] Garcia-Molina, H. and Shivakumar, N. (1995a). SCAM: A Copy Detection Mechanism for Digital Documents. In *Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries (DL'95)*, Austin, Texas.

[Garcia-Molina and Shivakumar, 1995b] Garcia-Molina, H. and Shivakumar, N. (1995b). The SCAM Approach To Copy Detection in Digital Libraries. *D-lib Magazine*.

[Gusfield, 1997] Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press.

[Heintze, 1996] Heintze, N. (1996). Scalable Document Fingerprinting. In *Proceedings of the Second USENIX Workshop on Electronic Commerce*, Oakland, California. URL: http://www.cs.cmu.edu/afs/cs/user/nch/www/koala/main.html.

[Manber, 1994] Manber, U. (1994). Finding similar Files in a Large File System. In *Proceedings of the 1994 USENIX Conference*, pages 1–10. URL: http://www.cs.cmu.edu/afs/cs/user/nch/www/koala/main.html.

[Monostori et al., 1999] Monostori, K., Zaslavsky, A., and Schmidt, H. (1999). Parallel Overlap and Similarity Detection in Semi-Structured Document Collections. In *6th Annual Australasian Conference on Parallel And Real-Time Systems (PART '99)*.

[Navarro et al., 1999] Navarro, G., Baeza-Yates, R., and Ribeiro-Neto, B. (1999). Indexing and searching. In *Modern Information Retrieval*, chapter 8, pages 191–228. ACM press and Addison Wesley, Edinburgh Gate, Harlow, Essex CM20 2JE, England, 1st edition. See also http://www.dcc.ufmg.br/irbook or http://sunsite.dcc.uchile.cl/irbook.

[Ukkonen, 1995] Ukkonen, E. (1995). On-Line Construction of Suffix Trees. *Algorithmica*, 14:249–260.