Using a Variety of n-Grams for the Detection of Different Kinds of Plagiarism

Notebook for PAN at CLEF 2013

Prasha Shrestha and Thamar Solorio

University of Alabama at Birmingham
prasha@cis.uab.edu, solorio@cis.uab.edu

Abstract A text can be plagiarised in different ways. The text may be copied and pasted word by word, parts of the text may be changed, or the whole text may be summarised into one or two lines. Different kinds of plagiarism require different strategies to detect them. But rarely do we know beforehand what type of plagiarism we are dealing with. In this paper we present a system that can detect verbatim plagiarism and obfuscated plagiarism with similar accuracy. Our system uses three different types of n-grams: stopword n-grams, n-grams with at least one named entity, and all words n-grams. After detecting and merging the detections to obtain passages, the system finds new detections incrementally based on the idea that the passages on the vicinity of plagiarised passages are more likely to be plagiarised. The system performs well on verbatim as well as obfuscated plagiarism cases.

1 Introduction

Plagiarism detection is the task of finding out plagiarised portions within a piece of text. The process of extrinsic plagiarism detection can be clearly separated into two steps: source retrieval and text alignment. Source retrieval involves pointing out all the source documents, parts of which might have been reused in a given suspicious document. After source documents have been recovered by the source retrieval step, the next step is text alignment. The purpose of text alignment is to locate plagiarised content in the suspicious document along with the corresponding original text in the source document. If however, there are no detections, the suspicious document will be categorised as being non-plagiarised. The focus of the source retrieval task is on efficiently labeling a small number of documents as source from a huge pool of documents, whereas the focus of text alignment is on accurately finding out the plagiarised content. This paper deals with the latter.

In the passage retrieval task of PAN-CLEF 2013, a pair of suspicious and source documents is given. From this pair a participant's program has to identify all the plagiarised passages in the suspicious document and the corresponding original passages in the source document.

In order to compare two documents, they must first be divided into segments. A common technique used is sliding windows of n-grams. Both character and word n-grams have been used in previous approaches. The windows can either be overlapping or non-overlapping. The former requires more comparisons and hence gives better accuracy

in most cases, while the latter requires less comparisons and hence less computational time.
Our system uses n-grams having different characteristics to find out plagiarism ranging from verbatim plagiarism to plagiarism with different levels of obfuscation. Our system extracts the following types of n-grams:

Stopword n-grams
: n-grams of just the stopwords present in the document

Named entity n-grams
: n-grams containing at least one named entity

All word n-grams
: n-grams of all the words in the document

Comparison of stopword n-grams by exact matching is good for detecting no obfuscation to moderate obfuscation cases. But this method cannot catch plagiarised passages that have high word shuffling, unless the value of n is really small. However, using very low value of n for stopwords will greatly increase the number of false positives. Also, in the worst scenario, an obfuscating algorithm could shuffle words to a point that even the meaning of the text might be lost. Performing unordered matches useful in this case. Performing unordered matches useful in this case. But since stopwords are so common, allowing for unordered matches will again result in a lot of false positives.
Unlike stopwords named entities are not very common. Comparison of two named entity n-grams by finding out the common words between the two can catch even the high obfuscation cases while at the same time, keeping the number of false positives low. But because of the same reason that they are not very common, this comparison does not contribute much to the detections.
In order to increase the number of detections, a more lenient method is required. Comparison of all word n-grams in the same way as named entity n-grams is lenient enough to find very highly plagiarised passages. But this method produces a lot of false positives. A condition that there must be at least some number of detections adjacent to each other for those detections to be counted eliminates most of these false positives.
Consider the following example extracted from the randomly obfuscated case of PAN'13 training data:

Source text *Years later, a sword called "Souunga" reappears and recalls Takemaru from the grave, and Takemaru decides he wants to play dogcatcher*

Plagiarised Suspicious text *Years, sword called "Souunga" reappears and grave, Takemaru decides he wants to play dogcatcher…*

Here, the stopword n-gram method will not work even if n is lowered to 4. Only if 3-grams are considered, a 3-gram containing *and, he, to* will be detected. All word n-gram comparison will detect this plagiarised text for a range of values of n. One such example is when n is taken as 8 and the source and suspicious 8-grams are required having 5 common words for them be counted as plagiarised. In the example, the first 8-grams of the source and suspicious documents have six common words. The next pair of 8-grams has five common words. So, these n-grams will be counted as plagiarised. Since this example also contains named entities such as *Takemaru* and *Souunga*, named

entity n-gram method will also detect this.
After the detections have been merged into passages, some more plagiarised sections can be detected while keeping the number of false positives low by searching only in the adjacent parts of those already detected passages. After merging these new detections with the previous ones, the end result will be a good number of correct detections for a range of obfuscations.

