International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 5– No.2, January 2013 – www.ijais.org

Plagiarism Detection using Sequential Pattern Mining

Ali El-Matarawy
Faculty of Computers and Information, Cairo University

Mohammad El-Ramly
Faculty of Computers and Information, Cairo University

Reem Bahgat
Faculty of Computers and Information, Cairo University

ABSTRACT
This research presents a new technique for plagiarism detection using sequential pattern mining titled EgyCD. Over the last decade many techniques and tools for software clone detection have been proposed such as textual approaches, lexical approaches, syntactic approaches, semantic approaches …, etc. In this paper, the research explores the potential of data mining techniques in plagiarism detection. In particular, the research proposed a plagiarism technique based on sequential pattern mining (SPM), words/statements are treated as a sequence of transactions processed by the SPM algorithm to find frequent itemsets. The research submits an experiment to discover copy/paste in the text source and it gave good results in a reasonable and acceptable time.

Keywords
Plagiarism Detector, Plagiarized Clones, Textual Approach, Lexical Approach, Syntactic Approach, Data Mining, Apriori Property, Sequential Pattern Mining.

1. INTRODUCTION
When the work of someone else is reproduced without acknowledging the source, this is known as plagiarism [1]. Probably the most frequent cases appear in academic institutions where students copy material from books, journals, the Internet, their peers etc. without citing references. Although sometimes intentional, there are many cases where students actually plagiarize unintentionally simply because they are not aware of how sources should be used within their own work. This problem is not just limited to written text, but also regularly found in software code where chunks are copied and re-used without reference to the original author/s [2].

Computer technology spreads too fast, and hence everyone can see easily that using computer is in everywhere especially in schools, colleges and universities. Most of student's work assignments are now expected to be submitted in electronic form. Although convenient and easier for both student and lecturer alike, the electronic version provides the student with an easier opportunity to plagiarize. With advanced word processors it is much easier to cut and paste large amounts of text to create a single work from a number of electronic sources including the Internet, electronic journals, books, newspapers and magazines etc [2].

Helping to make access to electronic versions of written text easier is the Internet. The Internet is growing at a remarkable rate and fast becoming a common resource for students. A recent study by IBM, Compaq and Alta Vista involved analyzing more than 600 million unique pages1 across a wide variety of subjects. It is probably true to say that a search on the Internet today for even the most obscure of topics will almost certainly return some relevant result. The Internet provides a global resource accessible by anyone from anywhere in the World that makes keeping track of electronic documents much harder than ever before and plagiarism much easier. However, the teacher's foe can also be their friend [2]. Using Internet search engines such as Google, Alta Vista and Yahoo, teachers can search for "unusual" phrases they find in student's work to identify potential sources [1].

As mentioned in [3] there are Four Categories of Plagiarism:-

a. **Unauthorized and/or unacknowledged collaborative work:** While students are expected to do their own research and writing, instructors also understand that students may discuss their own research projects with other students in the same course. Instructors strongly suspect collaborative plagiarism when the same or similar phrases, quotations, sentences, and/or parallel constructions appear in two or more papers on the same topic. To protect yourself, you should acknowledge—in a footnote or endnote—any significant discussions you have had with others, as well as any advice, comments, or suggestions that you have received from others, including your instructor or other instructors if appropriate.

b. **Attempting to pass off, as your own work, a whole work or any part of a work belonging to another person, group or institution:** This includes borrowing, buying, commissioning, copying, receiving, downloading, taking, using, and/or stealing a paper that is not your own. Submitting an entire work which is not your own also constitutes research or academic fraud.

c. **The use of any amount of text that has been improperly paraphrased constitutes plagiarism.** Suggesting an improper reliance on a single source, this includes "mosaic plagiarism" or "cut-and-paste plagiarism."

d. **The use of any amount of text, that is properly paraphrased—but which is either not cited or which is improperly cited—constitutes plagiarism.** This includes papers in which a general failure to cite sources or a gross negligence in citing sources is apparent. Moreover, attaching false, misleading, or improper attributions/citations to properly paraphrased texts still constitutes plagiarism.

