A Concept for Plagiarism Detection Based on Compressed Bitmaps

Andreas Schmidt*†, Reinhold Becker‡, Daniel Kimmig†, Robert Senger* and Steffen Scholz†
* Department of Computer Science and Business Information Systems,
Karlsruhe University of Applied Sciences
Karlsruhe, Germany
Email: {andreas.schmidt, robert.senger}@hs-karlsruhe.de
† Institute for Applied Computer Science
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: {andreas.schmidt, daniel.kimmig, steffen.scholz}@kit.edu
‡ esentri AG, Ettlingen
Email: reinhold.becker@esentri.com

***Abstract*—In the last few years, several public persons in Germany have been convicted of inadequate scientific practice and or scientific misconduct in writing their dissertations. An examples is the former German Federal Minister of Defense Guttenberg, who had to resign as a consequence of this scandal. These events have led to an increased interest in methodologies to automatically detect plagiarism in documents. In today's digital society, however, the vast growth of available information makes this a challenging task. To address this situation, tools for the detection of plagiarism must be built up on highly efficient data structures and utilize very fast operations. In our approach, we propose the use of compressed bitmaps as a representation form. We introduce a new concept of plagiarism detection, which is based on mapping suspicious documents and potential source documents onto these compressed bitmaps. We will explain how the detection process can be accelerated.**

***Keywords*-Compressed bitmaps; plagiarism detection; visualization**

I. Introduction

The fully automated search for plagiarized sections in documents is gaining more and more importance. This search is a multi-stage process [1] with the initial point being a suspicious document that has to be examined as to whether its content is plagiarized. In a first preselection step, which is called *source retrieval*, a number of so-called "candidate documents" are extracted. As the number of these reference documents typically is very high (i.e., all documents in the WWW), the efficiency of this step is of high importance. The challenge is to extract a possibly small number of documents for further investigation, without neglecting potentially relevant documents (high recall). In a second step, the so-called *text alignment* procedure, the preselected candidate documents are examined for text fragments that also appear in the suspicious document. In this step, a mapping between text fragments in the suspicious and the candidate set is performed. After this mapping process, a final knowledge-based process is performed, by means of which overlapping text fragments are combined or deleted and visualized.

In practice, it is required to distinguish between two possible scenarios. In the first case, the algorithm has access to a complete data pool. For example, this is the case in a well-defined research area where all relevant literature is stored in a local database. In this case, the algorithm is in full control of the complete procedure of plagiarism search. This scenario is not very likely. In the second more likely case, the first step depends on the utilization of an internet search engine, such as Google or Yahoo!.

The structure of the paper is as follows: In the next section we give a brief overview of the state of the art in plagiarism detection. Then in Section III we give an introduction of Zipf's Law and the size of typical vocabularies. After that, we present our approach (section IV) and in Section V we provide an algorithm for this approach. We conclude our paper with a number of tasks we plan for the future.

II. State of the Art

Since 2009, a competition has been organized in the context of the "Conference and Labs of the Evaluation Forum" (CLEF). The Plagiarism Analysis, Authorship Identification, and Near Duplicate Detection) (PAN) competition uses a standardized collection to compare the different approaches. In 2013, 32 teams joined the competition, in which a number of plagiarized documents was to detected by the developed software systems. In the following paragraphs, we focus on some of the different approaches according to the different tasks of a Plagiarism Detection System (PDS):

A. Source Retrieval

In source retrieval the following steps can be distinguished [2]: In order to generate the search request, the document under evaluation will be chopped into a number of paragraphs (chunking). For example, this fragmentation can be based on chapters or a defined number of lines or sentences. Often, dynamic chunking techniques are utilized. For example, intrinsic methods / procedures which search the document for conspicuities, such as lexis or average word length, in order

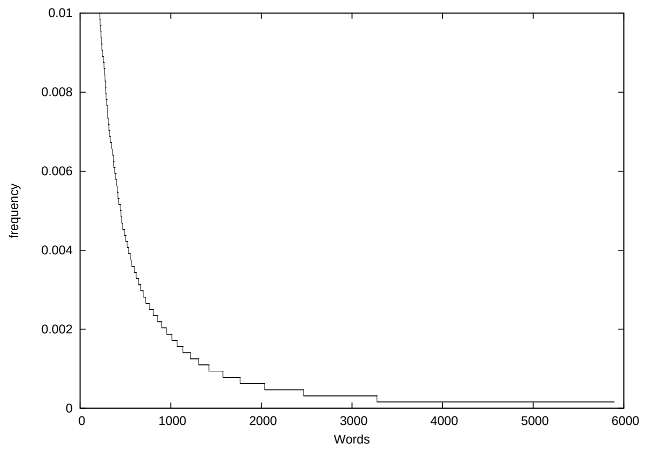

Figure 1. Zipf distribution of the words in all papers of the DBKDA 2013 conference

