Plagiarism Detection in Text using Vector Space Model

Asif Ekbal* Sriparna Saha* Gaurav Choudhary
Department of Computer Science and Engineering
Indian Institute of Technology Patna
Patna, India-800 013
{*asif,sriparna,c.gaurav*}*@iitp.ac.in*

***Abstract*—Plagiarism denotes the act of copying someone else's idea (or, works) and claiming it as his/her own. Plagiarism detection is the procedure to detect the texts of a given document which are plagiarized, i.e. copied from from some other documents. Potential challenges are due to the facts that plagiarists often obfuscate the copied texts; might shuffle, remove, insert, or replace words or short phrases; might also restructure the sentences replacing words with synonyms; and changing the order of appearances of words in a sentence. In this paper we propose a technique based on textual similarity for external plagiarism detection. For a given suspicious document we have to identify the set of source documents from which the suspicious document is copied. The method we propose comprises of four phases. In the first phase, we process all the documents to generate tokens, lemmas, finding Part-of-Speech (PoS) classes, character-offsets, sentence numbers and named-entity (NE) classes. In the second phase we select a subset of documents that may possibly be the sources of plagiarism. We use an approach based on the traditional Vector Space Model (VSM) for this candidate selection. In the third phase we use a graph-based approach to find out the similar passages in suspicious document and selected source documents. Finally we filter out the false detections[1].**

***Keywords*-Plagiarism detection; Vector Space Model; N-gram language model;**

I. Introduction

There can be many different ways to define plagiarism. One of the most interesting definitions is given by the IEEE (2008):
plagiarism is the reuse of someone else's prior ideas, processes, results, or words without explicitly acknowledging the original author and source.
Sometimes plagiarism occurs accidentally but most of the times it is the outcome of a conscious process. In recent years due to the availability of enormous web documents in the internet, plagiarism has become a huge problem. Plagiarism has now become a major problem in academics, science and commerce. It can be detected manually either by detecting text inconsistencies or by resembling previously consulted material. But due to large volume of scientific texts available it is very difficult to manually detect plagiarism. Thus in recent years there is an urgent need to develop an automatic plagiarism detection system. The motivation of these techniques is detecting potential cases of plagiarism in a suspicious document d_q and, if possible, providing the alleged source(s). With the information provided by the mechanism we can decide whether a text is plagiarized or not.

Automatic plagiarism detection has drawn significant attention to the research communities of Information retrieval (IR) due to its potential commercial applications. There are mainly two different types of plagiarism detection systems available: one being external, the other being intrinsic. In an external plagiarism detection, a set of pure documents will be provided. Each time we need to compare a suspicious document with a given set of reference collections. This requires a document model and a proper similarity metric. The detection task is equivalent to retrieve all documents those contain texts which are similar to a degree above a chosen threshold to texts in the suspicious documents. In case of Intrinsic plagiarism detection there is no need to supply any source document; the suspicious document is analyzed based on the changes in the unique writing style of the author as an indicator for potential plagiarism. The method is very difficult to analyze and needs human judgement to reliably identify plagiarisms.

Some of the existing methods for external plagiarism detection are based on string matching procedure [1], Vector Space Model [2] and fingerprinting [3]. **String matching procedure** for plagiarism detection [1] aims to identify longest pairs of identical text strings. Suffix document models are mostly used for the task. The strength of this procedure is the detection accuracy with respect to the lexical overlaps. One drawback of the algorithm is the relative difficulty of detecting disguised plagiarisms. The method also requires huge computational efforts. **Vector Space Model (VSM)** [2] is a standard technique for traditional Information Retrieval (IR). It can overcome the limitations of string matching technique. Here one or more vectors are used to represent a given document. In order to determine pair-wise similarity between documents, distance between the corresponding vectors is calculated. Traditional cosine similarity measure or more sophisticated similarity functions can be used to determine similarity between two

[1]*: Corresponding Authors.

