Machine Translation Evaluation Metric for Text Alignment

Notebook for PAN at CLEF 2014

Prasha Shrestha, Suraj Maharjan, and Thamar Solorio

University of Alabama at Birmingham
Department of Computer and Information Sciences,
Campbell Hall, 1300 University Boulevard,
Birmingham, Alabama 35294-1170
{prasha, suraj, solorio}@cis.uab.edu

Abstract As plagiarisers become cleverer, plagiarism detection becomes harder. Plagiarisers will find new ways to obfuscate the plagiarized passages so that humans and automatic plagiarism detectors are not able to point them out. So, a plagiarism detection system needs to be robust enough to detect plagiarism, no matter what obfuscation techniques have been applied. Our system attempts to do the same by combining two different methods, one stricter to catch mildly obfuscated passages and one more lenient to catch difficult ones. We use a machine translation evaluation metric and n-gram matching to detect overlaps between the source and suspicious documents. On the PAN'14 corpus, which contains data with various types of obfuscation to mimic human plagiarisers, we obtained plagdet scores of 0.84404 and 0.86806 on the two datasets.

1 Introduction

The first step in plagiarism detection is source retrieval, which finds the source documents from which a given document may have been plagiarised from. But we cannot stop here because at this point we have no solid proof of plagiarism. In order to get this proof, we need to find corresponding plagiarised passages in both the source and suspicious documents. This process, called text alignment is the focus of this paper. Text alignment is important because without it, we will not be able to provide reasoning for why we believe that the document has been plagiarised.

Text alignment may seem like simple text matching, but it is hardly so because a plagiariser rarely copies the text verbatim. Instead it is more likely to have been changed significantly in order to make it seem like an original piece of text. The task is now to find the same as well as similar pieces of text between the two documents. Also, there might be different levels of obfuscation applied to the text. A single system must be able to catch all these passages, from those that are perfectly similar to each other to those that only have a slight resemblance to one another. If the system is too strict, it might only catch those passages that are very similar and if it is too lenient it might catch a lot of false positives. This raises the difficulty level for the task.

In our system, we have tried to find a balance between a strict and a lenient system. We used two different methods, one stricter and one more lenient and then combined

the results obtained from them. Our strict system uses TER-p [4] and to capture the matches between two sentences. TER-p is a machine translation evaluation metric that finds the word edit distance from a hypothesis translation to a reference translation [4]. In TER-p along with matches, insertions, deletions, substitutions and shifts, it also considers stem and synonym matches as well as phrasal substitutions as edits. The edit costs in TER-p have been tuned to correlate with human judgments of how close an instance of machine translation is to corresponding reference translations. Since, it measures the edit distance, this metric can be used in tasks that need to measure similarity in text. Some of the creators of the metric themselves used this measure for paraphrase detection [1]. In that work, they had also tried to use other machine translation metrics for the same task, but TER-p produced the best results. Our TER-p method will produce a detection only when the similarity between two sentences is high. So, it will not consider similarity between fragments of sentences or between phrases. Our more lenient system uses n-grams, more specifically bigrams, to generate matches. Bigrams are very lenient and will more likely catch most of the plagiarised passages. But as in any lenient system, it is also inclined to catch a lot of coincidental matches. But when we combine the detections from these two methods, we merge those that are close to each other. So, we will catch sentence fragments or phrases around the passage caught by the TER-p method. Also, if the bigram detections are indeed from a plagiarised passage, there will be a high density of bigram detections in that area of the text, which will form a large passage. In the end, we will only keep large enough passages and throw out the rest. So, the coincidental matches are likely to be removed and we will end up with only the correct detections.

2 Methodology

Our system follows the same basic steps of any plagiarism detection system: seeding, match merging and postprocessing. The two methods that we have used to generate matches, produce different seeds: the method that uses TER-p scores finds matches between sentences and the other method finds matches between n-grams. So, the natural first step was to generate sentences and n-grams from the text.

The next step is to find matches between sentence pairs and n-gram pairs. Given a hypothesis and a reference file, TER-p gives the number of edits required to transform the sentences in the hypothesis file to the sentences in the reference file. It also provides scores per sentence, which we have used. TER-p score ranges from 0 to 1, with 0 being completely similar and 1 being most dissimilar. We considered the source document as the reference file and the suspicious document as the hypothesis file. We obtained TER-p scores for each sentence in the source compared with each sentence in the suspicious document. We then considered two sentences to be plagiarised if their TER-p score is above a certain value, which in our final system is set at 0.9. For n-grams, we performed exact matching. We used bigrams and thus there were a lot of false matches. To remove false positives we merged the detections from bigrams separately before combining it with results from the TER-p system and took only 1 matching bigram in source for each bigram in the suspicious document.

