NLP APPLICATIONS IN EXTERNAL PLAGIARISM DETECTION

Sorin AVRAM[1], Dan CARAGEA[2], Theodor BORANGIU[3]

The purpose of our present research is the development of a plagiarism detector, integrating natural language processing tools with similarity measures and n-grams techniques. Our detection target included both verbatim plagiarism and slightly modified passages, in the same language; while the prototype is developed for English documents, the solution can be successfully adapted to other languages. Test results using the prototype over a corpus of documents presented high rates of precision and recall. The current research is in-line with the latest trends in paraphrasing recognition, including high levels of obfuscation, in the quest of uncovering all the forms of plagiarism.

Keywords: plagiarism detection, natural language processing, overlapping n-grams, sentence similarity

1. Introduction

In the last decades, plagiarism has become an epidemic phenomenon in academia, being more and more difficult to detect and withstand. The widely available access to texts on digital libraries and the Internet, promoted though opaque educational practices has led to an increased number of plagiarism cases, which can now happen across languages and have a high level of obfuscation. Different reports showed that the volume of publications has a doubling period for science of about 15 years, corresponding to an annual growth rate of 4.73% [1], which means that any manual detection process is a waste of resources.

Ministries and higher education institutions have formed and delegated different bodies and committees to render policies and procedures on plagiarism. As people can copy, translate and paraphrase any sources from the digital space, without mentioning its source, there's an obvious need for building an accurate automatic plagiarism detector.

In recent years, many research papers on plagiarism detection have been published, basically oriented on two directions: intrinsic and external plagiarism detection. Intrinsic plagiarism detection is based on style processing, detecting variations in text's readability, vocabulary richness, the average sentence length

[1] PhD student, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: avram.sorin@gmail.com

[2] Eng., The Executive Agency for Higher Education Research Development and Innovation Funding. e-mail: dan.caragea@uefiscdi.ro

[3] Prof., Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania

and the average word length [2]. External plagiarism detection has attracted more attention because of its close relation to information retrieval. Still, external plagiarism detection had the focus, because it employed confirmed IR techniques and proved to be significantly more reliable. The difficulty of the task has its source in the large number of comparisons with source documents and the obfuscation techniques, used to disguise the fraud.

In this paper we report a new approach in detecting external plagiarism, implementing and testing a prototype, based on lexical analysis tools and n-grams techniques. Despite many attempts to incorporate more sophisticated information into the models, the n-gram model remains the state of the art, used in virtually all speech processing systems [3] and offers the basis for any of the top Part-Of-Speech (POS) taggers [4].

The research's objective is to enhance the latest designs for detecting paraphrasing with the capacity of recognizing derived versions of the same word, while computing plagiarism likelihood. The advantage of this solution is that the effort for similarity computing remains the same, while the text processing can be done only once per document, in a totally isolated preprocessing stage. As a positive side-effect, this plug-in property of the design allows further integration with different similarity algorithms like bag-of-words, SCAM [5], YAP [6] etc.

The structure of the article is organized as follows: section 2 presents the design of the algorithm, section 3 evaluates the performance of the prototype and section 4 is the conclusion.

2. Prototype design

In this section, we describe the context and the methods used in plagiarism detection. There are three phases in our detection method such as preprocessing, identifying similar passages and postprocessing stage.

The context of the research is defined by the input data: a corpus containing scientific documents, written in English, saved in text files. At this stage, the research is only focused on improving the detection of same-language plagiarism, so no translation mechanisms or cross-language dictionaries are involved. Since the large majority of the well-recognized research is published in English, our aim is to use an English ontology tool for text and word processing.

As this research is mainly focused on maximizing detection performance in terms of precision and recall, and less oriented on the execution speed, we opted for high level programming language and a Java implementation of our prototype.

2.1 Preprocessing

The main objective of this phase is to cut through the word-level obfuscation. If paraphrasing cases involve rewriting techniques, we have also found that minor changes of the words can be a good way to disguise a

plagiarized text. In such cases, changing the tense of verb or the number of a noun can provide a very different word set for the same sentence, while the sense of the phrase is nearly identical with the original.

In order to prevent working with two different word sets and an inconsistent outcome of the detection phase while the inputs are basically the same, each word (possibly derived) from the two compared documents has to be reduced to its canonical form (lemma). During this phase, each text is processed, split into sentences and afterwards in words, then each word is then substituted with its lemma. In order to identify the suspect passages, the text has to be processed in three steps:

- sentence splitting;
- word tokenizing;
- word lemmatization.