given vectors. Performance of this model greatly depends on the individual plagiarism incidence to be analyzed and the parameter configuration [4]. The global similarity assessment of VSMs sometimes hampers the overall performance of the system. **Fingerprinting** [1] is one of the most widely applied approaches for plagiarism detection. A representative digests of documents is formed by selecting a set of multiple substrings (n-grams) from them. The sets represent the fingerprints and their elements are called minutiae. For a given suspicious document, its fingerprint is first computed and the minutiae is compared with a pre-computed index of fingerprints for all documents of a reference collection. The inherent challenge of fingerprinting is finding a tradeoff between document dimension and detection accuracy. The reduction in dimension for document representation can cause to the information loss that, in turn, affects system performance. The method requires to tune a number of parameters such as the chunking strategy, chunk size (granularity of the fingerprint) or number of minutiae (resolution of the fingerprint) [5]. Determining the best parameter combination is strongly dependent on the nature and size of the document collection as well as on the amount and the forms of plagiarisms.

Citation-based plagiarism detection [6] is a computer-assisted plagiarism detection approach which can be used in academics. It does not rely on the texts of the given documents but depends on the references given with a particular research paper. It first identifies similar patterns in the citation sequences of two academic works [6]. Subsequent non-exclusively containing citations shared by both documents being compared is represented by using citation patterns. Citation patterns are identified based on the factors of similar order and proximity of citations within the text. In order to quantify the pattern's degree of similarity, some other factors, for example the absolute number or relative fraction of shared citations in the pattern as well as the probability that citations co-occur in a document are considered.
Stylometry [5] applies some statistical methods in order to determine an author's unique writing style. It is mainly used for identifying the authorship attribution or intrinsic plagiarism detection.

In this paper we propose an external plagiarism detection system within the benchmark setup of PAN-11 evaluation challenge. The method we propose comprises of four phases, *viz* pre-processing, document subset selection, passage selection, and filtering out the false positives. The proposed approach is evaluated with the PAN-11 datasets [2].

II. Proposed Approach

We propose an external plagiarism detection system based on the traditional Vector Space Model (VSM) and n-gram language model technique. Our proposed approach comprises of four main steps. In the first step, we process all the documents to generate tokens, lemmas, finding Part-of-Speech (PoS) classes, character-offsets, sentence numbers and named-entity (NE) classes. The documents are passed to the second step in the pipeline. In the second phase we select a subset of documents that may possibly be the candidates of sources of plagiarism. In the third phase we use a graph-based approach to find out the similar passages in suspicious document and selected source documents. Finally we filter out the false detections to improve the performance.

A. Pre-processing

We pre-processed all our selected training and test documents before applying our proposed algorithm. This step is very crucial to filter out the irrelevant information. Some documents contain multi-lingual information. To maintain coherence and uniformity across all the documents we run **text_cat** program on all the source-documents for the purpose of language detection. The English language was chosen as the default language. Currently we discard the non-English documents. We then run **Stanford-core-NLP** [3] suite on all these processed documents for (i). sentence splitting; (ii). tokenisation; (iii). finding character offsets; (iv). determining PoS class for each token; and (v). determining named entity (NE) class for each token. Various kinds of information obtained through this pre-processing stage helps to discard a lot of irrelevant information.

B. Subset selection

The PAN-PC-11 corpora that we want to use for our system contains several thousands suspicious and source documents, where every suspicious document may include plagiarisms of one or more source documents. If we were to compare all of them with each other, we would have no way other than performing millions of document comparisons. A particular document can contain thousands or millions of characters. This results in an unmanageable task. In order to minimize the complexity of this problem the number of document comparisons can be reduced by generating a subset of candidate source documents for every suspicious document. After this subset is determined, some complex function can be applied to detect the plagiarized fragments, if any. To find out the candidate source documents for each suspicious document, we use traditional Vector Space Model (VSM) popularly used in information retrieval (IR). The idea is to represent the documents in terms of vectors. At first a vocabulary is extracted from all the documents. We use lemmas instead of original tokens for generating the vocabulary. Also, stop-words are removed and lemmas belonging to certain Part-of-Speech (PoS) classes are ignored. This is done because words belonging to certain