2 Related Work

The stopword n-grams approach used in this paper was originally proposed by Stamatatos in [7]. The main idea behind that paper is that stopwords represent the syntax of the document. By using stopwords, it is possible to find plagiarism in texts in which there exists moderate obfuscation such as synonym replacement. According to Stamatatos, stopword n-grams can be used in plagiarism detection because stopwords represent the structure of the document, which cannot be changed easily if the meaning needs to be preserved.
The PAN'12 plagiarism detection summary paper summarizes all the approaches used to solve the task in that year [5]. Among the systems submitted during PAN'12, the task of document retrieval consisted of the following phases: seeding (finding matches between segments of the documents), match merging (obtaining paragraphs of maximum possible length from these seeds) and extraction filtering (filtering out overlapping passages and too short passages). Our system goes through these same phases in order to detect plagiarised sections of the text.
For comparison of documents, the participants of the competition in 2012 have mostly used sorted and/or unsorted n-grams of words and characters. Suchomel et al. have used the idea of comparison using multiple similarity measures that they call common features [8]. They have used lexicographically sorted word 5-grams and stopword 8-grams as their common features. In order to improve granularity, they have applied heuristics based upon these features. Kong et al. use sentences as the basic unit of comparison and they take cosine distance and the overlap between two sentences as the marker of similarity between those sentences [4].
Barrón-Cedeño and Rosso in [1] have tried to investigate the best value for n when performing an n-gram comparison. They used word n-grams in their method and found out that low values of n such as 2 to 3 generally work better.

3 Methodology

This section discusses the steps that we took in building our system. Since n-grams are the main components of our approach, we first extracted the necessary kinds of n-grams from both the source and suspicious documents. We used overlapping windows with a single word step. We compared these n-grams of source and suspicious documents to create seeds. Then we merged the seeds that were close enough in both suspicious and source documents in order to obtain plagiarised passages in both source and suspicious documents. After this we applied a postprocessing step in which we

compared the parts of the document that were in the vicinity of already detected passages. Finally, we merged these newly detected passages as well. As mentioned before, out approach includes three steps: seeding, matching and postprocessing. We provide a detailed discussion of each below.

3.1 Seeding

Stopword n-grams We took a list of stopwords as used by Stamatatos in [7]. Using this list as reference, we found out all the stopwords in the document and then created n-grams from these stopwords. After this, we compared each n-gram of the suspicious document and each n-gram of the source document by finding out exact matches between them.

Named entity n-grams Similarly as above, we first found out all the named entities present in the document. Then we took all the word n-grams containing each of these named entities, i.e. if we detected a named entity 'x', we used as seeds all the word n-grams starting from the one that has 'x' as the first word to the n-gram that has 'x' as the last word. We then compared each n-gram in the suspicious document with each n-gram in source document. We found out the count of common words between each n-gram and if it was above a certain threshold called *cmatch*, we considered that n-gram as detection. We took *cmatch* as a function of n.

All word n-grams Finally, we also computed n-grams of all words in the source and the suspicious document. The method used for comparison was similar to that used in the comparison of named entity n-grams.

For all three methods described above, we chose parameters empirically by measuring performance on the training set. We used those values that produced a good balance between precision and recall.

3.2 Matching

After obtaining the seeds, we combined them to form passages. Two detections were merged to create a single passage if they were close enough in both the source and suspicious documents. We considered two detections as being close to each other if the gap between them was less than six words.
For the stopword n-grams method, we performed the matching step separately because doing so was less computationally expensive. We then merged the passages obtained this way with the passages obtained by using the other two methods. Due to this reason, we had many overlapping passages. We dealt with overlapping passages by using the following method. If one passage was fully contained by another in both the source and suspicious documents, we completely removed the shorter passage. If overlap existed between two passages in both the source and suspicious documents, we merged them into a single passage. But if overlap existed between two passages in only one of either the source or suspicious document, we truncated the shorter one (in terms of number of characters) in order to get rid of the overlap.

3.3 Postprocessing

We performed postprocessing based upon the idea that the passages on the vicinity of plagiarised passages are more likely to be plagiarised. In the suspicious document, we took the words that were in the 30-word neighbourhood of a detected passage and computed the n-grams. We then compared these with n-grams computed from the corresponding source passage along with the 30-word neighbourhood of the source passage. We employed a more relaxed all word n-grams method for postprocessing.

