Online Plagiarism Detection Through Exploiting Lexical, Syntactic, and Semantic Information

Wan-Yu Lin
Graduate Institute of Networking and Multimedia, National Taiwan University
r99944016@csie.ntu.edu.tw

Nanyun Peng
Institute of Computational Linguistic, Peking University
pengnanyun@pku.edu.cn

Chun-Chao Yen
Graduate Institute of Networking and Multimedia, National Taiwan University
r96944016@csie.ntu.edu.tw

Shou-de Lin
Graduate Institute of Networking and Multimedia, National Taiwan University
sdlin@csie.ntu.edu.tw

Abstract

In this paper, we introduce a framework that identifies online plagiarism by exploiting lexical, syntactic and semantic features that includes duplication-gram, reordering and alignment of words, POS and phrase tags, and semantic similarity of sentences. We establish an ensemble framework to combine the predictions of each model. Results demonstrate that our system can not only find considerable amount of real-world online plagiarism cases but also outperforms several state-of-the-art algorithms and commercial software.

Keywords

Plagiarism Detection, Lexical, Syntactic, Semantic

1. Introduction

Online plagiarism, the action of trying to create a new piece of writing by copying, reorganizing or rewriting others' work identified through search engines, is one of the most commonly seen misusage of the highly matured web technologies. As implied by the experiment conducted by (Braumoeller and Gaines, 2001), a powerful plagiarism detection system can effectively discourage people from plagiarizing others' work. A common strategy people adopt for online-plagiarism detection is as follows. First they identify several suspicious sentences from the write-up and feed them one by one as a query to a search engine to obtain a set of documents. Then human reviewers can manually examine whether these documents are truly the sources of the suspicious sentences. While it is quite straightforward and effective, the limitation of this strategy is obvious. First, since the length of search query is limited, suspicious sentences are usually queried and examined independently. Therefore, it is harder to identify document level plagiarism than sentence level plagiarism. Second, manually checking whether a query sentence plagiarizes certain websites requires specific domain and language knowledge as well as considerable amount of energy and time. To overcome the above shortcomings, we introduce an online plagiarism detection system using natural language processing techniques to simulate the above reverse-engineering approach. We develop an ensemble framework that integrates lexical, syntactic and semantic features to achieve this goal. Our system is language independent and we have implemented both Chinese and English versions for evaluation.

2. Related Work

Plagiarism detection has been widely discussed in the past decades (Zou et al., 2010). Table 1. summarizes some of them:

Author	Comparison Unit	Similarity Function
Brin et al., 1995	Word + Sentence	Percentage of matching sentences.
White and Joy, 2004	Sentence	Average overlap ratio of the sentence pairs using 2 pre-defined thresholds.
Niezgoda and Way, 2006	A human defined sliding window	Sliding windows ranked by the average length per word.
Cedeno and Rosso, 2009	Sentence + n-gram	Overlap percentage of n-gram in the sentence pairs.

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 145–150,
Jeju, Republic of Korea, 8-14 July 2012.

Pera and Ng, 2010	Sentence	A pre-defined resemblance function based on word correlation factor.
Stamatatos, 2011	Passage	Overlap percentage of stopword n-grams.
Grman and Ravas, 2011	Passage	Matching percentage of words with given thresholds on both ratio and absolute number of words in passage.

Table 1. Summary of related works

Comparing to those systems, our system exploits more sophisticated syntactic and semantic information to simulate what plagiarists are trying to do.

There are several online or charged/free downloadable plagiarism detection systems such as Turnitin, EVE2, Docol© c, and CATPPDS which detect mainly verbatim copy. Others such as Microsoft Plagiarism Detector (MPD), Safeassign, Copyscape and VeriGuide, claim to be capable of detecting obfuscations. Unfortunately those commercial systems do not reveal the detail strategies used, therefore it is hard to judge and reproduce their results for comparison.

3. Methodology

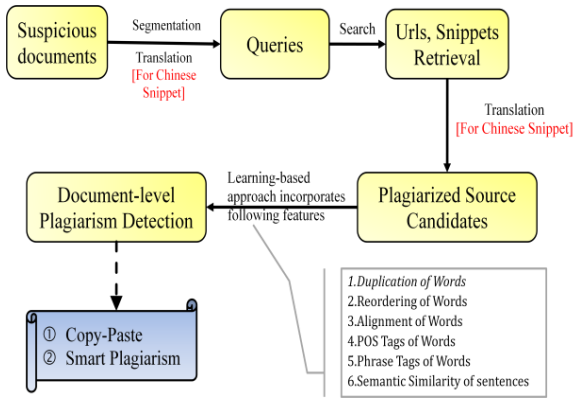

Figure 1. Detection Flow

The data flow is shown above in Figure 1.