The rest of this paper is organized as following: some related work on plagiarism detection in section 2. In sections 3, an overview for data mining and its techniques are introduced, particularly the ones relevant to plagiarism detection. In sections 4, 5 and 6, the research introduces the new approach for detecting plagiarism. A case study is reported in section 7. Section 8 introduces the conclusion and future work. Finally section 9 and 10 are the acknowledgements and references.

2. RELATED WORK

According [4], copy prevention and detection methods can be combined to reduce plagiarism. While copy detection methods can only minimize it, prevention methods can fully eliminate it and decrease it. Notwithstanding this fact, prevention methods need the whole society to take part, thus its solution is non trivia [5]. Copy plagiarism detection methods, on the other hand, are easier to implement, and tackle different levels, from simple manual comparison to complex automatic algorithms [6,7]. A short discussion on plagiarism detection methods is presented.

Some methods have been developed in order to find original plagiarized text pairs on the basis of flexible search strategies (able to detect plagiarized fragments even if they are modified from their source). If two (original and suspicious) text fragments are close enough, it can be assumed that they are a potential plagiarism case that needs to be investigated deeper [8]. A simple option is to carry out a comparison of text chunks based on word-level n-grams. In Ferret [9], the reference and suspicious texts are split into trigrams, composing two sets that are after compared. The amount of common trigrams is considered in order to detect potential plagiarism cases. Another option is to split the documents into sentences. PPChecker [10] detects potentially plagiarized sentences on the basis of the intersection and complement of the reference and suspicious sentences vocabulary. Considering complement avoids detecting casual common text substrings as plagiarism cases.

Our algorithm depends on data mining, it is Apriori based so it detects all plagiarized text inside the source text files in an reasonable and acceptable time.

3. DATA MINING OVERVIEW.

Data mining [11, 12] is the process of extracting interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from large information repositories such as: relational database, data warehouses, XML repository, etc. Also data mining is known as one of the core processes of Knowledge Discovery in Database (KDD).

Sequential Pattern Mining.

Definition 1: Sequential pattern mining [12] is trying to find the relationships between occurrences of sequential events, to find if there exists any specific order of the occurrences.

In data mining [13] frequent itemsets are used to illustrate relationships within large amounts of data. The classical example is the analysis of the buying-behavior of customers. The database consists of a set of transactions, and each transaction is a set of items from a universal itemset *I*.

The goal is to find itemsets I that are subsets of many transactions T in the database D, ($I \in T$). An itemset is called frequent, if it occurs in a percentage that exceeds a certain given support count σ [14]:

$$\sigma(I) = \frac{|\{T \in D\}|\{I \subseteq T\}|}{|D|} \geq \sigma$$

In EgyCD, no interest in the percentage of itemsets is needed. Instead the algorithm is interested in their count

$$\sigma(I) = |\{T \in D\}|\{I \subseteq T\}| \geq \sigma \text{ where } \sigma > 1$$

Most SPM algorithms are based on Apriori algorithm [13]. AprioriAll. Sequential pattern mining was first introduced in [15] by Agrawal, three Apriori based algorithms were proposed. Given the transaction database with three attributes customer-id, transaction-time and purchased-items, the mining process were decomposed into five phases:

Sort Phase: the original transaction database is sorted with customer-id as the major key and transaction time as the minor key, the result is set of customer sequences.

L-itemsets Phase: the sorted database is scanned to obtain large 1-itemsets according to the predefined support threshold..

Transformation Phase: the customer sequences are replaced by those large itemsets they contain, all the large itemsets are mapped into a series of integers to make the mining more efficient.

Sequence Phase: all frequent sequential patterns are generated from the trans-formed sequential database.

Maximal Phase: those sequential patterns that are contained in other super sequential patterns are pruned in this phase, since only interesting in maximum sequential patterns.

Since most of the phases are straightforward, researches focused on the sequence phase in [16].

4. GENERAL DESCRIPTION OF EGYCD.

Following Apriori-based approaches, our approach builds up larger itemsets (words/statements in this case) from combining smaller ones and then efficiently searches inside the text files to verify their presence.