Since stopword n-grams and n-grams with named entities are very strict, the probability of obtaining false positives with these is very low. When we used only these two methods, the precision was very high but the recall was high only for no-obfuscation cases. We then tried a more lenient method by making use of all word n-grams. Adding this to the system increased recall for higher obfuscation cases. But now, precision was too low due to a lot of false positives being detected. To the all word n-grams method, we added a criterion that for some detection to be counted, it should constitute at least n+2/3n consecutive detections. Since the method employed for postprocessing also had a tendency to give low precision, we applied the same criteria here as well. We repeated the postprocessing step three times to check for new matches in the boundaries of new detections. At each step of the iteration, we applied the n+2/3n detections criterion and merged the new detections with one another before merging them with the previously detected passages. We allowed a gap of 8 words in this step.

4 Experimental Setting

In this section we provide a description of resources used and the details necessary to implement our system. First and foremost was the task of creating n-grams. For this, we had to begin by dividing the document into words. We removed all the periods in the document and then tokenized the whole document into words by using the PunktWordTokenizer of Python Natural Language Toolkit [2]. We used PunktWordTokenizer because it separates punctuations other than periods. After obtaining this list of words from the documents, we used it to create our n-grams. In order to find out the named entities present in the document, we used the Stanford-NER [3]. Since a single named entity can have more than one word, such as "The University of Alabama at Birmingham", we considered only those n-grams that contained all words of the named entity and not just some words of it.

While seeding, we compared each named entity in the suspicious document with each one in the source. We represented seeds as a pair of suspicious and source n-gram indices. After seeds were obtained, we performed merging. Merging produced passages that were represented as a pair of seeds consisting of the first seed belonging to the passage and the last seed belonging to the passage. For the competition, we needed the results as character indices and lengths. Since converting from word index to character index was a computationally expensive task, we used word indices during the merging step and then converted the obtained passages to character indices rather than converting each detection to character indices first.

In order to merge the seeds, we sorted them by their suspicious document n-gram index

and then by source document n-gram index. Then we iterated over this list and kept on merging the seeds into a single passage until the difference between the index of the last word of the last n-gram in the current passage and the index of the current seed was greater than a predefined threshold either in the source or suspicious document. If there were two detections in the source document for the same n-gram in the suspicious document, this algorithm might split passages. We dealt with this in the case of stopword n-grams by removing the one that would create a shorter passage.
However, named entity n-gram comparison and all word n-gram comparison were not performed by exact matching and only a fraction of the words in the n-grams need to be common for them to be counted as seeds. This creates a different scenario. Let us consider the following case: if n-gram with index *u1* in suspicious document and n-gram with index *r1* in the source document have more than *cmatch* common words, *u1* and *r1* pair will surely be counted as detection. But so will *u1+1* and *r1* pair, *u1* and *r1+1* pair and other such combinations. Even if the number of common words is equal to *cmatch*, this case may occur. So, we did not remove duplicate detections in this case during matching but did so after passages were already obtained.
We removed all passages shorter than n+2/3n in the case of all word n-grams.

5 Results

We tested our system on the corpus provided for the PAN'13 competition. The corpus contains sets of source and suspicious document pairs categorised according to the type of plagiarism in them. The first is a set of non-plagiarised document pairs. Along with that there are four sets of plagiarised document pairs with different levels of obfuscation. The no obfuscation set has documents that have content plagiarised without any obfuscation. Random obfuscation set contains documents with artificially obfuscated passages. The translation obfuscation set has document pairs in which the passages are machine translated into various languages before being translated back to English. The summary obfuscation set of documents have plagiarised passages that have been summarised by humans.
The metrics used in the PAN competition for assessment of the results are macro-averaged precision and recall, granularity and pladget as proposed by Potthast et al [6]. The results obtained for training and test set are shown in Tables 1 and 2 respectively.

Plagiarism Type	Pladget	Precision	Recall	Granularity
No Obfuscation	0.89040	0.99723	0.80425	1.00000
Random Obfuscation	0.67649	0.90482	0.71842	1.27195
Translation Obfuscation	0.62194	0.87069	0.61710	1.23666
Summary Obfuscation	0.12174	0.91405	0.10747	1.98930

Table 1. Evaluation results for the training corpus

The results show that our approach works for the no obfuscation, random obfuscation and translation obfuscation cases. Except for the system that won the competition,

Plagiarism Type	Pladget	Precision	Recall	Granularity
No Obfuscation	0.89369	0.99902	0.80933	1.00083
Random Obfuscation	0.66714	0.92335	0.71461	1.30962
Translation Obfuscation	0.62719	0.88008	0.63618	1.26184
Summary Obfuscation	0.11860	0.90455	0.09897	1.83696

Table 2. Evaluation results for the test corpus

other systems that do well in the no obfuscation case do not perform well in the higher obfuscation cases and vice versa. Their approaches might either be too strict and thus more focused towards detecting no obfuscation plagiarism correctly or more lenient and thus more focused towards detecting plagiarism with higher levels of obfuscation correctly. On the other hand, our approach produces a more balanced result across different forms of plagiarism. We have used both strict (stopwords and named entities) and lenient (all-words) methods as described in previous sections and combined them in a way that the lenient method will not take down the precision. When we first used all-words, the recall for random obfuscation increased but the precision for no obfuscation decreased by more than 20%. After we added the condition that there must be at least n+2/3n consecutive detections for them to be counted, the precision for no obfuscation increased to a similar value as before.