3.1 Query a Search Engine

We first break down each article into a series of queries to query a search engine. Several systems such as (Liu at al., 2007) have proposed a similar idea. The main difference between our method and theirs is that we send *unquoted* queries rather than quoted ones. We do not require the search results to completely match to the query sentence. This strategy allows us to not only identify the copy/paste type of plagiarism but also re-write/edit type of plagiarism.

3.2 Sentence-based Plagiarism Detection

Since not all outputs of a search engine contain an exact copy of the query, we need a model to quantify how likely each of them is the source of plagiarism. For better efficiency, our experiment exploits the snippet of a search output to represent the whole document. That is, we want to measure how likely a snippet is the plagiarized source of the query. We designed several models which utilized rich lexical, syntactic and semantic features to pursue this goal, and the details are discussed below.

3.2.1 Ngram Matching (NM)

One straightforward measure is to exploit the n-gram similarity between source and target texts. We first enumerate all n-grams in source, and then calculate the overlap percentage with the n-grams in the target. The larger n is, the harder for this feature to detect plagiarism with insertion, replacement, and deletion. In the experiment, we choose n=2.

3.2.2 Reordering of Words (RW)

Plagiarism can come from the reordering of words. We argue that the permutation distance between S_1 and S_2 is an important indicator for reordered plagiarism. The permutation distance is defined as the minimum number of pair-wise exchanging of matched words needed to transform a sentence, S_2, to contain the same order of matched words as another sentence, S_1. As mentioned in (Sörensena and Sevaux, 2005), the permutation distance can be calculated by the following expression $d(S_1, S_2) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} z_{ij}$
where

$$z_{ij} = \begin{cases} 1, & if\ S_1(j) > S_1(i) and\ S_2(j) < S_2(i) \\ 0, & otherwise \end{cases}$$

$S_1(i)$ and $S_2(i)$ are indices of the i^{th} matched word in sentences S_1 and S_2 respectively and n is the number of matched words between the sentences S_1 and S_2. Let $\mu = \frac{n^2 - n}{2}$ be the normalized term, which is the maximum possible distance between S_1 and S_2, then the reordering

score of the two sentences, expressed as $s(S_1, S_2)$, will be $s(S_1, S_2) = 1 - \frac{d(S_1,S_2)}{\mu}$

3.2.3 *Alignment of Words (AW)*

Besides reordering, plagiarists often insert or delete words in a sentence. We try to model such behavior by finding the alignment of two word sequences. We perform the alignment using a dynamic programming method as mentioned in (Wagner and Fischer, 1975).

However, such alignment score does not reflect the continuity of the matched words, which can be an important cue to identify plagiarism. To overcome such drawback, we modify the score as below.

$$New\ Alignment\ Score = \frac{\sum_{i=1}^{|M|-1} G_i}{|M|-1}$$

where $G_i = \frac{1}{\#\ of\ words\ \ between\ \ (M_i, M_{i+1})+1}$

M is the list of matched words, and M_i is the i^{th} matched word in M. This implies we prefer fewer unmatched words in between two matched ones.

3.2.4 *POS and Phrase Tags of Words (PT, PP)*

Exploiting only lexical features can sometimes result in some false positive cases because two sets of matched words can play different roles in the sentences. See S_1 and S_2 in Table 2. as a possible false positive case.

S_1: The man likes the woman				
S_2: The woman is like the man				
Word	S_1: Tag	S_2: Tag	S_1: Phrase	S_2: Phrase
man	NN	NN	NP	PP
like	VBZ	IN	VP	PP
woman	NN	NN	VP	NP

Table 2. An example of matched words with different tags and phrases

Therefore, we further explore syntactic features for plagiarism detection. To achieve this goal, we utilize a parser to obtain POS and phrase tags of the words. Then we design an equation to measure the tag/phrase similarity.

$$Sim = \frac{num\ (matc\ hed\ \ words\ \ wit\ h\ identical\ \ \ tag\ )}{num\ (matc\ hed\ \ words\ )}$$

We paid special attention to the case that transforms a sentence from an active form to a passive-form or vice versa. A subject originally in a Noun Phrase can become a Preposition Phrase, i.e. "by …", in the passive form while the object in a Verb Phrase can become a new subject in a Noun Phrase. Here we utilize the Stanford Dependency provided by Stanford Parser to match the tag/phrase between active and passive sentences.

