2013 12th International Conference on Document Analysis and Recognition

Achieving Linguistic Provenance via Plagiarism Detection

Nwokedi Idika, Harry Phan, Mayank Varia
MIT Lincoln Laboratory
Lexington, MA 02421
{*nwokedi.idika, harry.phan, mayank.varia*}*@ll.mit.edu*

***Abstract*—To go beyond what current provenance systems can capture for natural language text documents, we propose the Lincoln Laboratory Plagiarism for Provenance System (LLPlā) as an approach for capturing linguistic provenance. Linguistic provenance infers the origin of text based on its linguistic structure. We take a plagiarism detection approach to this task as identifying similar sections of text is fundamental to linguistic provenance and central to LLPlā's performance. Thus, to determine the most viable plagiarism detection algorithm for use in LLPlā, we evaluate three state-of-the-art plagiarism detection algorithms. Moreover, we propose extensions to the best-performing algorithm that improve its precision with negligible effects on recall.**

***Keywords*-provenance; plagiarism detection; graphs;**

I. Introduction

Current system-based provenance systems are concerned with the provenance of data and processes [1]–[3]. To capture this type of provenance, one must log the actions of the processes under inspection. Under this scheme, understanding the provenance of data equates to understanding the inputs and actions taken on inputs to the data. However, there are cases where such a scheme is insufficient for understanding the provenance of data. For example, when an analyst is interested in understanding the origins of statements in a document, the analyst obtains limited help from the scheme current provenance systems employ. The most an analyst can determine from current approaches is what network the document came from along with any other annotations included with the document.

Thus, in this paper, we present Lincoln Laboratory Plagiarism for Provenance System (LLPlā), an approach for capturing linguistic provenance. Central to LLPlā, is its ability to identify similar sections of text. To demonstrate viable candidates for achieving this crucial feature, we evaluate three state-of-the-art plagiarism detection algorithms, and demonstrate an algorithm extension that enhances the precision of the best one while having little effect on recall.

II. LLPlā: Achieving Linguistic Provenance

Linguistic provenance refers to the pedigree of a word, phrase, or statement usage. Due to missing or incomplete information, and the potential scope of candidate documents, achieving exact linguistic provenance for all words, phrases and statements is currently impractical. However, if the scope of candidate documents is constrained and timing information for documents are known, linguistic provenance becomes possible.

Under the aforementioned constraints, linguistic provenance can be achieved through the use of plagiarism detection algorithms. Using these algorithms, LLPlā can identify sufficiently similar text across documents and assert where information is derived from. More formally, let d_k and d_l correspond to documents with words, phrases, or statements where k and l correspond to the creation times for their respective documents such that $k < l$. Then a plagiarism detector (PD) establishes the linguistic provenance of a section of text $q(d_l)$ as being derived from $q(d_k)$ if PD's similarity function $z(d_k, d_l) > t$, where t is the required similarity threshold for deeming d_l to be directly derived from d_k.

Part of the novelty of LLPlā resides in how it uses plagiarism detection. The difference between how plagiarism detection is traditionally used and how we are using plagiarism detection is depicted in Figure 1.

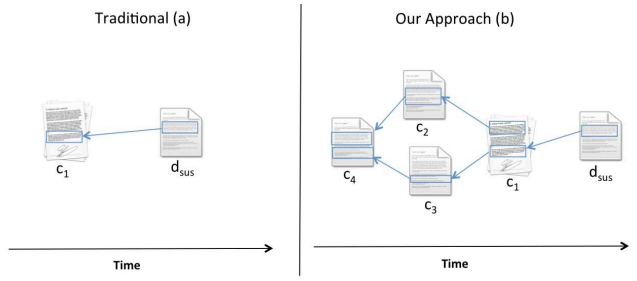

Figure 1. (a) Traditional plagiarism detection approach (b) Our approach; recursively applying plagiarism detection

In traditional plagiarism detection, there exists a corpus C containing a collection of documents. There also exists a suspicious document d_{sus} that we are interested in determining whether sufficient portions of its contents correspond to previously created sections of text in $c_i \in C$. This approach is depicted on the left of Figure 1. In our approach, depicted on the right side of Figure 1, we apply the traditional concept of plagiarism detection recursively. That is, the corpus C is comprised of documents that were d_{sus} in the past. Hence, when d_{sus} has been compared to all relevant documents in the corpus, it joins the corpus.