EgyCD tool consists of four steps:

a. The user selects the source files either it is in the directory or in different directories to apply the tool on.
b. The tool transforms the source files to transactions of itemsets.
c. EgyCD algorithm is applied to discover frequent itemsets in the text files that exceed a given frequency threshold.
d. The algorithm prunes all plagiarized text that appear completely in other plagiarized text to avoid duplicate results and report only original plagiarized not included in others.

Now a brief description for how EgyCD algorithm works is introduced. Assume that T is the set of all statements, where each statement is considered a transaction. First, the algorithm starts by getting the first itemset F which is the set of all repeated statements in the text files. Then it initializes a counter i to 1. It also initializes a set CC equal to F where CC is a set will always contain all plagiarized text discovered so far. Set CC_i is a sub set of CC always contains all plagiarized text of length i while i increases for an iteration to the next. The second step is to do Cartesian product CC_i x F and store the results in CC_{i+1}. The third step is checking each item in the Cartesian product of length i + 1 against Apriori property which states that any subset of any frequent itemset should be frequent, to reduce the time of this check, only two subsets for any item in CC_{i+1} are checked, the first subset is equal to the same item in CC_{i+1} but after removing its first element, and the second subset is equal to the same item in CC_{i+1} but after removing its last element. If any of those two subsets is not

frequent the item will be removed from CC_{i+1}. The fourth step is checking each item in the Cartesian product of length i + 1 to see if it exists in the set of all transactions T (i.e., the set of all statements in sequence) or not. If an item in the Cartesian product (after checking apriori property for all of its elements) set exists as subsequence of transactions in T, then it is added to the plagiarized text set CC. Since the result of the Cartesian product can be massive, it is possible to generate the results on the fly in the memory without storing them and process them directly in the third step by checking their presence in the transactions. The fifth step is prune all plagiarized text in CC of length i that exist in plagiarized text of length i + 1. The fifth step is incrementing i by 1. The sixth step is trying to reduce the set F by pruning all items that didn't appear as a last item in any of plagiarized text of length i. Finally the algorithm iterates over steps two to six until all items of the Cartesian product don't exist in any transactions. Below is the pseudo code of the algorithm.

```
1. T = set of all source lines
2. F = set of repeated source lines
3. CC = F
4. stillMore = true
5. i = 1
6. While (stillMore)
7. {
8.       stillMore = false
9.       CC_{i+1} = CC_i x F
10.      If i > 1 then
11.         CC_{i+1} = Check_Apriori(CC_{i+1},CC_i)
12.      End if
13.      For all e ∈ CC_{i+1}
14.      {
15.         if e ∈ T then
16.               add e to CC
17.               stillMore = true
18.         end if
19.      }
20.      prune CC by removing all e ∈ CC
         where |e| = i and e ⊂ S and S ∈
         CC where |S| = i+1
21.      i = i + 1
22.      prune all non used elements in F
23. }
```

Pseudo-code of EgyCD Algorithm

```
1.  Check_Apriori(CC_{i+1},CC_i)
2.  {
3.       For all e ∈ CC_{i+1}
4.       {
5.           a = all elements in e except
             first element
6.           if a ∉ CC_i then
7.               prune e from CC_{i+1}
8.           else
9.               b = all elements in e
                 except last element
10.              if b ∉ CC_i then
11.                  prune e from CC_{i+1}
12.              end if
13.          end if
14.      }
15.      Return CC_{i+1}
16. }
```

Pseudo-code of `Check_Apriori(CC`$_{i+1}$`,CC`$_i$`)`

To explain how the algorithm work on an example of detecting plagiarized text

Example

Suppose the following text:

My name is Ali
I live in Egypt
.....
.....
My name is Ali
I live in Egypt
.....
.....