3.2.5 *Semantic Similarity (LDA)*

Plagiarists, sometimes, change words or phrases to those with similar meanings. While previous works (Y. Lin et al., 2006) often explore semantic similarity using lexical databases such as WordNet to find synonyms, we exploit a topic model, specifically latent Dirichlet allocation (LDA, D. M. Blei et al., 2003), to extract the semantic features of sentences. Given a set of documents represented by their word sequences, and a topic number n, LDA learns the word distribution for each topic and the topic distribution for each document which maximize the likelihood of the word co-occurrence in a document. The topic distribution is often taken as semantics of a document. We use LDA to obtain the topic distribution of a query and a candidate snippet, and compare the cosine similarity of them as a measure of their semantic similarity.

3.3 Ensemble Similarity Scores

Up to this point, for each snippet the system generates six similarity scores to measure the degree of plagiarism in different aspects. In this stage, we propose two strategies to linearly combine the scores to make better prediction. The first strategy utilizes each model's predictability (e.g. accuracy) as the weight to linearly combine the scores. In other words, the models that perform better individually will obtain higher weights. In the second strategy we exploit a learning model (in the experiment section we use Liblinear) to learn the weights directly.

3.4 Document Level Plagiarism Detection

For each query from the input article, our system assigns a degree-of-plagiarism score to some plausible source URLs. Then, for each URL, the system sums up all the scores it obtains as the final score for document-level degree-of-plagiarism. We set up a cutoff threshold to obtain the most plausible URLs. At the end, our system highlights the suspicious areas of plagiarism for display.

4. Evaluation

We evaluate our system from two different angles. We first evalaute the *sentence level plagirism detection* using the PAN corpus in English. We then evaluate the capability of the full system to detect on-line plagiarism cases using annotated results in Chinese.

4.1 Sentence-based Evaluations

We want to compare our model with the state-of-the-art methods, in particular the winning entries in plagiarism detection competition in PAN [1]. However, the competition in PAN is designed for off-line plagiarism detection; the entries did not exploit an IR system to search the Web like we do. Nevertheless, we can still compare the core component of our system, the *sentence-based measuring* model with that of other systems. To achieve such goal, we first randomly sampled 370 documents from PAN-2011 external plagiarism corpus (M. Potthast et al., 2010) containing 2882 labeled plagiarism cases.

To obtain high-quality negative examples for evaluation, we built a full-text index on the corpus using Lucene package. Then we use the suspicious passages as queries to search the whole dataset using Lucene. Since there is length limitation in Lucene (as well as in the real search engines), we further break the 2882 plagiarism cases into 6477 queries. We then extract the top 30 snippets returned by the search engine as the potential negative candidates for each plagiarism case. Note that for each suspicious passage, there is only one target passage (given by the ground truth) that is considered as a positive plagiarism case in this data, and it can be either among these 30 cases or not. However, we union these 30 cases with the ground truth as a set, and use our (as well as the competitors') models to rank the degree-of-plagiarism for all the candidates. We then evaluate the rank by the area-under-PR-curve (AUC) score. We compared our system with the winning entry of PAN 2011 (Grman and Ravas, 2011) and the stopword ngram model that claims to perform better than this winning entry by Stamatatos (2011). The results of each individual model and ensemble using 5-fold cross validation are listed in Table 3. It shows that NM is the best individual model, and an ensemble of three features outperforms the state-of-the-art by 26%.

NM	*RW*	*AW*	*PT*	*PP*	*LDA*
0.876	0.596	0.537	0.551	0.521	0.596

(a)

	Ours ensemble	*Pan-11 Champion*	*Stopword Ngram*
AUC	0.882 (NM+RW+PP)	0.620	0.596

(b)

Table 3. (a) AUC for each individual model (b) AUC of our ensemble and other state-of-the-art algorithms

4.2 Evaluating the Full System

To evaluate the overall system, we manually collect 60 real-world review articles from the Internet for books (20), movies (20), and music albums (20). Unfortunately for an online system like ours, there is no ground truth available for recall measure. We conduct two differement evalautions. First we use the 60 articles as inputs to our system, ask 5 human annotators to check whether the articles returned by our system can be considered as plagiarism. Among all 60 review articles, our system identifies a considerablely high number of copy/paste articles, 231 in total. However, identifying this type of plagiarism is trivial, and has been done by many similar tools. Instead we focus on the so-called *smart-plagiarism* which cannot be found through quoting a query in a search engine. Table 4. shows the precision of the smart-plagiarism articles returned by our system. The precision is very high and outperforms a commertial tool Microsoft Plagiarism Detector.