[2]Please note we did not use all the available training documents for our implementation

[3]nlp.stanford.edu/software/corenlp.shtml

PoS classes do not contribute to similarity measurement, instead they interfere with the process; they are essentially stop-words in this sense. For example, determiners (PoS-tag: DET) like *the* should not be considered for similarity calculation. Others include cardinal number (CD), existential (Ex), preposition (IN), list item marker (LS), predeterminer (PDT), symbol (SYM), to (TO) etc. Each term of the vector is then weighted with term frequency-inverse document frequency, more widely known as tf-idf. Each source and suspicious document thus form a weighted vector. Similarity between a suspicious document and a source document is computed using cosine similarity measure.

The weight $tf - idf$ is calculated for each term t_i in a document d_j:

$$\text{tf-idf}_{i,j} = tf_{i,j}.idf_i$$

where the term frequency of t_i in document d_j, or $tf_{i,j}$, is:

$$tf_{i,j} = n_{i,j}$$

$n_{i,j}$ being the number of occurrences of the term, t_i, in document d_j. The inverse document frequency of term t_i is:

$$idf_i = \frac{1}{|d : t_i \epsilon d|}$$

where $|d|$ represents the number of documents in the collection of source documents that contain the term, t_i (a term that appears in both the suspicious and the source documents).

Finally, the similarity score of two documents, being d_j the suspicious and d_k the source one, will be defined as shown in the following equation:

$$sim_{d_j,d_k} = \frac{\sum_{t_w \epsilon d_j \cup d_k} (\text{tf-idf}_{w,j} \cdot \text{tf-idf}_{w,k})}{\sqrt{\sum_{t_w \epsilon d_j \cup d_k} \text{tf-idf}_{w,j}{}^2 \cdot \sum_{t_w \epsilon d_j \cup d_k} \text{tf-idf}_{w,k}{}^2}}$$

C. Passage selection

Once the candidate source documents are available for a suspicious document, each candidate source document is compared with the suspicious document to find out the plagiarized passages. At first vector of sorted n-grams (tokens sorted lexicographically inside n-gram) are created for each of the source documents and suspicious document. Sorted n-grams are used to detect low obfuscation. Both the n-gram vectors are then sorted lexicographically. The n-grams common to both the vectors are then determined by employing a kind of merge sort algorithm (especially incorporating the merging phase). It is to be noted that we considered trigram (i.e. n=3). Each trigram of suspicious document may have exact matches with many trigrams in source document. This is also known as dot-plot [7]. Thereafter we calculate **Similarity Coefficient** [8] $J(A, B)$ as below:

$$J(A, B) = \frac{|\text{n-grams of A} \cap \text{n-grams of B}|}{min(\text{n-grams of A}, \text{n-grams of B})}$$

If value of the similarity coefficient is greater than a threshold then the plagiarized passages are obtained using a **graph-based technique**. Prior to the construction of graph, common n-grams are sorted according to the suspicious positions of n-grams. The vertex of the graph is a common n-gram. Now the graph is constructed with each vertex containing links to other vertices (common n-grams) whose suspicious position values fall in the range of current vertex suspicious position and some pre-decided distance value. At this stage, we have a complete graph. The passage-detection is done using **depth first search algorithm** on the graph. The algorithm is presented in Figure 1. The algorithm starts with the root and recursively descends down the graph if it finds link to the vertices (common n-grams) with value of suspicious and source position within some pre-defined distance. If while at any vertex, it cannot find next vertex satisfying the previous condition then a check is done to see if the length of this detection is greater than some threshold and that this detection is not a subset of some previous detection. If it satisfies both the conditions then a check is made to see if it can be merged with some adjacent detections using the following heuristics:

- The length of the text in-between two matches, m_i and m_j, is smaller than the length of m_i plus the length of m_j. This applies to the source and the suspicious part of the matches[9].