But our system fails to catch most of the plagiarism in the summary obfuscation case. We were not particularly focused on the summary obfuscation case. We will need to do a deeper semantic analysis of the text in order to catch these. But as evidenced by the results of other participants in the competition, better recall in this case is more likely to come at cost of less precision in the no obfuscation case.

Although we removed a lot of short detections and allowed for gaps between seeds so that they would be merged into a single passage, our granularity is not as good as some of the other systems. To improve granularity, we removed passages that were shorter than n+*cmatch* in length. Better granularity could be achieved by removing longer passages but that might hurt recall if we end up removing true positives. We dealt with passages that overlapped only in suspicious document by attributing the overlap to the longer of the two passages. Granularity would be improved by completely removing the other passage but it would hurt recall. Hence we opted not to do so.

Most of the plagiarised parts that we missed consisted of instances where the words were highly shuffled or had comparatively large chunks of text removed from them. Decreasing *cmatch* without changing n to make the system more lenient would be able to catch some of the detections missed by our current system. But doing so will introduce a lot of false positives, hence reducing the precision. Increasing n while keeping the ratio of *cmatch* constant will capture plagiarism in the case where adjacent words have been moved very far from each other. But by doing this, the system will end up missing most of the detections shorter than n and thus decreasing recall. Precision might also be harmed by doing this because even if only a short portion (but still $\geq$ *cmatch*) of the whole n-gram was plagiarised, the whole n-gram will be added to the list of detections.

6 Conclusion

In this paper we have presented an approach to detect plagiarised passages with different levels of obfuscation. Certain methods are more suited towards detecting certain types of obfuscations. We have used three different types of n-grams, each with a different characteristic, so that collectively they can catch passages obfuscated differently. These methods should be combined in such a way that they do not hurt the overall quality of detections of the system. By filtering out those detections that would have decreased our precision, we were able to do precisely that. Along with that, we were able to find a good postprocessing approach that helped in detecting more passages without compromising the precision. In conclusion, we found out that detection of plagiarised passages with different levels of obfuscations by a single system is indeed possible.

7 Acknowledgement

We would like to thank the organizers of the PAN competition for providing us with the opportunity to participate in this event. This research is partially funded by The Office of Naval Research under grant N00014-12-1-0217.

References

1. Barrón-Cedeño, A., Rosso, P.: On automatic plagiarism detection based on n-grams comparison. In: Advances in Information Retrieval, pp. 696–700. Springer (2009)
2. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python. O'Reilly Media, Inc., 1st edn. (2009)
3. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by Gibbs sampling. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics. pp. 363–370. Association for Computational Linguistics (2005), http://nlp.stanford.edu/ manning/papers/gibbscrf3.pdf
4. Kong, L., Qi, H., Wang, S., Du, C., Wang, S., Han, Y.: Approaches for candidate document retrieval and detailed comparison of plagiarism detection. In: CLEF (Online Working Notes/Labs/Workshop) (2012)
5. Potthast, M., Gollub, T., Hagen, M., Kiesel, J., Michel, M., Oberländer, A., Tippmann, M., Barrón-Cedeño, A., Gupta, P., Rosso, P., Stein, B.: Overview of the 4th international competition on plagiarism detection. In: Forner, P., Karlgren, J., Womser-Hacker, C. (eds.) CLEF (Online Working Notes/Labs/Workshop) (2012)
6. Potthast, M., Stein, B., Barrón-Cedeño, A., Rosso, P.: An evaluation framework for plagiarism detection. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters. pp. 997–1005. Association for Computational Linguistics (2010)
7. Stamatatos, E.: Plagiarism detection using stopword n-grams. Journal of the American Society for Information Science and Technology 62(12), 2512–2527 (2011), http://dx.doi.org/10.1002/asi.21630
8. Suchomel, Š., Kasprzak, J., Brandejs, M.: Three way search engine queries with multi-feature document comparison for plagiarism detection. In: CLEF (Online Working Notes/Labs/Workshop). pp. 0–12 (2012)