Term Paper CompSci 725

Diana Butters dbut026

3364715

Comparison of matching algorithm used in three plagiarism detection systems.

Abstract

SCAM (Stanford copy analysis Mechanism, MDR (Match Detect Reveal) and SE (Signature Extraction) are three copy detection mechanisms that take a document and compare it to a corpus of documents for similarities. Scam and SE use similar comparison methods, but with different data, while MDR uses similar data to Scam with a much more accurate comparison method.

1. Introduction

Plagiarism detection is a growing concern in many educational facilities because access to material has greatly improved since the advent of the web. Cut and Paste Plagiarism is becoming more and more common in student papers. There are many commercial plagiarism detection systems available for this purpose such as turnitin.com [Turnitin] and the Glatt system [Glatt]. Another area of concern that is harder to check is plagiarism in conference papers. These systems have been developed with this area in mind. SCAM has also been successfully used in a case of plagiarism detection details available at http://www.dlib.org/dlib/november95/scam/plag.html

In this paper I am comparing three plagiarism detection systems. First I am comparing the process of compiling a corpus of documents to be compared against, In the next section I shall describe then compare the methods of breaking the documents into smaller bits called chunking and then in the following section I discuss the comparison techniques of the systems, followed by a summary of the results of the three systems.

2. Corpus Compilation

In the following sections I discuss the methods involved in creating the corpus of documents that each system scans against.

2.1 Scam Corpus

Scam just involves a database compiled of submitted documents these are stored as an inverted structure as shown in Figure 1.

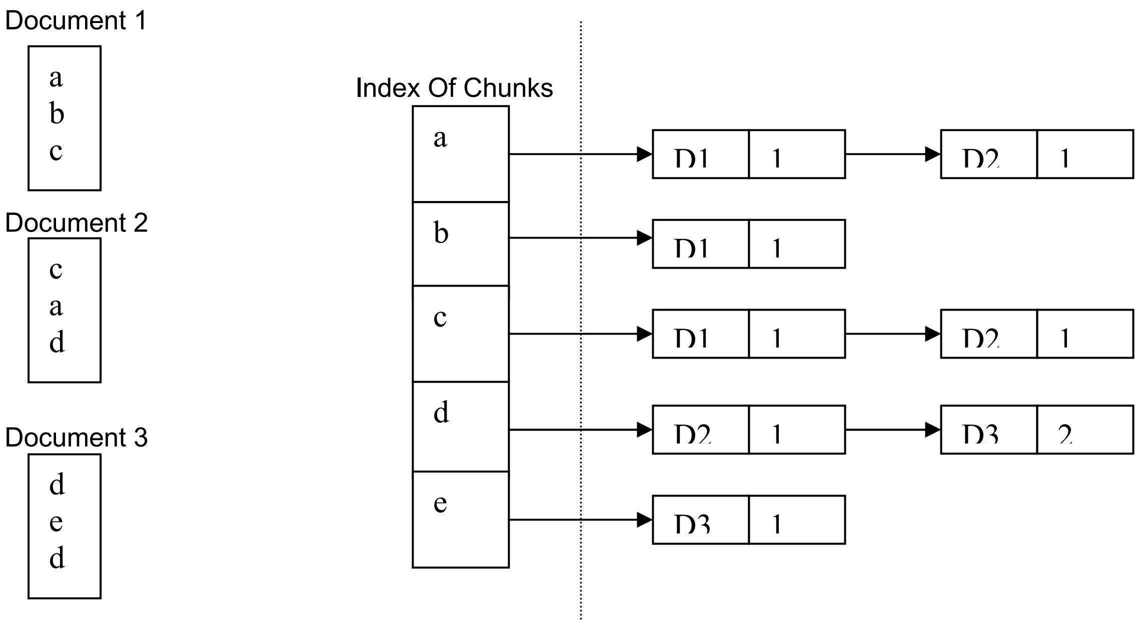

Figure 1. Inverted Index Storage Structure [Shivakumar, 1995]

2.2 MDR Corpus

MDR uses a document generation system, which generates documents from a pool depending on specified variables. Each document is stored in a database and then is converted to a suffix tree for comparison if the corpus generator selects it. This reduces the number of suffix trees required when checking the comparisons as you can remove unrelated documents, which are highly unlikely to have been used to plagiarise from, for example if you where checking for plagiarism on the topic of automata, it is highly unlikely that you need to check articles on cats.

2.3 SE Corpus

SE provides three different ways of doing comparisons the first and second involves submitting directories in which all the files are compared against each other, while the third method involves the submission of a directory in which all files are processed then stored in a database. This database is the equivalent of the corpus of documents that the other systems use. These are stored using an inverted index structure like Scam's storage system.

