TEXT PLAGIARISM CHECKER USING FRIENDSHIP GRAPHS

Soumajit Adhya[1] and S.K. Setua[2]

[1]Department of Management, J.D. Birla Institute, Kolkata, India
[2]Dept. of Computer Science, University of Calcutta, Kolkata, India

ABSTRACT

The paper proposes a method to check whether two documents are having textual plagiarism or not. This technique is based on Extrinsic Plagiarism Detection. The technique that is applied here is quite similar to the one that is used to grade the short answers. The triplets and associated information are extracted from both texts and are stored in friendship matrices. Then these two friendship matrices are compared and a similarity percentage is calculated. This similarity percentage is used to take decisions on the plagiarized paper. This technique can detect copy paste plagiarism and disguised plagiarism.

KEYWORDS

Friendship Graph, Friendship Matrix, Original Text, Candidate Text, Triplets, Textual Plagiarism

1. INTRODUCTION

Plagiarism means using another person idea or words and claiming as his own idea or words. [8] Plagiarism is a big problem in today's world. It hampers the academic integrity of an article. Many authors of various journals are resorting to this unethical means and publishing their journals. So in order to curb this problem various software tools are developed. [7]

There are various kinds of plagiarism like Copy Paste Plagiarism, Disguised Plagiarism, Plagiarism by Translation, Shake and Paste Plagiarism, Structural Plagiarism, Mosaic Plagiarism, Metaphor Plagiarism, Idea Plagiarism, and Self Plagiarism. In Copy Paste Plagiarism the candidate copies from the source text directly. In Disguised Plagiarism the candidate copies from the source and changes some words or letters. In Plagiarism by Translation the candidate translates the text from one language to another. In Shake and Paste Plagiarism the candidate copies the text from various paragraphs and they are well written and they are not in functional order. Structural Plagiarism deals with idea of persons, the arguments order, the footnotes, selection of certain quotations. Mosaic Plagiarism refers to getting the content from various sources and rephrasing the sentences, changing words and using synonyms. Metaphor plagiarism happens when author's creative style is stolen. Idea Plagiarism occurs when someone steals somebody's original innovative idea or solution and uses it as his/her own idea. Self Plagiarism occurs when author reuses his/her own work. [8]

The Plagiarism Detection Methods (PDM) are: (a) Extrinsic PDM and (b) Intrinsic PDM. In Extrinsic PDM requires Reference Text(s) and Intrinsic PDM do not require Reference Text(s) [8].

In Extrinsic PDM the techniques are: Grammar Based PDM, Semantic Based PDM, Cluster Based PDM, Cross Lingual Based PDM, Citation Based PDM, Character Based PDM. In

DOI:10.5121/ijcsit.2016.8402

Grammar Based PDM uses string based matching approach to detect and measure similarity between the documents available in the database. The Semantic Based PDM uses a vector space model to determine the similarities in the use of words between documents stored in the database. The Cluster Based PDM is similar to Grammar Based PDM but it has 3 steps namely, pre selecting to narrow the scope using same successive fingerprint, locating the fragments and merging them and post processing to calculate merging errors. Cross Lingual PDM is used for Plagiarism by translation. Citation Based PDM is a type of Semantic Based PDM used for identifying similarity in citation sequences in academic journals. The Character Based PDM uses the concept of Fingerprinting and String Matching. [8]

In Intrinsic PDM the techniques are: Grammar Semantics Hybrid PDM, Structure Based PDM, and Syntax PDM. In Grammar Semantics Hybrid PDM uses NLP Techniques and can detect Mosaic Plagiarism. In Structure Based PDM focuses on structure features of Text, and in Syntax PDM also called Syntax Similarity PDM focuses on syntactical structure like Part of Speech Tagger (POS). [8]

This paper proposes a system which can be used for checking Textual Plagiarism (TP) using computer system. This paper proposes a method by which the document has Textual Plagiarism (TP) can be checked. It proposes a system which compares friendship matrices of Original Text (OT) and Candidate Text (CT) and accordingly provides the similarity percentage. This algorithm enters one sentence at a time from OT and stores it in a friendship matrix. Similarly the CT is stored in another friendship matrix and is compared with the OT to check how many sentences exactly match and then the similarity percentage is calculated.