The final result should be CC= {(My name is Ali, I live in Egypt)}

Tracing of the algorithm

F = { My name is Ali, I live in Egypt }
CC = F
i = 1
stillMore = true

iteration 1:

{
stillMore = false
CC_{i+1} = {My name is Ali, I live in Egypt} x {My name is Ali, I live in Egypt}
CC_{i+1} = {(My name is Ali, My name is Ali)
, (My name is Ali, I live in Egypt)
, (I live in Egypt , My name is Ali)
, (I live in Egypt; , I live in Egypt)}
CC_{i+1} = { My name is Ali, I live in Egypt
, (My name is Ali, I live in Egypt)}
CC_{i+1} = { (My name is Ali, I live in Egypt) }
CC = { My name is Ali, I live in Egypt ,(My name is Ali, I live in Egypt)
//after pruning CC will be
CC = { (My name is Ali, I live in Egypt) }
stillMore = true
i = 2
F = { I live in Egypt } // after pruning

}

Iteration2:

{
stillMore = false
CC_{i+1} = { (My name is Ali, I live in Egypt) } x { I live in Egypt }
CC_{i+1} = { (My name is Ali, My name is Ali , I live in Egypt) }
CC_{i+1} = Φ //After Apriori property check
stillMore = false
CC = { (My name is Ali, I live in Egypt) }
i = 3
F = Φ
}

No more loops since stillMore = false and
CC = { (My name is Ali, I live in Egypt) }

5. OPTIMIZATION TRICKS ADDED TO APRIORI

The research did some modifications to Apriori for increasing the speed of EgyCD such as:-

a. Pruning F set at the end of each iteration to decrease the cardinality of the first itemset and consequently the cardinality of the resultant set of the Cartesian product.
b. The Apriori property states that any subset of a frequent set is frequent [12]. For stores system sorting items in transactions is meaningless but in plagiarized text sorting statements is a major concept, so a check to Apriori property only for two subsets that are the union of a plagiarized text but after removing the first statement or the last statement of that plagiarized text and the new added statement.
c. By using the SQL features in where conditions, all items of CC_{i+1} that exist in sequence in the text files has been got then the algorithm checks if it is a plagiarized text or not.
d. Applying EgyCD inside the database not in the application, this speed its execution time.

6. IMPLEMENTATION DETAILS

The algorithm was implemented in a database application. using Adaptive Server SQL Anywhere version 11.0 with add on In-Memory version 11.0 and PowerBuilder 11.5. This has multiple advantages. First, it perfectly matches the application of Apriori-based algorithms which are developed for mining databases. Second, the expressive power of SQL supports processing of transactions very easily and smoothly. Finally, PowerBuilder has powerful visualization capabilities that allow us to visualize plagiarized text in very simple ways and can also be upgraded with new views if needed. For every language to be supported, language specific tables are filled with the style of comments, reserved words and symbols, begin and end markers of compound statements, statements separator, etc.

The proposed algorithm can be used to detect plagiarism in written text.

There are two modes for EgyCD, prune and no prune, if the user want to see all plagiarized text and its subsets plagiarized text in the source text files he will use no prune mode and if the user wants to see only all plagiarized text and the program should remove delete all of plagiarized text subsets, hence the user should select prune mode.

In detecting plagiarism a minor transformation is done to the imported text such as replacing multiple consequent spaces with only one space and all tab or multiple consequent tabs with one space.

6.1 VISUALIZING THE PLAGIARIZED TEXT AND ITS QUALITY

To make good visualization to the plagiarized clones, EgyCD lets the user defines the quality of the text clones in the application setting screen. Four fields control that, two fields for defining the excellent quality for text clones, the length of the text clone field and the count of text clone field. The same two fields are used for defining good quality for text clone. If the resultant text clone length is greater than or equal the length field value for excellent quality and its repetition is greater than or equal to the count text clone field value for excellent quality then the background of this text clone will be in red. And if the resultant text clone length is greater than or equal the length field value for good quality and its repetition is greater than or equal to the count text clone field value for good quality then the background of this texct clone will be in orange otherwise the text clone background color is green.

By using this way, the user can easily notice and differentiate the most important text clones.