3. Chunking methods

Each of the three methods use similar but different techniques for breaking the documents into small pieces. I shall discuss how they break the documents into smaller pieces followed by the advantages and disadvantages of each system.

3.1 Scam – Chunking Method

Scam parses each document and removes white space and punctuation. Each word is then stored in an inverted storage structure as shown in Figure 1. The advantage of this method is it will allow the detection of partial sentences, as opposed to using larger chunks, which may miss these. However using smaller chunks increases the occurrence of false positives, which are unrelated papers being returned as plagiarised.

3.2 MDR – MatchDetectReveal

Parses document into a format suitable for creating a suffix tree. This format changes all alphabetic characters are converted to lowercase, and it changes all non-alphanumeric characters to a single non-alphanumeric character, and leaves all alphanumeric characters as they are. Some characters are shifted to create a contiguous 38-character alphabet including a termination character.

An example of this conversion is

Mr. X. plagiarized

a lot of documents

according to (Garcia Molina et al.,1996b)

is converted to

‘mr’x’plagiarised’a’lot’of’documents’according’o’garcia’molina’et’al’W__\b’
Example taken from [Monostori, 2000]. This is then converted into a modified suffix tree. This conversion uses Ukkonen’s building algorithm cited by[Monostori, 2000], but they have taken into consideration that only overlaps at the beginning of words are required, rather than starting halfway through a word. This reduces the size of the suffix tree that is required. An example of a suffix tree is shown in Figure
2.

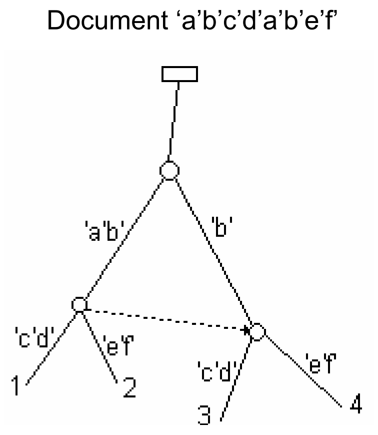

Figure 2. Example of suffix tree.

3.3 SE – Chunking Method

SE uses a hashed breaking point function described in [Shivakumar, 1996], the first word is hashed h a value is computed using $h \bmod c$ where c is a constant. This value is then compared to another constant often 0. If it equals this constant then that is the chunk used for the next step otherwise the process is repeated and both words comprise the chunk, this continues until the comparison = 0. After the document has been chunked a culling process takes place. This involves the removing of the shortest chunks, which two unrelated documents may contain and the longest chunks because a plagiariser is not likely to have copied long passages of text. Two methods of performing this culling where tested the better of the two involves calculating the variance using $|L-m| \leq b$ starting with $b=0.1$ increasing b until $\sqrt{n}$ chunks are selected [Finkel, 2000]. After the chunks have been culled they are digested. The chunks are transformed into 128-bit numbers using the MD5 algorithm [Rivest, 1992]. Only the leading numbers are kept up to a constant i.e. only the first 10 numbers identifying each chunk are kept. These digested chunks are notated by $d(F)$ for a File F. These are then stored in a pair of hash tables, nameData and keyData. nameData contains the personal information of the file submitter, the number of chunks in a file and the date submitted. The name of the submitter should be unique and difficult to guess if required. keyData contains where each digest is located. Figure 3 shows the structure of these two files.