If it can be merged with some adjacent detection then the merged detection is added to the result otherwise the detection itself is added to the result. After the current call returns, the depth-first search is again run using a vertex with lowest suspicious position that has not been visited so far. Note that in one call to depth-first search algorithm none, one or more than one detection may be found. *This algorithm is exponential in the number of vertices if no restriction is put on number of times we visit the vertices. But we make it into linear in the number of vertices by visiting a node only once.* This hypothesis works here because of the reason that if any vertex has been visited earlier then the start vertex of the first visit must be higher (with lower suspicious position) in the graph than the start vertex of the current visit and hence the current detection is the subset of the previous detection.

D. Filtering of false detections

There could be false detections in passage selection phase. According to our passage selection algorithm we have a detection if we find consecutive common n-grams satisfying the distance limit criteria. So, our approach is a greedy approach which doesn't mind fluctuations from normal plagiarism patterns. Gottron [10] has used a dot-plot technique in which he looks for common sub-sequences within a stripe which has slope of 45°. We use a similar technique but slope in our case is given by ratio of *source-*

length and *suspicious-length* of the detection.

$$\text{slope} = \frac{\text{source-length}}{\text{suspicious-length}}$$

We then compute the distance d_{t_j} of the n-grams t_j which have contributed to the detection from the line AB with slope calculated above. The distance $d(det, AB)$ of detection det from line AB is calculated as:

$$d(det, AB) = \frac{\sum_{t_j \epsilon N_{det}} d_{t_j}}{|N_{det}| \cdot min(\text{source-length, suspicious-length})}$$

N_{det} refers to n-grams which have contributed to the detection det.
The detections with distance $d(det, AB)$ more than a threshold value are discarded. Also, the detections with $|N_{det}|$ less a fixed threshold are discarded.

III. Results and Discussions

A. Datasets

The datasets we used for our experiments is PAN plagiarism corpus 2011 (PAN–PC–11) [11] which contains a large and diverse set of artificial and simulated plagiarism cases. For the sake of clarity and proper explanation many texts of the particular subsections A and B are directly taken from [11]. To create artificial plagiarism, three obfuscation strategies were used. Given a source passage s_{src}, a plagiarized passage s_{plg} is created as follows:

- **Random Text Operations.** s_{plg} is created from s_{src} by shuffling, removing, inserting, or replacing words or short phrases at random. Insertions and replacements are taken from the document d_{plg} where s_{plg} is to be inserted.
- **Semantic Word Variation.** s_{plg} is created from s_{src} by replacing words by one of their synonyms, antonyms, hyponyms, or hypernyms, chosen at random. A word is kept if none of them is available.
- **PoS-preserving Word Shuffling.** The sequence of PoS in s_{src} is determined and s_{plg} is created by shuffling words at random while retaining the original PoS sequence.

The documents used in the data (or, corpus) are derived from books from the Project Gutenberg. Every document in the corpus serves one of the two purposes: it is either used as a source for plagiarism or as a document suspicious of plagiarism. The latter set of documents are divided into documents that actually contain plagiarism and documents that don't. The documents without plagiarism allow to determine whether or not a detector can distinguish plagiarism cases from overlaps that occur naturally between random documents.
The fraction of plagiarism per document, the lengths of the documents and plagiarism cases, and the degree of obfuscation per case determine the difficulties: the corpus contains short documents with a short, un-obfuscated plagiarism case, resulting in a 5% fraction of plagiarism, but it also contains large documents with several obfuscated plagiarism cases of varying lengths, drawn from different source documents and resulting in fractions of plagiarism up to 100%. Since the true distributions of these parameters in real plagiarism are unknown, sensible estimations were made for the corpus. E.g., there are more simple plagiarism cases than complex ones, where simple refers to short cases, hardly plagiarism per document, and less obfuscation.
The PAN-PC-11 training corpus for external plagiarism comprises of 11,093 source documents and 11,093 suspicious documents. The PAN-PC-11 test corpus comprises of 11,093 source documents and 11,093 suspicious documents. The documents are mostly in English but there are German and Spanish source documents also.

