Arabic Plagiarism Detection Using Word Correlation in N-Grams with K-Overlapping Approach

Working Notes for PAN-AraPlagDet at FIRE 2015

Salha Alzahrani
College of Computing and Information Technology, Taif Univeristy, Taif, Saudi Arabia
s.zahrani@tu.edu.sa

ABSTRACT

This report explains our Arabic plagiarism detection system which we used to submit our run to AraPlagDetect competition at FIRE 2015. The system was constructed through four main stages. First is pre-processing which includes tokenisation and stop words removing. Second is retrieving a list of candidate documents for each suspicious document using K-gram fingerprinting and Jaccard coefficient. Suspicious documents are then compared in-depth with the associated candidate documents. This stage entails the computation of the similarity between constructed N-grams with K-overlapping where N and K were experimentally assigned to 8 and 3, respectively. The similarity between N-Gram pairs were computed based on word correlations. Each word was compared with words in candidate N-Gram and correlated by 1 if they are matched. Correlation values were averaged then compared to a threshold. The last step is post-processing whereby consecutive N-Grams were joined to form united plagiarised segments. Our performance measures on the training corpus were encouraging (recall=0.829, precision=0.843, granularity=1.11). The recall measure on the test collection was unfortunately less (recall= 0.530) but precision and granularity remained consistent with the train set (precision= 0.831, granularity= 1.18). This drop in recall may be due to the fact that our candidate retrieval stage retrieves only documents which share copied fragments but there exist plagiarised documents which have no exact-copied cases. Although this system can detect some means of obfuscation such as restructuring or rewording of few phrases, it might not work with handmade paraphrases. Our future work is to advance the candidate retrieval stage and contain semantic-based metrics in the detection stage.

1. INTRODUCTION

Methods for Arabic plagiarism detection can easily track verbatim plagiarism; however, finding excessive plagiarism cases on a high-scale dataset is challenging. Many current techniques rely on exactly matched substrings or some kinds of textual fingerprinting. This paper aims to detect cases of rephrasing and rewording the content in Arabic texts. In this regard, matching fragments of text can be approximated by correlating the words from two candidate texts. In this paper, the problem considered is stated as follows. Given a suspicious document dataset D_q and a large source collection D wherein a small subset D_x is similar to a suspicious document d_q, the task is to find all suspicious parts s_q from each $d_q : d_q \epsilon D_q$ that are similar to parts s_x from $d_x : d_x \epsilon D_x$ which is known as external plagiarism detection [1]. This work used on word correlations from N-grams with k-overlapping approach similar to our previous work in PAN'10 [2]. The languages of both suspicious and candidate documents are written in Arabic.

The steps used in our system are as follows. First, extract a set of features for each $d_q \epsilon D_q$ and $d \epsilon D$. Second, find a list of most promising documents D_x where $D_x \subset D$ based on K-gram fingerprinting and Jaccard similarity coefficient. Third, perform N-gram with K-overlapping segmentation, then perform comparison between N-grams using word correlation approach. Last, perform post-processing operations to merge subsequent similar statements into passages or paragraphs. As can be seen, intrinsic plagiarism detection (i.e. via variations in writing styles) is not handled by this work.

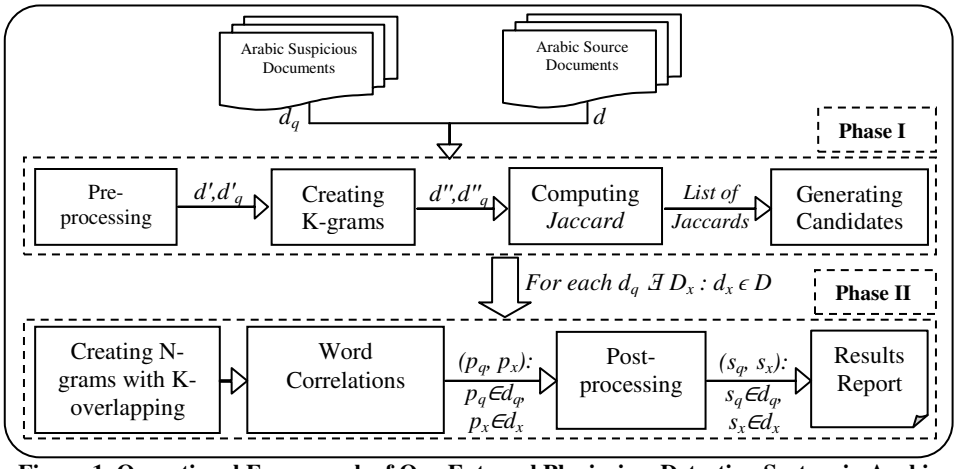

Figure 1. Operational Framework of Our External Plagiarism Detection System in Arabic