In this paper Section 2 deals with related work, Section 3 deals with terminologies associated with this paper, Section 4 deals with the problem definition, and Section 5 deals with the proposed method for TP Checker and Section 6 is the Conclusion.

2. RELATED WORK

Commonly Algorithms used for Automated Essay Grading are used for checking textual plagiarism. Maurer et al. (2006) describes 3 ways of plagiarism checking. First is the language independent way which compares word to word with the selected set of target documents which are the sources of copied materials. Second is quite similar to document check but here the target document is the set of all documents that is reachable on Internet and candidate document is searched for characteristic text or sentence. The third type is stylometry in which a language analysis algorithm is used to compare the style of different paragraphs and report if a style change has occurred. This requires a prior analysis of candidate's previous documents. [7]
The WCopyFind uses Text String matches to find plagiarism. The EVE2 examines the essays then makes quick search to possible sites from which the text might be copied. The Normalized Word Vector (NWV) developed in 2006 was used for Automated Essay Grading is also used for plagiarism checker. In this technique the semantic footprint of original text is compared to the mathematical representation of candidate text and it is graded. Similarly the semantic footprint of the candidate text can be calculated and plagiarism can be checked by footprint comparison. [7]

3. TERMINOLOGY

3.1. FRIENDSHIP GRAPH AND FRIENDSHIP MATRIX

A graph is called a friendship graph if every pair of its nodes has exactly one common neighbor (Refer to Figure 1). This condition is called the friendship condition [2]. This graph is used to

model the subject-predicate-object structure of a sentence. A friendship matrix is the relational or tabular structure of the friendship graph. In this case the common node (neighbor) is associated with the other nodes i.e. subjects and objects. The relation or table name is the common node and the subjects & objects are the nodes that are associated with common node (Refer to Table 1) [6].

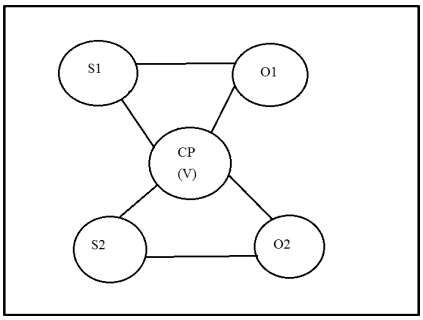

Figure 1: A friendship graph

Table Name: V (the common node)

Table 1: The Friendship Matrix which is derived from the Friendship Graph

SUBJECT	**PREDICATE**
S1	O1
S2	O2

The friendship graph structure is an ideal structure to store the RDF triplets. The common node can be used as the common MAIN PREDICATE, with the neighboring connected nodes as Subjects and Objects. Now this friendship graph structure can be saved in a matrix form by making the common MAIN PREDICATE as the TABLE NAME with PRE PREDICATE, SUBJECT and OBJECT as the column names (Refer to Table 2)[6].

Table 2: A friendship matrix to represent the above friendship graph

PRE PREDICATE	**SUBJECT**		**OBJECT**		
	PRE SUBJECT	MAIN SUBJECT	PRE OBJECT	MAIN OBJECT	POST OBJECT

Since the number of friendship matrix tables is going to be large so there will be a lot of overhead costs. So instead of maintaining individual friendship matrices we can save it in a single table (Refer to Table 3). [6]

Table 3: Final Friendship Matrix to store all the subject-predicate-object triplets derived from candidate and original text

Common main Predicate	**Corresponding Table**					
Predicate1	Pre Predicate	Subject		Object		
		Pre Subject	Main Subject	Pre Object	Main Object	Post Object

3.2. SENTENCE EXTRACTION

The sentence extraction algorithm extracts individual sentences from the document. We assume that the individual sentences are separated by full stops. By analyzing the full stops the sentence is extracted and each individual sentence is passed to the Co Reference Resolution pass one after another [4].