B. Evaluation Metrics

The system is evaluated in terms of precision, recall, granularity and plagdet score defined as below [11]:

Precision signifies what fraction of the detection cases detected by system are plagiarism cases (true cases) and **Recall** signifies what fraction of the plagiarism cases (true cases) are detected by the system. The macro-averaged precision and recall are unaffected by the length of a plagiarism case; they are defined as follows:

$$prec_{macro}(S, R) = \frac{1}{|R|} \sum_{r \epsilon R} \frac{|\cup_{s \epsilon S} (s \cap r)|}{|r|}$$

$$rec_{macro}(S, R) = \frac{1}{|S|} \sum_{s \epsilon S} \frac{|\cup_{r \epsilon R} (s \cap r)|}{|s|}$$

where S denotes the set of plagiarism cases in the corpus, and R denotes the set of detections reported by a plagiarism detector for the suspicious documents. To simplify the notation, a plagiarism case $s = h_s plg; d_p lg; s_s rc; d_s rc,\ s \in S$, is represented as a set s of references to the characters of $d_p lg$ and $d_s rc$, forming the passages $s_p lg$ and $s_s rc$. Likewise, a plagiarism detection $r \in R$ is represented as r. Besides precision and recall there is another concept that characterizes the power of a detection algorithm, namely, whether a plagiarism case $s \epsilon S$ is detected as a whole or in several pieces. Ideally, an algorithm should report detections R in a one-to-one manner to the true cases S. To capture this characteristic, a measure known as detection granularity of R under S is used which is defined as:

$$gran(S, R) = \frac{1}{|S|} \sum_{s \epsilon S_R} |R_s|$$

where $S_R \subseteq S$ are cases detected by detections in R, and $R_s \subseteq R$ are the detections of a given s The domain of $gran(S, R)$ is $[1, |R|]$, with 1 indicating the desired one-to-one correspondence and $|R|$ indicating the worst case, where a single $s \epsilon S$ is detected over and over again.

```
Step 1:  Compute sorted n-grams on processed suspicious and source documents.
Step 2:  Sort the n-grams created in Step 1 and Step 2 lexicographically.
Step 3:  Find common n-grams (dot-plot) using merge phase of merge-sort algorithm
         and sort the n-grams w.r.t. suspicious position.
Step 4:  Using the common n-grams obtained in Step 3, compute the n-gram-match-ratio.
         If n-gram-match ratio is greater than threshold then construct the graph.
         Step 4.1: Add to vertex list all the common n-grams.
         Step 4.2: Create links from each of the vertices to other vertices having suspicious distances
         less than Suspicious Length. Mark all the links as unvisited.
Step 5:  Find the first vertex which has not been visited and apply depth-first-search (see
         section 3.3) starting at this vertex. When the control returns from the child node,
         repeat this step until all nodes get marked. If all nodes got marked goto Step 10.
Step 6:  If the vertex pointed to by next link lies within Suspicious Length and Source Length
         go to this vertex and update the source passage length and suspicious passage length.
         Else try next link and repeat this step again.
Step 7:  If there is no link within Suspicious Length and Source Length,
         if source passage length is greater than threshold and suspicious passage length is
         greater than threshold then Go to step 8, else return to the parent node.
Step 8:  Check all the previous detections if the current detection is not a subset of previous
         detections, go to Step 9 else return to the parent node.
Step 9:  Check if this detection is adjacent to some previous detection. If it is so,
         merge the two detections and insert the merged detection to the result. Otherwise,
         insert the detection as it is. Return to parent node.
Step 10: For each detection perform the following:
         Step 10.1: If the distance $d(det, AB)$ of detection $det$ from line $AB$ is greater than
         threshold, then discard this detection. (see section 3.4 for details)
```