	Book	*Movie*	*Music*
Ours	280/288 (97%)	88/110 (80%)	979/1033 (95%)
MPD	44/53 (83%)	123/172 (72%)	120/161 (75%)

Table 4. Precision of Smart Plagiarism

In the second evaluation, we first choose 30 reviews randomly. Then we use each of them as queries into Google and retrieve a total of 5636 pieces of snippet candidates. We then ask 63 human beings to annotate whether those snippets represent plagiarism cases of the original review article. Eventually we have obtained an annotated

[1] The website of PAN-2011 is http://pan.webis.de/

dataset and found a total of 502 plagiarized candidates with 4966 innocent ones for evalaution. Table 5. shows the average AUC of 5-fold cross validation. The results show that our method outperforms the Pan-11 winner slightly, and much better than the Stopword Ngram.

NM	*RW*	*AW*	*PT*	*PP*	*LDA*
0.904	0.778	0.874	0.734	0.622	0.581

(a)

	Ours ensemble	*Pan-11 Champion*	*Stopword Ngram*
AUC	0.919 (NM+RW+AW +PT+PP+LDA)	0.893	0.568

(b)

Table 5. (a) AUC for each individual model (b) AUC of our ensemble and other state-of-the-art algorithms

4.3 Discussion

There is some inconsistency of the performance of single features in these two experiments. The main reason we believe is that the plagiarism cases were created in very different manners. Plagiarism cases in PAN external source are created artificially through word insertions, deletions, reordering and synonym substitutions. As a result, features such as word alignment and reordering do not perform well because they did not consider the existence of synonym word replacement. On the other hand, real-world plagiarism cases returned by Google are those with matching-words, and we can find better performance for *AW*.

The performances of syntactic and semantic features, namely *PT*, *PP and LDA*, are consistently inferior than other features. It is because they often introduce false-positives as there are some non-plagiarism cases that might have highly overlapped syntactic or semantic tags. Nevertheless, experiments also show that these features can improve the overall accuracy in ensemble.

We also found that the *stopword Ngram* model is not applicable universally. For one thing, it is less suitable for on-line plagiarism detection, as the length limitation for queries diminishes the usability of stopword n-grams. For another, Chinese seems to be a language that does not rely as much on stopwords as the latin languages do to maintain its syntax structure.

Samples of our system's finding can be found here, http://tinyurl.com/6pnhurz

5. Online Demo System

We developed an online demos system using JAVA (JDK 1.7). The system currently supports the detection of documents in both English and Chinese. Users can either upload the plain text file of a suspicious document, or copy/paste the content onto the text area, as shown below in Figure 2.

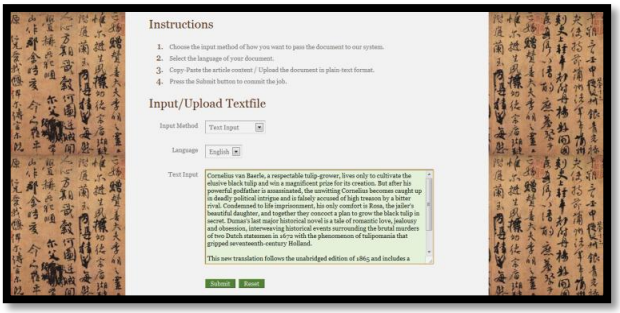

Figure 2. Input Screen-Shot

Then the system will output some URLs and snippets as the potential source of plagiarism. (see Figure 3.)

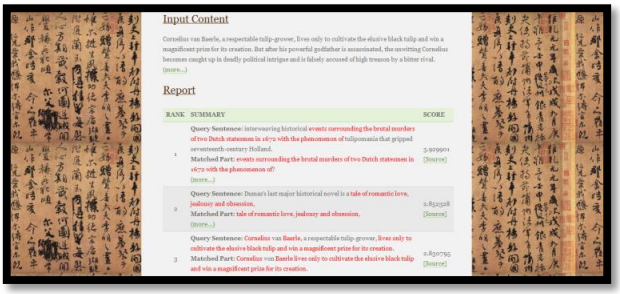

Figure 3. Output Screen-Shot

6. Conclusion

Comparing with other online plagiarism detection systems, ours exploit more sophisticated features by modeling how human beings plagiarize online sources. We have exploited sentence-level plagiarism detection on lexical, syntactic and semantic levels. Another noticeable fact is that our approach is almost language independent. Given a parser and a POS tagger of a language, our framework can be extended to support plagiarism detection for that language.