6.2 CALCULATING TEXT CLONE FILE RATIO.

To submit some information that may be useful to EgyCD users, we calculate a ratio called text clone file ratio (TCFR) for each file selected by the user for detecting text clones inside it. It is equal to the full size in lines of all text clones inside the file over the total size of the file in lines

TCFR = Size of text clones in the file in lines / size of the file in lines

The user can see this ratio if he displayed again his selected files. The user will find that this ratio is calculated and displayed in the row of each file. If the ratio is greater than a specific percent set by the user in the EgyCD setting then the background color will be red for this row otherwise the background will be in white.

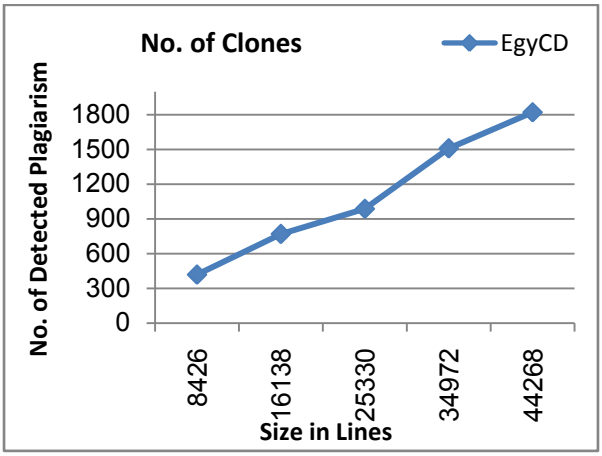
7. CASE STUDIES.

The research submits a case study that shows No. of plagiarism detected and its corresponding time. EgyCD is applied over 720 files and its size is 1.31 MB. These files are divided into 5 groups; the first group contains 144 files and each consequent group contains the files of the pervious group and has 144 additional files, so the last group contains 720 files. The total size of these files is 1.31MB and they collectively contain about 45454 lines. The hardware used in this case is Intel® Core™ 2 Duo CPU E7200 Processor, 2.53 GHz, 2GB RAM, running windows XP.

Table (1) – Case Information

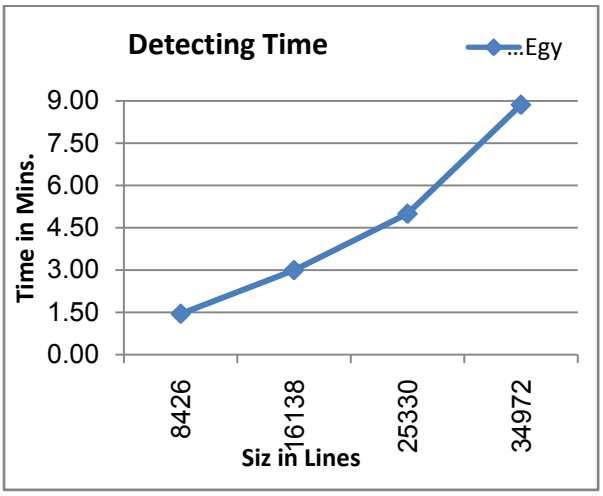
Seq.	Size in Lines	Clone Size	No. of Clones	Time in Min.
1	8426	1637	419	0.68
2	16138	2966	770	1.45
3	25330	4259	988	3.00
4	34972	5931	1510	5.00
5	44268	7564	1822	8.87

Table (1) and graph (1) show that No. of clones increases with the increase of size of files in lines.



Graph (1) - No. of Detected Clones

Table (1) and graph (2) show that the time of detecting clones increases with the increase of size of files in lines, and this is logic and reasonable since more text clones needs more detected time to detect. Also detecting time is short in minutes not hours and hence EgyCD execution time is acceptable.



Graph (2) – Time of Detected Clones

8. CONCLUSIONS & FUTURE WORK

The research presented a new plagiarism detection algorithm that utilizes sequential pattern mining to discover copy/paste. EgyCD detects all copy/paste in the source text files with 100% precision and recall, this is due to the nature of our Apriori-based algorithm. Precision and high recall was shown by experimental study to be excellent. Good visualization and some new information have been submitted such as text clone quality and text clone file ratio.

Future work will include the utilization of multi-threaded database programming and distributed systems to speed up EgyCD. It will also include the deployment of further data mining and non Apriori-based SPM algorithms to further investigate the value of this family of algorithms in plagiarism detection EgyCD.