1520-5363/13 $26.00 © 2013 IEEE
DOI 10.1109/ICDAR.2013.133

A. Vision: Operational System Features

What follows are a series of features that an operational LLPlā should contain. We did not build LLPlā for enterprise deployment; however, if we did, the following features would be critical:

- Assert relationships between/among sufficiently similar texts in documents to end users
- Allow users to create local mutable versions of provenance graphs
- Allow users to deploy custom analytics on the provenance graphs
- Allow users to subscribe to changes in provenance for documents or sections of text

The value of linguistic provenance is in the relationships it reveals. Thus, identifying and asserting discovered relationships to the user is critical to LLPlā. By allowing users to create local mutable versions of the linguistic provenance graph, end users can inject their own assumptions and run analytics on the graph based on those assumptions. This feature gives analysts the ability to perform what-if analyses. This feature also enables provenance graph modification so that classified or proprietary information is not shared if the analyst wishes to share a subset of the provenance graph with another analyst or organization (similar to MITRE PLUS [3]). Providing the ability to deploy custom analytics allows for useful analytics to be added to the system as they are discovered or needed. Allowing analysts to subscribe to relevant documents or sections of text gives them the ability to be notified of changes in provenance that may affect their perception on situations of interest. For instance, if the provenance of information utilized in an intelligence report is determined to be worse than initially anticipated, then analysts will want to know so they can modify any analysis depending that information.

B. Evaluation Architecture

The evaluation architecture depicted in Figure 2 contains four components: the Comparator, the Document Corpus, the Probabilistic Provenance Graph Generator, and the Probabilistic Provenance Graph Store. We discuss each component below. The evaluation architecture was created with some operational features for demonstration purposes.

1) Comparator: The Comparator contains one or more plagiarism detection algorithms, and when a document d comes in, d is compared against the documents in the Document Corpus. As a plagiarism detection algorithm produces plagiarism detections, the detections are sent to the Probabilistic Provenance Graph (PPG) Generator. When the plagiarism detection algorithms are done with d, the Comparator stores the document in the Document Corpus. This process makes the new document d a part of the corpus.

2) Document Corpus: The Document Corpus is a store of all documents of interest. Documents may be derived from any source. As long as the documents are normalized to a format amenable for the plagiarism detection algorithms used. In our evaluations, all documents are in plaintext format.

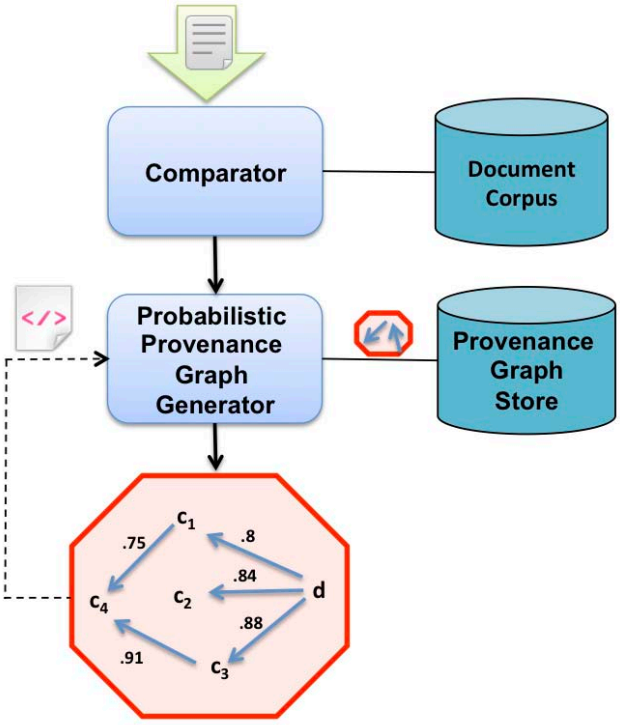

Figure 2. Evaluation architecture

3) Probabilistic Provenance Graph Generator: The Probabilistic Provenance Graph (PPG) Generator receives the detections sent from the Comparator. Each detection corresponds to an edge in a graph. The associated vertices correspond to sections of text from two documents. The edge weight corresponds to the strength of detection. Hence, values closer to 100% suggest that the plagiarism detection algorithms believe more strongly that the two documents share common text/concepts. This belief is interpreted as a probability. As can be seen in Figure 2 the PPG generator can also produce a PPG as its output. It may also accept modifications from a user who is interested in seeing or storing the PPG under alternative assumptions. Figure 3 shows our prototype. Although, in this work, the probabilities in the PPG representation are derived from the similarity scores of the plagiarism detection algorithms, the PPG model is far more general. Other methods (e.g., supervised learning techniques) can be used to produce the probabilities of interest.