A few tools for natural language processing are already available, capable to support different types of text processing and different programming languages. Two of the most appreciated and well-known tools in the field are the Stanford Core NLP and the Apache Open NLP; while the first is created by a group of researchers leaded by Prof. Chris Manning, from the famous Californian university [7], the second is an open-source initiative within the Apache Software Foundation [8]. In a more thorough evaluation, Ievgen Karlin [9] presents the differences between the two libraries, underlining the advantages and functionalities of Core NLP over the open-source alternative, as they are presented in table 1.

Table 1

Abilities of Open NLP and Core NLP [9]

Ability	Stanford Core NLP	Apache Open NLP
Sentence Detection	+	+
Token Detection	+	+
Lemmatization	+	-
Part-of-speech Tagging	+	+
Named Entity Recognition	+	+
Co-reference Resolution	+	-

As a second perspective, the lemmatizer offered by the Core NLP toolkit outputs 142,293 lemmas, also superior to the Open NLP dictionary [10]. Also, in terms of usability, Core NLP is available in different packages, for the most common programming languages: Java, Perl, Python and Ruby.

Having selected Stanford Core NLP as the tool for the preprocessing phase, the implementation followed the steps required for engine setup and running: using a dedicated *java properties* structure, Core NLP is loading the three *annotators*, which are the functional classes for text processing:

- *tokenize* - tokenizes the text;
- *ssplit* - splits a sequence of tokens into sentences;

- *pos – part of speech annotation*, labels tokens with their POS tag.

Table 2 describes the setup and processing steps, as all the text handling is done using Core NLP's optimized data structures.

Table 2

Pseudocode description of the preprocessing phase

```
Initialize CoreNLP properties_structure
//properties.put("annotators", "tokenize", " ssplit", " pos")  - "annotators" activation
Start StanfordCoreNLP engine
For each txt_ file
  While (SentenceAnnotation.hasMore())
    While (TokensAnnotation. hasMore())
        Return token.get(LemmaAnnotation.class).toLowerCase();
    End While// sentences are tokenized into words
  End While // text is split into sentences
Save ".ids" file //containing lemmatized text
End Foreach File
```

2.2 Identify similar passages

The detection of similar passages between two text documents can be done using different techniques, yet the objective of the present research is more focused on solutions capable of identifying obfuscation, like paraphrasing and summarization. Using the n-grams method ensures more flexibility, as reworded fragments could still be identified.

The n-grams method employs two steps for similarity detection:

- generate n-gram sets for each sentence;
- compute similarity (distance) between each pair of n-gram sets, originated from each of the two documents.

As n-grams generation is a highly used and well tested method, the issue of performance in translated in choosing the right parameters for gram's length.

As Alberto Barron-Cedeno and Paolo Rosso proved in an earlier study the tri-gram structure is found to be the most effective in this task. This method is recommended because the common n-grams between two documents are usually a low percentage of the total number of n-grams of both texts, as it's shown for four sample documents from the METER corpus, in table 3 [11].

Table 3

Common n-grams in different documents (avg. words per document: 3,700) [11]

Documents	1-grams	2-grams	3-grams	4-grams
2	0.1692	0.1125	0.0574	0.0312
3	0.0720	0.0302	0.0093	0.0027
4	0.0739	0.0166	0.0031	0.0004

Finalizing the tri-grams generation, all data is saved in vectors containing the number of occurrences of each gram generated, for each sentence, for each document, providing the input for the next step: distance calculation.

Computing the lexical similarity for each pair of sentences used one of the most popular metrics in text-mining: *the Cosine Similarity Index*, developed by Salton and MacGill in 1983 [12]. An important advantage of the *Cosine Index* over the alternative, *Jaccard Index*, is the lower impact of vector length, which in cases of text comparison can be a powerful factor. As Sternitzke and Bergmann proved in 2009 [13], *Jaccard Index* is highly influenced by the differences in size of the analyzed documents, showing similarity results with less than 25%, even when comparing subsets of same lexical lot. As it is defined (formula 1), the *Cosine Index* measures the similarity between two vectors of an inner product space (A_i and B_i), corresponding to the text documents d_1 and d_2:

$$similarity(d_1, d_2) = \frac{d_1 * d_2}{\| d_1 \| * \| d_2 \|} = \frac{\sum_{i=1}^{n} Ai * Bi}{\sqrt{\sum_{i=1}^{n} (A_i)^2} * \sqrt{\sum_{i=1}^{n} (B_i)^2}} \quad (1)$$

2.3 Postprocessing

In the postprocessing phase, we analyze the results for each of the pair sentences and save any matches between suspected and original documents.