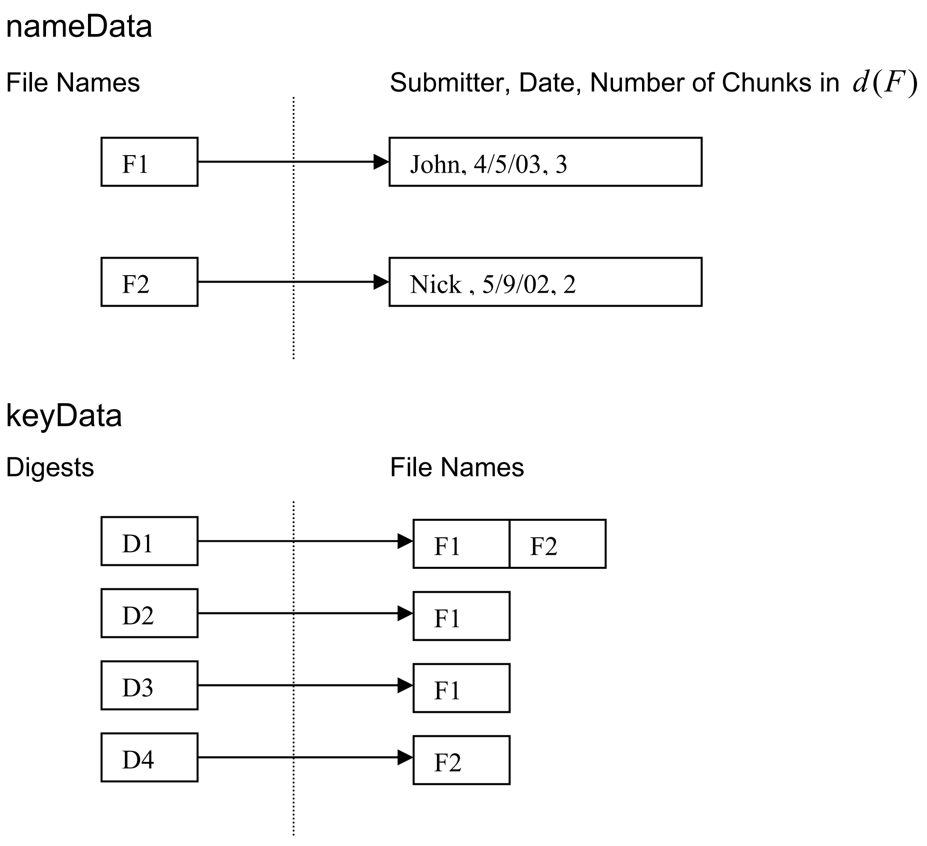

Figure 3. A diagram of the storage structure for Signature Extraction

3.4 Similarities and differences

Both Scam and MDR use words as chunks, but are stored very differently, while SE uses a hashing function to create the chunks, these chunks may contain different numbers of words. These chunks are then culled and hashed to create a digest. Although the storage of the digest is very similar to that used in Scam, the actual data is very different.

4 Matching Algorithms

Due to the different storage systems the matching algorithms are quite different. MDR involves walking a suffix tree while scam uses an index system. SE also uses an index system but comprises of chunk signatures or digests rather than words.

4.1 Scam Matching Algorithm

Scam uses a relative frequency model first a *closeness set* $c(D_1, D_2)$ is defined. Where D is a document. $F_i(D)$ is the occurrence of chunk w_i in D. The closeness set contains the chunks that appear a similar number of times in two documents. For example two documents may both contain ‘a’ and ‘b’ so $c(D_1, D_2) = \{a, b\}$. To be considered part of the closeness set the chunk must satisfy the condition

$$\varepsilon - \left(\frac{F_i(D_1)}{F_i(D_2)} + \frac{F_i(D_2)}{F_i(D_1)} \right) > 0 .$$

Where ε is a tolerance factor defined as $\varepsilon = (2^+, \infty)$. $F_i(D)$ is the number of occurrences of a chuck i in document D.

Next to be defined is the *subset measurement*.

$$subset(D_1, D_2) = \frac{\sum_{w_i \in \alpha(D_1, D_2)} \alpha_i^2 * F_i(D_1) * F_i(D_2)}{\sum_{i=1}^{N} \alpha_i^2 F_i^2(D_i)}$$

Where α_i is a weighting associated with the i^{th} chunk, and N is the size of $F(D)$.

This is for calculating if document A is part of document B. It is similar to the *cosine similarity* defined in [Shivakumar, 1995], but it is asymmetric while the cosine similarity is symmetric. The subset measurement is calculated with respect to document one, while the cosine similarity is calculated with respect to both documents.

Then the similarity between the two documents is defined by

$$sim(R,S) = \max\{subset(R,S), subset(S,R)\}$$

The maximum similarity value considered in 1 so any value of $sim(R,S) > 1$ is set to 1. This allows a similarity range of 0% to 100% to be defined.

4.2 MDR - Matching algorithm

MDR uses a matching statistics algorithm (msi) described in [Chang 1994 cited by Monostori, 2000]. The msi is described as calculating the longest substring starting at position I that matches a substring somewhere in p. This is applied to each node in the suffix tree. The tree is then walked to calculate the longest common subtree longest substring of T starting at *i* somewhere in P where T is the suspicious document and *i* is a starting position in T and P is the document in the generated corpus. The modification from Ukkonen's algorithm is that using the example 'a'b'c'd'a'b'e'f', which the suffix tree for is shown in Figure 2. An example of how the suffix link works is that the tree is traversed down branch 2 following 'a'b'c' this route proves incorrect so the suffix link donated by the dashed line, is then followed this allows path 'b'c'… to be followed.