9. ACKNOWLEDGMENTS

Thanks to. Chanchal K. Roy, for his support, technical comments and research as well as his encouragement for this work, also thanks for Auni Ku and Ira for his support.

10. REFERENCES

[1] D. A. Black, Tracing Web Plagiarism – A guide for teachers, Internal Document, Department of Communication, Seton Hall University, Version 0.3, Fall 1999.

[2] P. Clough ,Plagiarism in natural and programming languages: an overview of current tools and technologies, July 2000, Department of Computer Science, University of Sheffield

[3] L. R. Jones, Academic Integrity & Academic Dishonesty:A Handbook About Cheating & Plagiarism, Revised & Expanded Edition, Florida Institute of Technology, Melbourne, Florida.

[4] Schleimer, S., Wilkerson, D.S., Aiken, A.: Winnowing: local algorithms for document fingerprinting. In: SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data. pp. 76–85. ACM, New York, NY, USA (2003).

[5] Approaches for Intrinsic and External Plagiarism Detection Notebook for PAN at CLEF 2011, Gabriel Oberreuter, Gaston L'Huillier, Sebastián A. Ríos, and Juan D. Velásquez, Department of Industrial Engineering, University of Chile.

[6] Potthast, M., Barrón-Cedeño, A., Eiselt, A., Stein, B., Rosso, P.: Overview of the 2nd international competition on plagiarism detection. In: Braschler, M., Harman, D. (eds.) Notebook Papers of CLEF 2010 LABs and Workshops, 22-23 September, Padua, Italy (2010).

[7] Potthast, M., Stein, B., Eiselt, A., Barrón-Cedeño, A., Rosso, P.: Overview of the 1st international competition on plagiarism detection. In: Stein, B., Rosso, P., Stamatatos, E., Koppel, M., Agirre, E. (eds.) SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09). pp. 1–9. CEUR-WS.org (Sep 2009), http://ceur-ws.org/Vol-502.

[8] A. B. Cede˜no, P. Rosso ,On Automatic Plagiarism Detection Based on n-Grams Comparison, Natural Language Engineering Lab., Dpto. Sistemas Inform´aticos y Computaci´on, Universidad Polit´ecnica de Valencia, Spain.

[9] Lyon, C., Barrett, R., Malcolm, J.: A Theoretical Basis to the Automated Detection of Copying Between Texts, and its Practical Implementation in the Ferret Plagiarism and Collusion Detector. In: Plagiarism: Prevention, Practice and Policies Conference, Newcastle, UK (2004).

[10] Kang, N., Gelbukh, A.: PPChecker: Plagiarism Pattern Checker in Document Copy Detection. In: Sojka, P., Kope˘cek, I., Pala, K. (eds.) TSD 2006. LNCS, vol. 4188, pp. 661–667. Springer, Heidelberg (2006).

[11] M.-S. Chen, J. Han, and P. S. Yu. Data mining: an overview from a database perspective. IEEE Trans. On Knowledge And Data Engineering 8, 866-883 (1996).

[12] Q. Zhao, S.S. Bhowmick, Sequential pattern mining: a survey, Technical Report Center for Advanced Information Systems, School of Computer Engineering, Nanyang Technological University, Singapore, (2003).

[13] C. Liu, C. Chen, J. Han and P. Yu, GPLAG: Detection of Software Plagiarism by Program Dependence Graph Analysis, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2006, pp. 872-881 (2006).

[14] Vera Wahler, Dietmar Seipel, J¨urgen Wolff v. Gudenberg, and Gregor Fischer. Clone Detection in Source Code by Frequent Itemset Techniques, Source Code Analysis and Manipulation, 2004. Fourth IEEE International Workshop on16-16 Sept. 2004.

[15] M. Gabel, L. Jiang and Z. Su, Scalable Detection of Semantic Clones, in: Proceedings of the 30th International Conference on Software Engineering, ICSE 2008, pp. 321-330 (2008).

[16] A. Leitlao, Detection of Redundant Code Using R2D2, Software Quality Journal, 12(4):361-382 (2004).