4) Probabilistic Provenance Graph Store: The PPG Store persists the PPGs. Because users can modify a PPG, thereby creating a new PPG local to that user, users can have

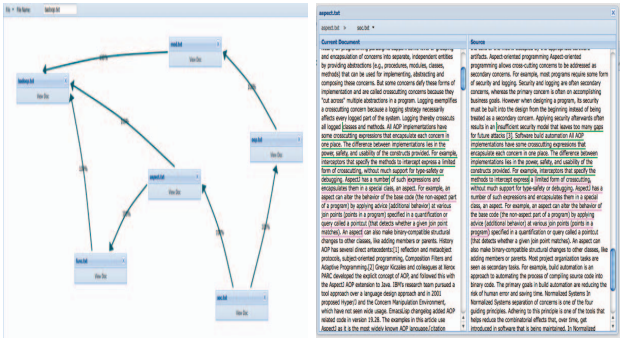

Figure 3. Example prototype-based probabilistic provenance graph (left), Two documents on the same linguistic provenance path (right)

different views of the PPG. This mechanism allows for a local view of the PPG of interest that does not affect the global view.

C. Similar Text Detection with SVOP-Trim

SVOP-Trim is a modification of the SVOP algorithm [4]. Extrinsic plagiarism detection algorithms (those utilizing a corpus), generally have three phases: preprocessing, detailed comparison, and postprocessing. Our modification occurs in the postprocessing phase of SVOP. In the postprocessing phase, candidate plagiarism detections have been identified, and the goal in this phase is to consolidate detections that are sufficiently close to each other and to prune false positive detections. Thus, our reason for extending SVOP was to improve its performance (see Section III-B). Our modification occurs after candidate plagiarism detections have been consolidated and is specified in the algorithm below.

---

function FINALTEST($M, L, t \leftarrow 15, t_s \leftarrow 0.2, t_m \leftarrow 10$)
  keep $\leftarrow \emptyset$
  for $(s_1, s_2) \in M$ **do**
    max $\leftarrow$ maxLength((s_1, s_2))
    min $\leftarrow$ minLength $((s_1, s_2))$
    if $(N_{\text{mw}}(s_1 - L, s_2 - L) \geq t \times t_s)$ and $(\min \times t_m >$ max) **then**
      keep.append((s_1, s_2))
    end if
  end for
  return keep
end function

The set M corresponds to the set of consolidated plagiarism detections. The set L corresponds to a set of stop words. The input t corresponds to the minimum plagiarism detection length. The variable t_s is used to scale down t. The variable t_m is used to filter out detections that map large sections of text to much smaller sections of text. The pair (s_1, s_2) corresponds to a plagiarism detection where s_1 and s_2 correspond to the two sufficiently similar sections of text from two separate documents. The function "maxLength" returns the maximum section text length (i.e., number of characters) from s_1 and s_2. The function "minLength" returns the minimum section text length from s_1 and s_2. The function N_{mw} returns the number of matched words between two documents or sections of text. The set "keep" stores plagiarism detections meeting the criteria specified in the if statement condition.

III. Evaluation of Plagiarism Detection Algorithms

We implemented each plagiarism detection algorithm based on the descriptions provided in their respective papers. We found some of the descriptions to be partially ambiguous and in some cases incomplete; in these situations, we made a best effort to use the best performing parameters and logic for each algorithm.

A. Experiment Setup

The top three plagiarism detection algorithms from the PAN 2011 competition were: the encoplot algorithm [10], the SVOP algorithm [4], and the FastDoCode algorithm [11]. To evaluate the top three algorithms, we utilized the PAN 2010 Competition Corpus [5]. This corpus contains 31850 suspicious files and 11148 source files. There are two types of plagiarism cases in the PAN 2010 corpus: artificial and simulated. Artificial plagiarisms are plagiarism cases generated by a machine. For artificial plagiarism cases, there are three labeled levels of obfuscation: high, low, and none. The level of a plagiarism case's obfuscation decreases from high to none. It is also possible for artificial plagiarism cases not to have an obfuscation label as well. Simulated plagiarisms are plagiarism cases generated by humans.

Of the various types of plagiarism cases, we are most interested in those that are generated by humans. Since simulated plagiarisms are created by humans, we would expect that of all of the plagiarism types in the PAN 2010 corpus, simulated plagiarism cases would be most representative of the types of similar texts we might see in the wild. As a lower bound on detection capability, a plagiarism detection algorithm should be able to identify direct copies of text with high precision and recall. Thus, in our evaluations, we also include the evaluation of plagiarism detection algorithms on direct-copy plagiarism cases, which are classified as having obfuscation level none. To get a gist of how the algorithms perform on the entire corpus, we also evaluate how the algorithms perform on each type of plagiarism case.