3.3. NATURAL LANGUAGE PROCESSING TECHNIQUES [4]

a. Co Reference Resolution (CRR) Pass: In this pass all the entities are recognized which are single words or a block of sequential words. Entities refer to any name, place etc. CRR attempts to find the words in a sentence that refers to an entity and replaces these references with the target entity. Then this modified sentence is passed to tokenization and parts of speech tagger.

b. Tokenization and Part of Speech Tagger: Each Sentence is tokenized and part of speech is tagged for each and every word. Then this tokenized sentence is sent to Full parsing phase.

c. Full parsing phase: In this case the sentence is written in Penn Tree Bank style which shows the phrasal structure and attachments. The nesting level is denoted by using tabs. Then this parse tree is sent for split coordinating conjunction phase.

d. Split Coordinating Conjunction Phase: Complex sentences are broken into simple sentences based on conjunction..

e. Extract Dependent Clauses: Sentences with dependent clauses, known as complex sentences in linguistics—as opposed to simple sentences with a single clause—are common in text. A dependent clause is introduced by either a subordinate conjunction (for adverbial clauses) or a relative pronoun (for relative clauses), so those two cases have to be handled differently. This pass also extracts parenthesized phrases and clauses as they can be handled similarly, although not all are technically dependent clauses. Adverbial clauses are extracted into modifiers, whereas relative or parenthesized clauses are broken off into separate sentences.

f. Extract Adjective Phrases: Adjective phrases typically appear in sentences between one or two commas, and appear in the parse tree as nested under their subject.

g. Extract Prepositional Clauses: Prepositional Phrases are the main type of adjunct that is converted into a triple modifier. Because the attachments of modifiers are ignored by this system, attachments don't need to be captured.

h. Lemmatization: Reducing the verbs to their base form.

i. Synonym Conversion: All synonyms are checked from synonym table and converted into base word.

4. PROBLEM DEFINITION

This paper proposes an algorithm by which two texts are similar can be compared and a similarity percentage can be calculated. It also provides a framework by which the subject, predicate and object of each sentence can be extracted so that the semantic meaning of each sentence is not lost. This algorithm can detect copy paste plagiarism and disguised plagiarism.
The technique is based on Extrinsic PDM.

5. PROPOSED ALGORITHM

This paper proposes an algorithm for checking TP. The CT is compared with the OT. Both the texts which are written in paragraph forms are converted into friendship matrix form and then the two matrices are compared. Based on number of matches of tuples a similarity percentage is calculated.

Each sentence of OT is extracted which is generally a complex sentence and is sent to NLP Converter. NLP converter converts the complex sentence which is extracted from the OT into simple sentence(s). Then each simple sentence is passed to OTripletExtractor to create the OT Friendship Matrix (Refer to Figure 2).

Each sentence of CT is extracted which is generally a complex sentence and is sent to NLP Converter. NLP converter converts the complex sentence, extracted from the CT into simple sentence(s). Then each simple sentence is passed to CTripletExtractor to create the CT Friendship Matrix (Refer to Figure 2).

The Comparator will be applied to compare and to find the number of matches of tuples between the OT friendship matrix and CT friendship matrix. Based on number of matches the similarity percentage is calculated (Refer to Figure 3) . Every unmatched tuples or part of tuples of CT friendship matrix and OT friendship matrix is treated as errors. There are 4 types of errors i.e.

Error1: Error due to missing words in pre subject, pre object, and post object for matching subject/object
Error2: Error due to object of candidate text not found in original text for a matching main subject.
Error3: Error due to main subject of candidate text not found in original text for a matching common predicate.
Error4: Error due to predicate of candidate text not found in original text

To convert a complex sentence to simple sentence the following NLP techniques are used in order:

CRR, Tokenization and Parts of speech tagger, Full parsing, Split Coordinating conjunction, Extract Dependent Clauses, Extract Adjective Clauses, Extract Prepositional Clauses, lemmatization and Synonym Conversion[1][3][4].

The overall method is formalized as below:

Sentence Extraction (*Text*)
{
Extract the sentences from the text one by one.
}

NLP Converter (*A_Complex_Sentence)*
{
Use the existing NLP techniques to convert the complex sentences to simple sentences in parse tree form for each and every sentence of the text.
}

OTripletExtractor (*A_Simple_Sentence)*
{

Step 1: Find the deepest verb from the Verb Phrase (VP) sub tree of the parse tree and match it in the predicate field. If the matching predicate is not found then add that predicate to the friendship matrix and go to Step 2. If the matching predicate is found then go to Step 2.

Step 2: While finding the deepest verb all the nodes that are encountered from the parse tree in the VP sub tree of the parse tree are combined to form a string and store it in the pre predicate field with corresponding to that common predicate which was found in Step 1.