For the final report, each pair of sentences that have at least three overlapping tri-grams and a similarity degree over the threshold of 0.25 is qualified as probable plagiarism cases. The threshold has been determined in series of tests using different text documents from *A Corpus of Plagiarized Short Answers (CPSA)* [14].

3. Performance assessment

Validating the results of our research involved the testing over a corpus of documents, available in text format, using only standard characters (ASCII) and all written in English. We adopted the CPSA, created by Paul Clough and Mark Stevenson from the University of Sheffield [14], which is a corpus for the development and evaluation of plagiarism detection systems. The corpus contains 19,599 words, available in 96 documents, from which 62% of the files are written by native English speakers and the remaining 36 (38%) by non-native speakers [14]. This particularity of the corpus was decisive, since our prototype is not designed for online translation or cross-language dictionaries integration.

Another important advantage of this option is related to the very diverse levels of obfuscation present in its documents; as the authors published, CPSA contains *near-copy* fragments, *light-revision* paragraphs and *heavy-revision*

passages, as well. This particularity allowed a thorough testing of the prototype and an optimization of its parameters, as well.

In the end we evaluated the precision and the recall of the exercise, obtaining the results presented in table 4:

Table 4

The evaluation result using CPSA corpus

Measures	Score
Precision	0.9456
Recall	0.9062

The most important result of the present research is the high recall rate: 90% of plagiarism cases were identified, only 10% having such an obfuscation degree, not to be detected. In Fig. 1, we can see a number of relevant cases from the detection report, for both low and high obfuscation.

Case 1 : Semantic similarity between phrase [1] of document 1 and phrase[1] of document 2 = 80.0%
Phrase [1] of document 1: "Bayes' theorem relates the conditional and marginal probabilities of two random events."
Phrase [1] of document 2: "In probability theory, Bayes' theorem (often called Bayes' law after Rev Thomas Bayes) relates the co nditional and marginal probabilities of two random events."

Case 2 : Semantic similarity between phrase [1] of document 1 and phrase[9] of document 2 = 70.0%
Phrase [1] of document 1: "Bayes' theorem relates the conditional and marginal probabilities of two random events."
Phrase [9] of document 2: "Bayes' theorem relates the conditional and marginal probabilities of events A and B, where B has a n on-vanishing probability: P(A|B) = frac{P(B | A), P(A)}{P(B)}."

Case 3 : Semantic similarity between phrase [2] of document 1 and phrase[4] of document 2 = 25.81%
Phrase [2] of document 1: "For example, a person may be seen to have certain medical symptoms; Bayes' theorem can then be us ed to compute the probability that, given that observation, the proposed diagnosis is the right one."
Phrase [4] of document 2: "Bayes' theorem can be used to compute the probability that a proposed diagnosis is correct, given th at observation."

Case 4 : Semantic similarity between phrase [3] of document 1 and phrase[11] of document 2 = 51.43%
Phrase [3] of document 1: "Bayes' theorem forms a relationship between the probabilities xcof events A and B. Intuitively, Bayes ' theorem in this form describes the way in which one's recognition of 'A' are updated by having observed 'B'."
Phrase [11] of document 2: "Intuitively, Bayes' theorem in this form describes the way in which one's beliefs about observing 'A' are updated by having observed 'B'."

Case 5 : Semantic similarity between phrase [4] of document 1 and phrase[9] of document 2 = 23.64%
Phrase [4] of document 1: "P(A | B) = P(B | A) P(A) / P(B), P(A|B) is the conditional probability of A given B. It is derived from or depends upon the specified value of B, therefore it is also known as the posterior probability."
Phrase [9] of document 2: "Bayes' theorem relates the conditional and marginal probabilities of events A and B, where B has a n on-vanishing probability: P(A|B) = frac{P(B | A), P(A)}{P(B)}."

Case 6 : Semantic similarity between phrase [5] of document 1 and phrase[10] of document 2 = 75.68%
Phrase [5] of document 1: "P(B|A) is the conditional probability of B given A. P(A) is the prior probability A. It doesn't take into account any information about B, so it is 'prior'."
Phrase [10] of document 2: "Each term in Bayes' theorem has a conventional name:
* P(A) is the prior probability or marginal probability of A. It is 'prior' in the sense that it does not take into account any infor mation about B.
* P(A|B) is the conditional probability of A, given B. It is also called the posterior probability because it is derived from or dep ends upon the specified value of B.
* P(B|A) is the conditional probability of B given A.
* P(B) is the prior or marginal probability of B, and acts as a normalizing constant."