After we obtained matches from both methods, we merged them to obtain passages. First we merged all the matches that were close to each other by a certain threshold in both the source and the suspicious documents. We had tried to set this threshold according to the length of the matching passages to be merged. The longer the passages, the larger would be the threshold. But we ended up using a fixed threshold of 80 characters because it produced better results. We repeated this merging process until the passages could not be merged anymore.

We also performed a postprocessing step after we found the matching passages. We kept our postprocessing step very simple by just removing the passages that were shorter than 200 characters. Although this reduced the recall by a small amount, we were able to obtain better granularity. The values for all the parameters and thresholds were chosen empirically as those that produce a good balance between precision and recall, while keeping the granularity as close to 1 as possible.

3 Results and Discussion

Our results on the PAN'14 text alignment corpus for the training and test datasets are shown in Tables 1 and 2 respectively. The evaluation metrics used are macro-averaged precision and recall, granularity and plagdet [2]. The plagiarised passages in the data have different levels of obfuscation.

Table 1. Evaluation results for the training corpus

Plagiarism Type	Plagdet	Precision	Recall	Granularity
No Obfuscation	0.88009	0.80287	0.97374	1.00000
Random Obfuscation	0.86150	0.89992	0.83358	1.00642
Translation Obfuscation	0.85427	0.85322	0.86086	1.00447
Summary Obfuscation	0.15853	0.98171	0.08975	1.05263

Table 2. Evaluation results for the test corpus

Dataset	Plagdet	Precision	Recall	Granularity
PAN'14 Test Corpus 2	0.84404	0.85906	0.83782	1.00701
PAN'14 Test Corpus 3	0.86806	0.84418	0.89839	1.00381

Our method performs the best when there is no obfuscation, which is what we expected. But even with random and translation obfuscations, we are very close to the plagdet we obtain for no obfuscation. We also have a good balance between precision and recall. When we participated last year, we had used larger values of n [3] while this year we used bigrams. When we use just the bigrams and leave out results from the part of the system that uses TER-p, we get plagdet of 0.92880, 0.74728, 0.67112, 0.09335 for the different levels of obfuscation in order on the training data. Our last

year's system had obtained plagdet scores of 0.89040, 0.67649, 0.62194, 0.12174 for respective levels of obfuscation on the same data. Even with just bigrams we get better results than last year's system. This shows that smaller n is better when trying to find matches between two texts. This might be because smaller n-gram matches can always be merged to get larger n-grams but the reverse is not possible.

The part of the dataset where our method fails to catch plagiarised passages is summary obfuscation. Here, the plagiarised passages have been summarized and then inserted into the suspicious document. We have not used any summarizer in our system, which might be why we are not able to catch these. Since several sentences are summarized into one, the TER-p method is unable to find exact matching sentences. Also, the bigram method catches some of the matching bigrams but since they are too far apart in the source document, they are not merged. They remain as short passages and are subsequently removed in the postprocessing step.

The postprocessing step did help to lower our granularity. Without this step, granularity was high and this brought down our plagdet significantly, although we had good values for precision and recall. So, this step helped lower the granularity considerably and in turn increase the plagdet score. But for summary obfuscation even after this postprocessing step, granularity is close to 2. This is because the detected passages, even when they are long enough to not be removed by the postprocessing step are the too sparse and too far between.

All in all, we were able to get good results as well as balanced precision and recall for all the obfuscation levels except for summary obfuscation. For the obfuscations on which we did perform well, we have similar plagdet scores that show that our system is equally capable of detecting these varieties of obfuscations.

4 Conclusion and Future Work

Our idea of using one strict and one lenient method did work in catching most types of obfuscations. We also improved on our results from last year although we used a significantly different system. We were able to use TER-p, which is a machine translation metric for a task that it was not designed for but was definitely suitable for. This also opens the doors to explore other similar metrics that can be used for similarity detection, which we plan to add to our system.

Acknowledgment

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

References

1. Madnani, N., Tetreault, J., Chodorow, M.: Re-examining machine translation metrics for paraphrase identification. In: Proceedings of the 2012 Conference of the North American

Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 182–190. Association for Computational Linguistics (2012)
2. 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)
3. Shrestha, P., Solorio, T.: Using a variety of n-grams for the detection of different kinds of plagiarism. In: Notebook for PAN at CLEF 2013. Valencia, Espana (2013)
4. Snover, M.G., Madnani, N., Dorr, B., Schwartz, R.: Ter-plus: paraphrase, semantic, and alignment enhancements to translation edit rate. Machine Translation 23(2-3), 117–127 (2009)