Figure 1. Main steps of our passage detection phase and filtering phase

Table I
RESULTS WITH BIGRAM APPROACH

Recall	Precision	Granu.	Pladget Score	Suspici. Distance	Source Distance	J(A,B)
24.15	2.10	1.38	3.08	1,500	4,000	0.01
26.93	2.44	2.04	2.80	1,000	4,000	0.01
22.83	2.76	1.00	4.93	1,000	4,000	0.004
25.80	2.80	1.02	5.01	1,500	3,000	0.001

The three measures are combined into a single overall score as follows:

$$plagdet(S, R) = \frac{F_1}{log_2(1 + gran(S, R)}$$

C. Results and Discussions

We perform experiments with the benchmark set up of PAN-11. Experiments were carried out in a machine with 16 GB RAM that runs at a frequency of 2.53 GHz. As the overall process involves huge computations we consider only a subset of 1,000 suspicious documents of the PAN-PC-11 training corpus. This is treated as the development set and the best configurations are determined from this. We present the results with bigram model (i.e. n=2) in Table I. Results show very low precision and so we filter the false positives (c.f. Section II-D). Results of this approach are reported in Table II. The J(A,B) value is fixed at 0.001 for all the later experiments. It clearly reveals improvement in precision by 15 points. The *granularity* comes to the perfect value of 1.0. But, there is a drop in recall. However, we need to bear this tradeoff.

Table II
RESULTS OF BI-GRAM MODEL WITH REMOVAL OF FALSE POSITIVES

Rec.	Preci.	Granu.	Pladget Score	Sus. Dist	Sour. Dist	$d(det, A, B)$	$\|N_{det}\|$
15.80	12.40	1.00	13.90	2,000	2,500	0.20	3
12.95	14.12	1.00	13.51	1,500	2,000	0.20	3
18.10	16.14	1.00	17.60	1,500	2,000	0.30	3

Thereafter we carried out experiments with tri-grams, and its results are shown in Table III. We observed a rapid growth of 30 points in precision by removing false positives.

Table III
RESULTS OF TRIGRAM MODEL AFTER REMOVAL OF FALSE POSITIVES

Rec.	Prec.	Gran.	Plad. Scor.	Sus. Dist	Sour. Dist	d(det,A,B)	$\|N_{det}\|$
18.74	63.54	1.00	28.94	2000	2500	0.20	3
22.67	64.40	1.00	31.77	2000	2500	0.30	3
17.80	68.80	1.00	28.26	3000	5000	0.20	3
23.25	52.60	1.00	32.25	3000	5000	0.30	3

On the basis of above experiments, we set the parameter values as follows:

1) J(A,B) = 0.001
2) Suspicious Length: 3,000
3) Source Length: 5000
4) $d(det, A, B)$ = 0.30

5) $|N_{det}| = 3$

Thereafter we evaluate the system on the test set obtained are:

- *Plagdet Score* = 28.91
- *Recall* = 19.04
- *Precision* = 65.93
- *Granularity* = 1.03

Comparisons with the state-of-the-art systems show that our proposed approach can attain performance comparable to some of the top-performing few systems. However, it is to be noted that we were not able to use all the available documents due to our hardware limitations. Moreover, our proposed approach makes use of a smaller feature set compared to the other higher ranked systems.