Fig. 1. Sample from the detection report

The high precision of the result, also called true positives, is the fraction of retrieved instances from the total plagiarism cases available [15]. In this case, we consider that the algorithm is characterized by a high sensitivity, being able to detect most of the suspected cases (94.56%), while only 5.44% are incorrect. This level of performance comes with an obvious side effect, due to a very high number of computations in comparison with the alternative solutions (e.g. fingerprinting). Fig. 1 shows a sample of a detection report significant in this sense.

Based on the present result, we need to explore further in terms of plagiarism with different level of obfuscation and NLP resources. Plagiarism based on paraphrasing is still the subject of further reflections and developments.

4. Conclusion

Our current research represents a technological endeavor in plagiarism detection, beyond its primitive form, known as copy/paste. In many cases, plagiarism continues to exist, despite rewording or words insertions, which are so hard to identify just by using the traditional tools, based on fingerprinting.

The implemented prototype presented high efficiency, proving a high level of recall (90%) and a precision rate of nearly 94%.

Adopting this technological innovation could represent the solution for detecting two of the most common plagiarism methods: verbatim and low level paraphrasing. Furthermore, the opportunity of migrating this solution to Romanic or Neo-Latin languages is very high, due to the elevated number of inflected forms and the lack or miss-use of diacritics.

Acknowledgements

The design and implementation of this solution are the result of a previous study in plagiarism detection and information retrieval, supported by The Executive Agency for Higher Education, Research, Development and Innovation Funding (UEFISCDI), from Bucharest, Romania.

REFERENCES

[1] *Larsen, P. O., Von Ins, M.*, "The rate of growth in scientific publication and the decline in coverage provided by Science Citation Index", Scientometrics, 2010, vol. 84, no. 3, pp. 575–603

[2] *Meyer Zu Eissen, S., Stein, B.*, "Intrinsic Plagiarism Detection", Advances in Information Retrieval: Proceedings of the 28th European Conference on IR Research, 2006, pp. 565–569, Springer-Verlag

[3] *Brill, E., Florian, R., Henderson, J. C., Mangu, L.*, "Beyond n-grams: can linguistic sophistication improve language modeling?", Proceedings of the 17th International Conference on Computational linguistics, 1998, vol. 1, pp. 186-190

[4] *Ramisch, C.*, "N-gram models for language detection", 2008, UE Ingenierie des Langues et de la Parole

[5] *Shivakumar, N., Garcia-Molina, H.*, "SCAM: A Copy Detection Mechanism for Digital Documents", Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries, 1995, Austin, Texas

[6] *Wise, M.*,"YAP3: Improved detection of similarities in computer programs and other texts", Proceedings of 27th SCGCSE Technical Symposium, 1996, pp. 130-134, Philadephia

[7] *Stanford University, NLP Group*, "The Stanford Natural Language Processing Group", 2013, http://nlp.stanford.edu/people.shtml

[8] *The Apache Software Foundation, Apache Open NLP*, "Open NLP", 2010, http://opennlp.apache.org/index.html

[9] *Karlin, I.*, "An Evaluation of NLP Toolkits for Information Quality Assessment", 2012, PhD Thesis, Vaxjo : Linnaeus University

[10] Ryzko, D., Rybinski, H., Gawrysiak, P., Kryszkiewicz, M., "Emerging Inteligent Technologies in Industry", 2011, ISBN: 978-3-642-22731-8, Springer-Verlag

[11] *Barron-Cedeno, A., Rosso, P.*, "On Automatic Plagiarism Detection Based on n-Grams Comparison", Advances in Information Retrieval, 2009, vol. 5478, pp. 696-700, ISBN 978-3-642-00957-0, Toulouse : Springer-Verlag

[12] *Salton, G., Macgill. M.J.*, "Introduction to Modern Information Retrieval", 1983, New York : McGraw-Hill

[13] *Sternitzke, C., Bergmann, I.*, "Similarity measures for document mapping: A comparative study on the level of an individual scientist", Scientometrics, 2009, vol. 78, pp. 113–130

[14] *Clough, P., Stevenson, M.*, "Language Resources and Evaluation: Special Issue on Plagiarism and Authorship Analysis", Developing A Corpus of Plagiarised Short Answers, 2009, University of Sheffield, http://ir.shef.ac.uk/cloughie/resources/plagiarism_corpus.html

[15] *Potthast, M., Stein, B., Barron-Cedeno, A., Rosso, P.*, "An Evaluation Framework for Plagiarism Detection", Proceedings of the 23rd International Conference on Computational Linguistics, 2010, pp. 997-1005, ACM