Step 3: Find the first noun from the Noun Phrase (NP) sub tree of the parse tree and store it in the main subject field with corresponding to that common predicate which was found in Step 1. While finding the first noun all the nodes that are encountered from the parse tree are combined to form a string and stored in the pre subject column.

Step 4: Find the first adjective, noun or pronoun from the VP sub tree of the parse tree and stored as object with corresponding to that common predicate which was found in Step 1.While finding the first noun/adjective/pronoun all the nodes that are encountered from the parse tree are combined to form a string and stored in the pre object column and other nodes which followed object are to form a string and stored in the post object column.

}
CTripletExtractor (*A_Simple_Sentence)*
{

Step 1: Find the deepest verb from the Verb Phrase (VP) sub tree of the parse tree and match it in the predicate field. If the matching predicate is not found then add that predicate to the friendship matrix and go to Step 2. If the matching predicate is found then go to Step 2.

Step 2: While finding the deepest verb all the nodes that are encountered from the parse tree in the VP sub tree of the parse tree are combined to form a string and store it in the pre predicate field with corresponding to that common predicate which was found in Step 1.
Step 3: Find the first noun from the Noun Phrase (NP) sub tree of the parse tree and store it in the main subject field with corresponding to that common predicate which was found in Step 1. While finding the first noun all the nodes that are encountered from the parse tree are combined to form a string and stored in the pre subject column.

Step 4: Find the first adjective, noun or pronoun from the VP sub tree of the parse tree and stored as object with corresponding to that common predicate which was found in Step 1. While finding the first noun/adjective/pronoun all the nodes that are encountered from the parse tree are combined to form a string and stored in the pre object column and other nodes which followed object are to form a string and stored in the post object column.
}

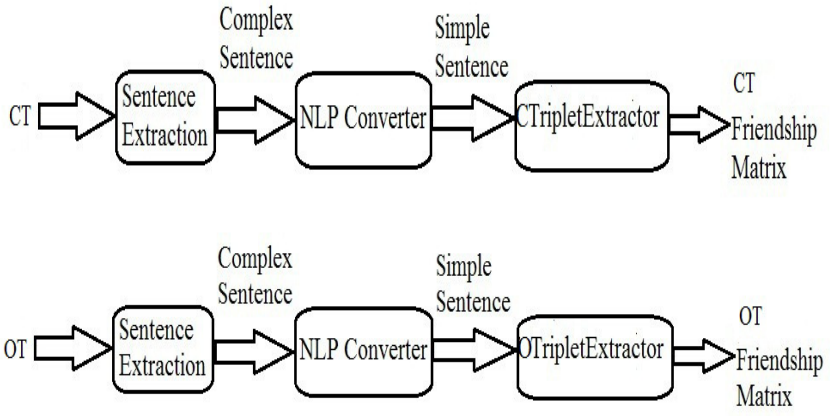

Figure 2: Process of Converting OT and CT into respective friendship matrix

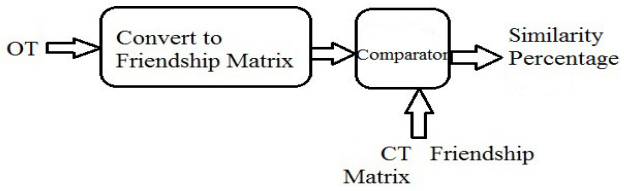

Figure 3: Calculating Similarity Percentages

Comparator (*OT_Friendship_Matrix, CT_Friendship_Matrix)*

Abbreviations:

SP=Similarity %
TL = Total lines of an text
EM = Total Error %
W = Average number of words present in pre subject, pre object, post object
ECC = Error Column Count
EL= Error_Count
ER1 = Total Error % occurred due to missing words in pre subject, pre object, post object for matching subject/object
ER = Total Error % occurred due to object of candidate text not found in original text for a matching main subject, OR main subject of candidate text not found in original text for a matching common predicate OR predicate of candidate text not found in original text

{
Step 1: Initialization of variables error_col_count, error_count.

Step 2: Take a common predicate from CT friendship matrix and match with the common predicate of the OT friendship matrix. If the common predicate is not found then go to Step 4 else go to Step 3.

Step 3: For each main subject for the matching common predicate from CT friendship matrix repeat from Step 3.1 to Step 3.6. If all main subjects for the matching common predicate is over then go to Step 2

Step 3.1: Take a main subject from the CT friendship matrix for the common matching predicate.