The tree is walked using the following algorithm taken from [Monostori, 2000].

```
last_position, last_value <- -1,
for i=0 to ‘length of suspicious document’ -1 do
    if msi[i]==0
        continue
    end if
    current_value,current_position <- msi[i],i
    if (current_position-last_position)>last_value
        overlap := overlap+last_value
    else if current_value>last_value-(current_position-last_position
        overlap:=overlap+(current_Position-last_position)
        last_Value,last_position <- current_Value,current_position
    end if
end for
overlap :=overlap + last_value
```

4.3 SE matching Algorithm

SE uses a simple matching algorithm it scans document F . The digests in F 's keyData are then compared with $d(G)$, the digests of the other files in the database. This gives $d(F) \cap d(G)$. For each digest in the document that is part of this intersection three measures of similarity are further computed

Asymmetric Similarity

$$a(F,G) = \frac{|d(F) \cap d(G)|}{|d(F)|}$$

Symmetric Similarity

$$s(F,G) = \frac{|d(F) \cap d(G)|}{|d(F) + d(G)|}$$

Global Similarity

$$g(F) = \frac{|d(F) \cap (\cup_G d(G))|}{d(F)}$$

Asymmetric and Symmetric similarity are used in the same way as in Scam, and the Global Similarity specifies the degree of overlap between all other documents in the database. Calculating these figures gives an overall view of the degree of overlap between the documents.

4.4 Summary

In summary Scam and MDR both chunks the document into words using white space as the divider, while SE chunks the document using a hashed breakpoint function. Scam and MDR then use the words as they are for pattern matching while SE performs further processing on the chunks to create a signature. Scam and SE both store chunks in an index structure, while MDR uses a suffix tree to store its data.

5 Performance

MDR uses exact string matching which is very accurate and doesn't create false positives, but is computationally slow, while Scam and SE are much faster, but not as accurate. Scam uses small chunks and therefore increases the amount of false positives that are returned, while SE uses a sampling method which means that very related documents the severity of plagiarism is under reported and the chance of false positives is increased.

6 Conclusion

Results documented in [Finkel 2002] show that MDR is more accurate than SE, but due to the time MDR takes to process the data, SE is the better system to use when there is a large corpus of documents.
If there are concerns about the size of the storage of the corpus Scam is the better system as there is only a fixed number of words in the English language, while SE contain a combination of words.

SE and Scam use very similar comparison methods. The results are quite similar except that SE is inaccurate when reporting the similarity of two very closely related documents, because of the sampling nature of SE. MDR creates suffix trees and does exact string matches to compare documents.

MDR is more accurate but at the expense of time. It takes the most time and computational power as it must create the suffix trees and then it must compare the suspect tree to the corpus of trees. This requires two sweeps over the data, whereas SE and Scam only require one.

A disadvantage of all three systems is that they do not detect the plagiarism of ideas. There is no measure of the similarity of topics between documents. Neither MDR nor SE would be accurate at detecting plagiarism if the sentence structure had been changed, because the order of words is important to both these systems, while for Scam it does not matter as only the frequency of words is considered.

These methods all work from a corpus of collected documents, but some symptoms of plagiarism are changes in the style of writing and changes in the tense of the writing. Methods could be devised to detect these subtle changes in text to allow possible detection of plagiarism of documents that are not part of the corpus. Synonyms also need to be taken into consideration when checking work, as well as the plagiarism of ideas.

7 References

[Chang, 1994] Chang W. I., Lawler E. L. (1994). Sublinear Approximate String matching and Biological Applications,*Algorithmica12. pp. 327-344.*

[Finkel. 2002] Finkel R. A., Zalsavsky A., Monostori K., Schmidt H.(2002) Signature extraction for overlap detection in documents, *25th Australasian Computer Science Conference (ACSC2002),* 28th Jan-1st Feb, 2002, *Melbourne, Australia.*

[Glatt, 2003]Glatt Plagiarism Screening Program, www.plagrisim.com *Glatt Plagiarism Services, Inc., available 06/06/20003.*

[Monostori,2000] Monostori K., Zaslavsky A., Schmidt H.,(2000) MatchDetectReveal: Finding Overlapping and Similar Digital Documents. *Information Resources management Association international(IRMA2000), 21-24 May, 2000 Anchorage , Alaska,USA.*

[Rivest,1992] Rivest R. L. (1992). *RFC 1321:The MD5 Message-Digest Alogrithm.* Internet Activities Board.

[Shivakumar,1995] Shivakumar N., Garcia-Molina H. (1995). Scam: A Copy Detection Mechanism for Digital Documents, *Proceedings of the 2nd International conference in Theory and Practice of Digital Libraries (DL'95), June11-13, Austin Texas.*

[Shivakumar, 1996] Shivakumar N., Garcia-Molina H., (1996). Building a Scalable and Accurate Copy Detection Mechanism, *Proceedings of the 1st ACM Conference on Digital libraries(DL'96), Bethesda, Maryland.*

[Turnitin, 2003], Turn it in, http://www.turnitin.com *iParadigms digital solutions, avaliabe at 06/06/2003.*