to find variations leading to potential plagiarized fragments in the document. After completing the first step, the chunks will be used to create keywords or key phrases which then will be implemented in the request for the search engine. This step is typically carried out by i.e. observation of the $td{*}idf$ values [3] for single words within the chunks or by the generation of one or more so-called n-grams (phrases with a typical length of 5 to 8 words). These keywords or key phrases will then be submitted to a search engine. Based on the response from the search engine, a number of documents will be downloaded and analyzed in the following step.

B. Text Alignment

The main purpose of the text alignment step is to identify text fragments from the source data / documents that found their way into the document under investigation. Not only 1:1 plagiarized text fragments should be identified, the objective is to identify disguised or faked fragments as well. These faked phrases are often characterized by simply changing the sequence of words, adding or deleting single words or combining sentences.

C. Post Processing and Final Visualization

After this mapping process, a final knowledge-based process is carried out in which overlapping text fragments are combined or deleted and accordingly visualized.

III. Words, Sentences, and Language

A. Zipf's Law

Zipf's law [4] postulates that the frequency of any word in a language is inversely proportional to its rank in the corresponding frequency table. Figure 1 shows the distribution of the words that appeared in the papers of the DBKDA-2013 conference [5]. It can be seen that, the distribution is represented by a hyperbolic function.

Examining the articles of the DBKDA 2013 conference, we made the following findings:

- Altogether, about 7000 different (stemmed) words are used.
- The average size of an article is about 5000 words.
- A typical article only contains about 1000 different (stemmed) words.
- The average length of a sentence was between 14 and 22 words for the different articles.
- A paragraph comprised between 7 and 14 sentences for the different articles on the average.

These values were not specific of the conference selected, but can be reproduced with other scientific publications as well. The average length of a sentence in one of the famous literature books *Moby Dick* or *The Adventures of Tom Sawyer* was only about 7 words.

Based on the finding of Zipf's law, it can be postulated that there are words that are more important to identify a document than other words. For example, when we used the following six keywords "the, of, and, in, to, for" in a google query, we got about 11.5 billion results. When searching for the three words "bitmap, index, encoding", only 1.4 million results were obtained. Because a lot of words are inflections or derived from a base form, a stemming process in general reduces all the words to their (morphological) stem. This process reduces the number of different words, without losing the meaning when used in an information retrieval process. The porter stemmer [6] is the most well-known stemmer for the English language.

B. Size of Vocabulary

The vocabulary of a native English speaker is about 10,000 to 12,000 words. This is quite small compared to the size of the entire English vocabulary which is between 500,000 to 600,000 words (twice the size of the French vocabulary which has only about 300,000 words). This does not include technical terminology. The chemical nomenclature terminology, for example, contains at least 20 million words [7].

IV. Chosen Approach

A. Representation of Text

In this subsection, we examine how the basic units of a document (word, n-gram, sentence, paragraph, whole document) can be represented to support the work of a PDS. The task of a PDS is to find overlapping sections between the suspicious document and the candidate documents. In the trivial case of a 1:1 copy & paste plagiarism, n-grams (a sequence of n words) can be used, which slide over the document base to find matching regions (technically, this is typically implemented with an inverted index). This approach works very well if the plagiarized extracts are not too obfuscated by changes in the syntax or exchange of words (synonyms, hypernyms).

Another approach is to use the well-known vector space model (VSM). In this model, the whole document or a part of it is represented as an n-dimensional vector. The vector space is defined by the used vocabulary (words) in the document base. The similarity between two documents (or parts of it) is defined by a similarity function based on the vector representation. A typical similarity function, for example, is the cosine similarity between two vectors. Often, VSM is combined with the $td{*}idf$ weighting.

A major difference between the n-grams representation and the VSM approach is that in VSM the words are represented as an unordered collection. This makes this approach more robust against syntactic modifications of the text. A serious drawback of this approach is that when using a single vector for one document, small plagiarized fragments can easily be overlooked [8]. This can be compensated by building vectors from smaller fragments of the document, which, on the other hand, makes the computation more expensive, because more vectors have to be compared.