The experiments were ran on the LLGrid system [6]. To achieve parallelism, we utilized LLGrid's LLGrid_MapReduce facility [7]. To evaluate the algorithms,

we used the macro-averaged precision and recall to obtain an F_1 score for each algorithm as defined in [8].

B. Evaluating Algorithms on Varied Types of Plagiarisms

In this section, we go through the results we have obtained for the implemented plagiarism detection algorithms for samples of the PAN 2010 Competition Corpus. After demonstrating the performance of the algorithms on direct-copy plagiarism cases, we demonstrate their performance on a sample of the various types of plagiarisms in the corpus, and then we show how the algorithms perform on simulated plagiarism cases.

1) Evaluating on Plagiarism Cases with Obfuscation Level None: To evaluate the algorithms' performance on direct copy plagiarism cases, we sampled the PAN Corpus artificial plagiarism cases having an obfuscation level of "none." The sample was composed of 200 randomly selected suspicious files and their corresponding 727 source files.

Algorithm	Precision	Recall	F_1 Score
Encoplot	78%	83%	81%
FastDoCode	61%	63%	62%
SVOP	79%	96%	87%
SVOP-Trim	91%	96%	93%

Table I
RESULTS ON NONOBFUSCATED PLAGIARISM CASES

On nonobfuscated plagiarism cases, the two SVOP algorithms outperform the others on precision and recall. Based on the PAN 2011 Competition results [9], this should not be surprising.

As can be seen, the precision of SVOP-Trim is 12% better than SVOP and has the same recall as SVOP. This accounts for the improved F_1 score as the recall is the same for the two algorithms. This result suggests that SVOP-Trim is the best choice among considered algorithms if direct-copy plagiarism cases are of primary interest.

2) Evaluating on Plagiarism Cases with Each Type of Plagiarism Case: To evaluate the algorithms' performance on each type of plagiarism case in the PAN Corpus, we randomly selected suspicious files (and their associated source files) of each type. The types were artificial plagiarism cases for all obfuscation levels: artificial plagiarism cases for high obfuscation levels, artificial plagiarism cases for low obfuscation levels, and artificial plagiarism cases for obfuscation level none. The fourth type included was the simulated plagiarisms. This yielded a total of 199 suspicious files (one suspicious file appeared in two groups). There were 602 source files associated with these suspicious files.

The results above mimic the results in the PAN 2011 Competition (excluding, of course, SVOP-Trim). If the F_1 score is computed for the results obtained for these three algorithms from the PAN 2011 Competition, SVOP would have a score of 56%, Encoplot would have a score of 48%, and FastDoCode would have a score of 37%. We have obtained the same ranking for the algorithms.

Algorithm	Precision	Recall	F_1 Score
Encoplot	76%	58%	66%
FastDoCode	77%	52%	62%
SVOP	79%	87%	83%
SVOP-Trim	90%	85%	87%

Table II
RESULTS ON A MIXED SET OF PLAGIARISM CASES

SVOP-Trim exhibits a higher F_1 score than SVOP due to its 11% improvement in precision. The 2% reduction in recall was not significant enough to offset the overall improvement. Table II suggests that if given a mixed set of plagiarism cases, SVOP-Trim would be desirable over SVOP.

3) Evaluating on Simulated Plagiarism Cases: To evaluate the algorithms' performance on simulated plagiarism cases, we extracted all simulated plagiarism cases from the PAN Corpus, and applied the algorithms to the dataset containing these cases. There were 847 suspicious files that were classified as simulated plagiarism cases. There were 1316 source files associated with these suspicious files.

Algorithm	Precision	Recall	F_1 Score
Encoplot	48%	28%	35%
FastDoCode	82%	32%	46%
SVOP	87%	59%	70%
SVOP-Trim	95%	56%	71%

Table III
RESULTS ON SIMULATED PLAGIARISM CASES

Similar to the previous evaluations, the SVOP algorithms outperform the other algorithms on precision and recall. SVOP's performance on this type of plagiarism case, and its performance on other forms of plagiarism, is the reason why we chose to base our improvements on this algorithm.

SVOP-Trim has a slightly better F_1 score than SVOP. Hence, it's not clear that SVOP-Trim would always better on plagiarism cases created by humans. The choice of algorithm here may be dictated more by the users' preferences. If users are more interested in precision, then SVOP-Trim is the clear choice. However, if getting every viable plagiarism case is important, and if users do not mind the generated false positives, then SVOP would be the better choice.