2. EXTERNAL PLAGIARISM DETECTION IN ARABIC TEXTS

2.1 General Framework

This work implements a system that tackles monolingual external plagiarism detection. The operational framework is shown in Figure 1. The process starts with a set of source collection D and suspicious/query documents D_q**.** Then new representatives namely d' and d'_q are generated for cleansed and tokenised d_q and d documents respectively. Then, d' and d'_q are used for K-gram fingerprinting and computing the similarity between the shingles. The list of most similar documents for each $d_q \in D_q$ is called candidate set D_x whereby $D_x \subset D$. After generating D_x, detailed analysis stage is performed to obtain similar N-grams (s_q, s_x) where $s_q \in d_q$, $s_x \in d_x$. The similarity score is gained by implementing a word-based correlation approach between words in both N-grams as will be seen shortly. Finally, the system performs post-processing in order to merge similar N-grams into passages (p_q, p_x) such that $p_q \in d_q$, $p_x \in d_x$. Subsequent sections detail each stage.

2.2 Phase I: Retrieval of Candidate Documents

Near duplicate detection methods can be used to bring similar sources and discard dissimilar ones. We use K-gram fingerprinting and Jaccard coefficient approach [3]. The *k*- gram referred to a sequence of consecutive words of size *k*. The value for *k* is typically 3 or 4. Intuitively, two documents *A* and *B* are similar if they share enough *k*-grams. By performing union and intersection operations between the *k*-grams, we can find the Jaccard similarity coefficient between *A* and *B* as stated in equation (1).

$$J(A,B) = |K\text{-}grams\_A \cap K\text{-}grams\_B| \,/\, |K\text{-}grams\_A \cup K\text{-}grams\_B| \quad (1)$$

Therefore for each suspicious document d_q, documents of Jaccard coefficient above a threshold value are taken to form the set of candidate documents D_x. We set the threshold of $Jaccard \geq 0.1$ because we found that this value derives about 1 to 20 candidate documents for each d_q. It was found that when we compare documents of Jaccard similarity less than *0.1*, either none or about 1-2 plagiarised statements are detected. Also by using this method, we find that some suspicious documents do not have any candidates. That means that they might contain very obfuscated plagiarism or do not contain plagiarism at all. More interesting, using this approach assumes that the number of candidates for each suspicious document is dynamic and may be small. That saves the computation time in contrast to having a fixed number of candidates for each suspicious document.

2.3 Phase II: In-Depth Detailed Analysis of (d_q , d_x) pairs

At this stage, a detailed analysis between each suspicious document d_q and its candidate document $d_x \in D_x$ is performed. At first, d_q and d_x are segmented into N-grams with K-overlapping S_q and S_x respectively. N and K were experimentally assigned to 8 and 3 because these values were found the best to achieve optimum precision and recall on the training set. An example of this segmentation is shown in Figure 2.

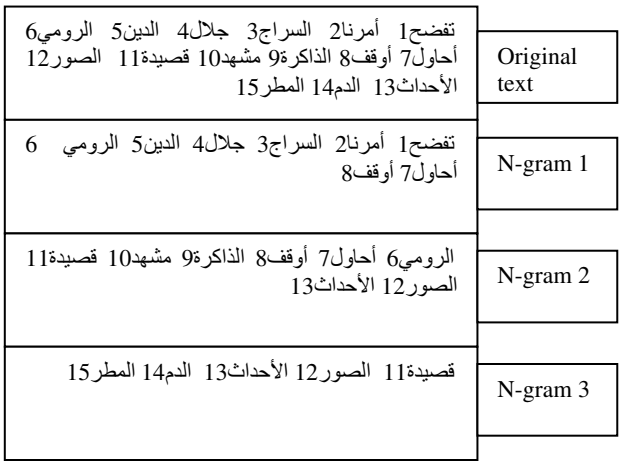

Figure 2. N-gram with K-overlapping segmentation, N=8, K=3

Next, to compute the similarity between two N-grams (s_q, s_x), a word to N-gram correlation factor for each word w_q in s_q and the N-gram s_x is computed as in [4]:

$$\mu_{q,x} = 1 - \prod w_k \in S_x (1 - F_{q,k}) \quad (2)$$

where w_k are words in s_x and $F_{q,k}$ is proposed as follows:

$$F_{q,k} = \begin{cases} 1 & \text{if } w_k \text{ and } w_q \text{ are identical} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

For example,

S_1="تفضح أمرنا السراج جلال الدين الرومي أوقف" and

S_2="تفضح أمرنا الرومي السراج جلال الدين"

are almost identical except for one word and with some restructuring. The similarity between (s_q, s_x) is expressed using the equation

$$Sim(s_q, s_x) = (\mu_{1,x} + \mu_{2,x} + \ldots + \mu_{q,x} + \ldots + \mu_{n,x}) / n \quad (4)$$

where n is the total number of words in s_q. Thus, we can calculate the degree of similarity between the S_1 and S_2 as shown in Figure 3 taking into account that stop words were removed. This indicates that N-grams are very similar. Thus to judge two sentences as equal (i.e. plagiarised), the minimum similarity score should be above a threshold value as stated in equation (5). The value of α was set to 0.85 in the experimental works. According to this, the pair shown in Figure 3 is considered similar (i.e. plagiarised) because the minimum similarity is greater than α.