Step 3.2: Match main subject from the CT friendship matrix with the main subject of the OT friendship matrix for the corresponding matching predicate one at a time. If a match is found then go to Step 3.3 else go to Step 3.7

Step 3.3: Match the words present in the pre subject field to that corresponding matching main subject of the CT friendship matrix with the pre subject field to that corresponding matching main subject of the OT friendship matrix. For each unmatched word increase the error_col_count by 1 and go to Step 3.4

Step 3.4: Take the main object of the CT friendship matrix to that corresponding main subject and match it with the main object of the OT friendship matrix to that corresponding main subject. If match is found then go to Step 3.5 else increase the error_count by 1 and go to Step 3.7

Step 3.5: Match the words present in the pre object field to that corresponding matching main object of the CT friendship matrix with the pre object field to that corresponding matching main object of the OT friendship matrix. For each unmatched word increase the error_col_count by 1 and go to Step 3.6

Step 3.6: Match the words present in the post object field to that corresponding matching main object of the CT friendship matrix with the post object field to that corresponding matching main object of the OT friendship matrix. For each unmatched word increase the error_col_count by 1 and go to Step 3.1

Step 3.7: If no match is found increase the error_count by 1 and go to Step 3

Step 4: Increase the error_count by 1. If all the common predicates are exhausted then go to Step 5 else go to Step 2

Step 5: Calculation based on errors.

SP = 100-EM
where,
ER1 = (ECC/W)*100
ER = (EL/TL)*100
EM = ER1 + ER

}

6. CONCLUSIONS

This paper proposes an algorithm to check textual plagiarism using the algorithm devised for short answer grading. It uses an extrinsic PDM. In this case the CT is taken and converted into the CT friendship matrix. The OT is taken and converted into OT friendship matrix . Both the matrices are compared and a similarity percentage (SP) is calculated. SP depends on unmatched tuples. So, this algorithm is very useful to detect Copy Paste Plagiarism and Disguised Plagiarism. This technique can be enhanced further to detect Shake Hand Plagiarism and Mosaic Plagiarism.

REFERENCES

[1] Delia Rusu, Lorand Dali, Blaž Fortuna, Marko Grobelnik & Dunja Mladenić, (2007) "Triplet Extraction from Sentences," Proceedings of the Conference on Data Mining and Data Warehouse (SiKDD 2007) held at 10th International Multi conference on Information Society

[2] Endre Boros, Vladimir A. Gurvich & Igor E. Zverovich, (2008) DIMACS Technical Report, 1 RUTCOR, Rutgers Center for Operations Research Rutgers, The State University of New Jersey

[3] Jonathan Hayes and Claudio Gutierrez, (2004) "Bipartite Graphs as Intermediate Model for RDF", The Semantic Web – ISWC 2004, Springer Berlin Heidelberg, Vol. 3298, pp 47 – 61.

[4] Aaron De Fazio, (2009) "Natural Question Answering over Triple Knowledge Bases", Australian National University

[5] Steven Burrows, Iryna Gurevych & Benno Stein, (2015) "The Eras and Trends of Automatic Short Answer Grading," International Journal of Artificial Intelligence in Education25, IOS Press, p. 60-117

[6] Soumajit Adhya, S.K.Setua,(2016) "Automated Short Answer Grader Using Friendship Graphs", Computer Science and Information Technology-Proceedings of The Sixth International Conference on Advances in Computing and Information Technology (ACITY 2016) , Vol. 6 No. 9, pp 13-22

[7] Heinz Dreher, (2007) "Automatic Plagiarism Detection Using Conceptual Analysis", Issues in Informing Science and Information Technology, Vol. 4, pp. 601-614

[8] Ramesh R. Naik, Maheshkumar B. Landge, C. Namrata Mahender, (2015) "A Review on Plagiarism Detection Tools", International Journal of Computer Applications (0975 – 8887), Vol. 125 No. 11, pp. 16-22

AUTHORS

Soumajit Adhya` holds a M.Sc degree in Computer and Information Science from University of Calcutta and currently employed as a IT faculty in JDBI, Department of Management.

S.K.Setua is an Associate Professor in the Department of Computer Science & Engineering at University of Calcutta. His research interest includes distributed computing, information & network security, big data analytics, SDN. He has more than 50 research publications in international journals and conferences.