IV. Conclusion

In this paper we proposed a Vector Space Model based approach for external plagiarism detection. Our proposed approach shows encouraging performance with a benchmark setup. The corpus we have used contain documents of English, German and Spanish. There are cross-lingual plagiarism cases. Since we have not used language translation, our system cannot detect these cases. As a result, the system shows a lower recall value. In future we would like to employ Google translator. Our algorithm does not detect well the cases of high obfuscation. Our algorithm which is based on n-grams cannot detect plagiarized cases where common n-grams are not enough. One way to solve this problem can be to use fuzzy similarity measure to compare sentences[12]. We propose an optimization over this method. Instead of matching fuzzy similarity over sentences, 10-grams can be made and then similarity between n-grams can be calculated using fuzzy similarity approach. WordNet can be used to deal with semantic word variations [11] in which plagiarized passage is created from source passage by replacing words by one of their synonyms, antonyms, hyponyms, or hypernyms, chosen at random.

References

[1] Z. Su, B.-R. Ahn, K.-Y. Eom, M.-K. Kang, J.-P. Kim, and M.-K. Kim, "Plagiarism detection using the levenshtein distance and smith-waterman algorithm," in *Proceedings of the 2008 3rd International Conference on Innovative Computing Information and Control*, ser. ICICIC '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 569–. [Online]. Available: http://dx.doi.org/10.1109/ICICIC.2008.422

[2] J. Kasprzak and M. Brandejs, "Improving the reliability of the plagiarism detection system-lab report for pan at clef 2010," in *Notebook Papers of CLEF 2010 LABs and Workshops*, 2010.

[3] T. C. Hoad and J. Zobel, "Methods for identifying versioned and plagiarized documents," *J. Am. Soc. Inf. Sci. Technol.*, vol. 54, no. 3, pp. 203–215, Feb. 2003. [Online]. Available: http://dx.doi.org/10.1002/asi.10170

[4] M. A. Gutbrod, "Nachhaltiges e-learning durch sekundre dienste," *Technischen Universitt Braunschweig Institut fr Betriebssysteme und Rechnerverbund*.

[5] S. Meye Zu Eissen and B. Stein, "Intrinsic plagiarism detection," in *Advances in Information Retrieval 28th European Conference on IR Research, ECIR 2006*, 2006, pp. 565–569.

[6] B. Gipp and N. Meuschke, "Citation pattern matching algorithms for citation-based plagiarism detection: greedy citation tiling, citation chunking and longest common citation sequence." in *ACM Symposium on Document Engineering*, M. R. B. Hardy and F. W. Tompa, Eds. ACM, 2011, pp. 249–258. [Online]. Available: http://dblp.uni-trier.de/db/conf/doceng/doceng2011.htmlGippM11

[7] K. W. Church and J. I. Helfman, "Dotplot: a Program for Exploring Self-Similarity in Millions of Lines of Text and Code." [Online]. Available: http://imagebeat.com/dotplot/rp.jcgs.pdf

[8] R. M. A. Nawab, M. Stevenson, and P. Clough, "University of sheffield - lab report for pan at clef 2010," in *CLEF (Notebook Papers/LABs/Workshops)*, 2010.

[9] D. Micol, scar Ferrndez, F. Llopis, and R. Muoz, "A textual-based similarity approach for efficient and scalable external plagiarism analysis - lab report for pan at clef 2010," in *CLEF (Notebook Papers/LABs/Workshops)'10*, 2010, pp. –1–1.

[10] T. Gottron, "External plagiarism detection based on standard ir technology and fast recognition of common subsequences - lab report for pan at clef 2010." in *CLEF (Notebook Papers/LABs/Workshops)*, M. Braschler, D. Harman, and E. Pianta, Eds., 2010. [Online]. Available: http://dblp.uni-trier.de/db/conf/clef/clef2010w.htmlGottron10

[11] M. Potthast, B. Stein, A. Barrón-Cedeño, and P. Rosso, "An evaluation framework for plagiarism detection," in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, ser. COLING '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 997–1005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1944566.1944681

[12] S. Alzahrani and N. Salim, "Fuzzy semantic-based string similarity for extrinsic plagiarism detection - lab report for pan at clef 2010," in *CLEF (Notebook Papers/LABs/Workshops)*, 2010.