7. References

Salha Alzahrani, Naomie Salim, and Ajith Abraham, "Understanding Plagiarism Linguistic Patterns, Textual Features and Detection Methods " in IEEE Transactions on systems , man and cyberneticsPart C: Applications and reviews, 2011

D. M. Blei, A. Y. Ng, M. I. Jordan, and J. Lafferty. Latent dirichlet allocation. Journal of Machine Learning Research, 3:2003, 2003.

Bear F. Braumoeller and Brian J. Gaines. 2001. Actions Do Speak Louder Than Words: Deterring Plagiarism with the Use of Plagiarism-Detection Software. In Political Science & Politics, 34(4):835-839.

Sergey Brin, James Davis, and Hector Garcia-molina. 1995. Copy Detection Mechanisms for Digital Documents. In Proceedings of the ACM SIGMOD Annual Conference, 24(2):398-409.

Alberto Barrón Cedeño and Paolo Rosso. 2009. On Automatic Plagiarism Detection based on n-grams Comparison. In Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval, ECIR 2009, LNCS 5478:696-700, Springer-Verlag, and Berlin Heidelberg,

Jan Grman and Rudolf Ravas. 2011. Improved implementation for finding text similarities in large collections of data.In Proceedings of PAN 2011.

NamOh Kang, Alexander Gelbukh, and SangYong Han. 2006. PPChecker: Plagiarism Pattern Checker in Document Copy Detection. In Proceedings of TSD-2006, LNCS, 4188:661-667.

Yuhua Li, David McLean, Zuhair A. Bandar, James D. O' Shea, and Keeley Crockett. 2006. Sentence Similarity Based on Semantic Nets and Corpus Statistics. In Proceedings of the IEEE Transactions on Knowledge and Data Engineering, 18(8):1138-1150.

Yi-Ting Liu, Heng-Rui Zhang, Tai-Wei Chen, and Wei-Guang Teng. 2007. Extending Web Search for Online Plagiarism Detection. In Proceedings of the IEEE International Conference on Information Reuse and Integration, IRI 2007.

Caroline Lyon, Ruth Barrett, and James Malcolm. 2004. A Theoretical Basis to the Automated Detection of Copying Between Texts, and its Practical Implementation in the Ferret Plagiarism and Collusion Detector. In Proceedings of Plagiarism: Prevention, Practice and Policies 2004 Conference.

Sebastian Niezgoda and Thomas P. Way. 2006. SNITCH: A Software Tool for Detecting Cut and Paste Plagiarism. In Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, p.51-55.

Maria Soledad Pera and Yiu-kai Ng. 2010. IOS Press SimPaD: A Word-Similarity Sentence-Based Plagiarism Detection Tool on Web Documents. In Journal on Web Intelligence and Agent Systems, 9(1).

Xuan-Hieu Phan and Cam-Tu Nguyen. GibbsLDA++: A C/C++ implementation of latent Dirichlet allocation (LDA), 2007

Martin Potthast, Benno Stein, Alberto Barrón Cedeño, and Paolo Rosso. An Evaluation Framework for Plagiarism Detection. In 23rd International Conference on Computational Linguistics (COLING 10), August 2010. Association for Computational Linguistics.

Kenneth Sörensena and Marc Sevaux. 2005. Permutation Distance Measures for Memetic Algorithms with Population Management. In Proceedings of 6th Metaheuristics International Conference.

Efstathios Stamatatos, "Plagiarism Detection Based on Structural Information" in Proceedings of the 20th ACM international conference on Information and knowledge management, CIKM'11

Robert A. Wagner and Michael J. Fischer. 1975. The String-to-string correction problem. In Journal of the ACM, 21(1):168-173.

Daniel R. White and Mike S. Joy. 2004. Sentence-Based Natural Language Plagiarism Detection. In Journal on Educational Resources in Computing JERIC Homepage archive, 4(4).

Du Zou, Wei-jiang Long, and Zhang Ling. 2010. A Cluster-Based Plagiarism Detection Method. In Lab Report for PAN at CLEF 2010.