$$EQ(s_q, s_x) = \begin{cases} 1 & \text{if } MIN(Sim(s_q, s_x), Sim(s_x, s_q)) \geq \alpha \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

Finally, the output of this algorithm is a list of pairs (s_q, s_x): $s_q \in d_q$, $s_x \in d_x$, $d_x \in D$ marked as similar/plagiarised. Because of using N-grams as comparison scheme, post-processing is required to merge subsequent sentences marked as plagiarised into passages. Also, we consider small distances under the predicate *less than or equal to 300 characters* to merge subsequent plagiarised N-grams and their corresponding source N-Grams into passages pairs $(p_q, p_x) : p_q \in d_q, p_x \in d_x, d_x \in D_x$.

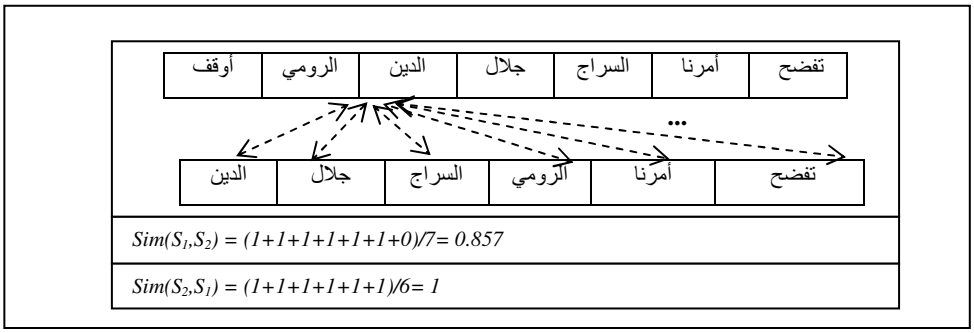

Figure 3. Examples of N-gram pairs and word correlation similarity approach

3. EXPERIMENTAL SETUP

3.1 Instrumentation

We used a server with 16-core processors, 3 GHz. To utilise all cores, we worked on C#.NET 2010 which has introduced the concept of parallel computing[1].

3.2 Code Configuration

Below is a list of some parameters and settings that we have configured in our code.

- For pre-processing, stop words removal were used.
- For generating *k-grams*, the k was set to 3 (i.e. *word 3-grams*).
- For computing Jaccard similarity and finding candidates, a threshold value of *Jaccard=0.1* was set to filter out non-candidate documents.
- For plagiarism detection, the equations (2)-(5) were employed, and the threshold in (5) was set to α= 0.85 which was found to be the most suitable based on our experimental trials.

3.3 Run Time

The run time of the training set was 12 hours (using parallel computing on 16 cores). The same time (12 hours) was consumed for the test set using parallel computing on 16 cores.

4 Evaluation and Conclusion

Our results from the training and test datasets are shown in Table 1. The results from training set showed that we detected about 83% of the plagiarism cases and about 85% of the detections were correct. These results are highly encouraging and good for Arabic plagiarism systems. The experimental works on the test set shows high precision where 83% of detected cases were correct but moderate recall equals to about 53%. The decrement in recall between the training and test sets might be due to these reasons: (i) unlike in the training set, the candidate retrieval stage was not sufficient to retrieve all relevant documents to the plagiarism cases in the test set, (ii) we used high threshold in the detailed analysis stage (α= 0.85) but in previous works it was set between 0.75-0.65. Our future work is to improve it for more detection efficiency and less time complexity. We will consider using semantic measures for simulated and hand-made plagiarism cases.

Table 1. Results of our system.

Dataset	α	*Recall*	*Precision*	*Granularity*	*Plagdet*
Train	0.85	0.829	0.843	1.11	0.707
Test	0.85	0.530	0.831	1.18	0.574

4. ACKNOWLEDGEMENTS

We thank the organizers of PAN at FIRE 2015 evaluation lab, Imene Bensalem in particular, for her generous support and instantaneous answer through AraPlagDet mailing list.

5. REFERENCES

[1] Potthast M, et al. *An evaluation framework for plagiarism detection*. in 23rd International Conference on Computational Linguistics (COLING 2010). 2010, Beijing, China.

[2] Alzahrani S M and Salim N. *Fuzzy semantic-based string similarity for extrinsic plagiarism detection: Lab report for pan at clef'10*. in 4th International Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN-10) in conjunction with CLEF'10. 2010, Padua, Italy.

[3] Manning C D, Raghavan P, and Schütze H, *Web search basics: Near-duplicates and shingling, in Introduction to information retrieval.* 2009, Cambridge University Press. p. 437-442.

[4] Yerra R and Ng Y-K, *A sentence-based copy detection approach for web documents*, in Fuzzy systems and knowledge discovery. 2005. p. 557-570.

[1] http://channel9.msdn.com/learn/courses/VS2010/Parallel/