IV. RELATED WORK

To focus our evaluation, we limited our evaluation of plagiarism detection algorithms to those that appeared in the Plagiarism analysis, Authorship identification, and Near-duplicate document detection (PAN) competition [5]. We chose the top three performing plagiarism detection algorithms based on the metrics provided in [9]. Because we are ultimately interested in the plagiarisms that resemble the

type of plagiarisms a human would commit, in the PAN 2011 Corpus [9], manual plagiarism cases are of highest interest. There are two things to note about these results. One, the top three scores are close enough, that if a sufficiently different corpus had been presented to these algorithms the outcome could have been different. The second relevant observation is that the fourth-best performing algorithm is 30% worse than the third-best performing algorithm, whereas the third-best performing algorithm is 3% worse than the best performing algorithm.

The major differences between SVOP and SVOP-Trim is the usage of stop words and the calibration and introduction of thresholds. SVOP-Trim uses stop words in the post-processing phase whereas SVOP does not. SVOP-Trim also shortens the minimum detection length after removal of these stop words. This action calibrates the minimum detection length to accommodate the potentially shorter plagiarism detections. SVOP-Trim also introduces a new threshold to remove detections that have significant text length differences.

V. Conclusion

In this work, we have described LLPlā, and we have demonstrated the performance of three plagiarism detection algorithms. We have provided some rationale for why the algorithms may perform the way they do. We have introduced enhancements to one algorithm, SVOP, that outperforms the original version. We were able to achieve this result by modifying the post-processing phase of the algorithm to significantly improve its precision. We have focused solely on the use of plagiarism detection algorithms for LLPlā; however, semantics are critical as well. Future work will involve incorporating semantics into linguistic provenance.

Acknowledgment

This work is sponsored by the Assistant Secretary of Defense for Research & Engineering under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

The authors would like to thank the following people and organization for their assistance in this work: Suresh Damodaran (MIT LL), Yaron Rachlin (MIT LL), Jonathan Kurz (formerly of MIT LL), Dan Van Hook (MIT LL), Matthew Daggett (MIT LL), Douglas Marquis (MIT LL), Jeffrey Gottschalk (MIT LL), Joshua Haines (MIT LL), Arkady Yerukhimovich (MIT LL), Nabil Schear (MIT LL), Joseph Cooley (MIT LL), and In-Q-Tel.

References

[1] P. Groth, S. Miles, and L. Moreau, "Preserv: Provenance recording for services," in *in Proc. AHM05*, 2005.

[2] Y. L. Simmhan, B. Plale, and D. Gannon, "Karma2: Provenance management for data driven workflows," *International Journal of Web Services Research, Idea Group Publishing*, vol. 5, 2008.

[3] A. Chapman, B. T. Blaustein, L. Seligman, and M. D. Allen, "Plus: A provenance manager for integrated information," in *IRI*. IEEE Systems, Man, and Cybernetics Society, 2011, pp. 269–275.

[4] J. Grman and R. Ravas, "Improved implementation for finding text similarities in large sets of data - notebook for pan at clef 2011," in *CLEF (Notebook Papers/Labs/Workshop)*, 2011.

[5] PAN, http://pan.webis.de, 2012, [Online; Accessed 24-September-2012].

[6] A. Reuther, J. Kepner, T. Currie, K. Hahn, A. McCabe, M. Moore, and N. Travinin, "Llgrid: Enabling on-demand grid computing with gridmatlab and pmatlab," in *HPEC Workshop*, 2004.

[7] C. Byun, W. Arcand, D. Bestor, B. Bergeron, M. Hubbell, J. Kepner, A. McCabe, P. Michaleas, J. Mullen, D. O'Gwynn, A. Prout, A. Reuther, A. Rosa, and C. Yee, "Driving big data with big compute," in *2012 IEEE High Performance Extreme Computing Conference (HPEC '12)*, 2012.

[8] M. Potthast, B. Stein, A. Barrón-Cedeño, and P. Rosso, "An evaluation framework for plagiarism detection," in *COLING (Posters)*, 2010, pp. 997–1005.

[9] M. Potthast, A. Eiselt, A. Barrón-Cedeño, B. Stein, and P. Rosso, "Overview of the 3rd international competition on plagiarism detection," in *CLEF (Notebook Papers/Labs/Workshop)*, 2011.

[10] C. Grozea and M. Popescu, "The encoplot similarity measure for automatic detection of plagiarism - notebook for pan at clef 2011," in *CLEF (Notebook Papers/Labs/Workshop)*, 2011.

[11] G. Oberreuter, G. L'Huillier, S. A. Ríos, and J. D. Velásquez, "Fastdocode: Finding approximated segments of n-grams for document copy detection - lab report for pan at clef 2010," in *CLEF (Notebook Papers/LABs/Workshops)*, 2010.