B. *Compressed Bitmaps*

In our approach, text fragments are represented as compressed bitmaps. Every bit in the bitmap represents a fixed word of our language (i.e., all words sorted alphabetically). Setting the bit at position i to 1 means, that the i^{th} word appears in the given text fragment. As in the case of the VSM, we maintain an unordered list of words ("Bag of Words") for each text fragment. But, in contrast to VSM, we do not keep the information of how often a word appears in text fragment, we only keep the information whether it appears or not. This seems to be a substantial loss of information at first, but for small units of text (like sentences) the difference is not so big, because in general, most of the words (and especially the relevant words) would not appear more than once anyway. The advantage of our approach is the very fast computation of similarity between two bitmaps using the Jaccard similarity coefficient [3] (equation below).

$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$.

In our case, the sets A and B represent the words in two text fragments. The Jaccard measure is then defined as the coefficient between the number of elements in the intersection of the two sets and the number of elements in the union of the sets. If both sets are equal (i.e., they contain the same words), the value is 1, in the case of no common words the coefficient is 0. In our implementation using bitmaps, the operation consists of an `and` and an `or` operation between two bitmaps and an integer division.

When the chunks of text are small (i.e., sentence or paragraph size), the amount of 1-bits is quite small and the bitmap can be compressed very effectively using Run Lengh Encoding (RLE) [9]. The required operation, namely, the computation of the Jaccard similarity coefficient can also be performed on the compressed bitmaps, even with higher speed. One important question is how many words the vocabularity should contain? Our experiments with the DBKDA conference proceedings suggest, that probably 10,000 words should be enough. But on the other hand, the whole English vocabulary contains about 500,000 words and the technological terminology can be even much bigger. So in a next experiment, we compare the memory consumption between uncompressed and compressed bitmaps and vector representations. In this experiment, we use different vocabularity sizes starting from 10,000 words up to 5,000,000 words as well as different sizes of text fragments, ranging from 15 words (a typical sentence) to 200 words (a paragraph) up to 5000 words (average paper size). The size of the vector (sparse vector implementation) was calculated by the number of used dimensions in the vector, multiplied by the size of two integers ($2 * 16$ bit). Figure 2 shows the results of this experiment.

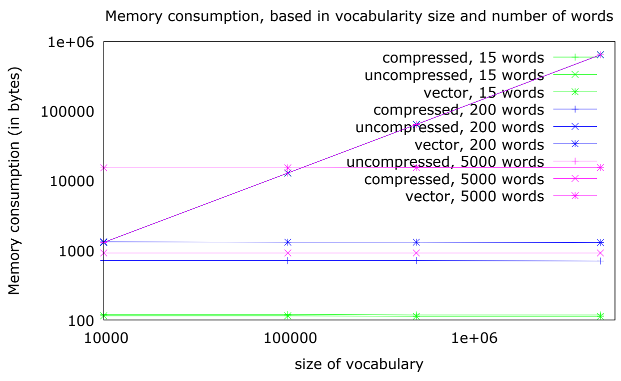

Figure 2. Memory consumption for different representation forms

The most relevant findings of this experiment are:

- The memory consumption for the uncompressed bitmap grows linearly with the size of the vocabulary.
- The memory consumption for the compressed bitmaps and the sparse vector are independent of the vocabulary.
- The size of the compressed bitmaps and the vector depends on the length of the text fragment.
- For the sentence case (15 words), vector and compressed bitmap have nearly the same memory consumption (vector 6% less). For the 200-word paragraph, the bitmap only needs 53% of the memory of the vector representation. For the full paper (5000 words) case, the compressed bitmap memory consumption is only about 6% of the vector representation.

C. *Influence of Word Ordering in Bitmap*

The position of the words in the bitmap has an influence on the size of the compressed bitmaps. Figure 3 shows two different ordering schemes. In the first (sentence with about 15 words) and third bitmaps (paragraph with 100 words), the words are ordered alphabetically, which results in a nearly equal distribution (the vertical lines represent 1-bits). In the second and fourth bitmaps, by contrast, the order is based on the Zipf distribution shown in Figure 1. Words with a very high frequency appear at the beginning of the bitmap and the words with low frequency appear at the end of the bitmap.

The ordering based on the Zipf distribution has two advantages:

1) The gaps between two 1-bits are very small at the beginning of the bitmap, but grow towards the right. Based on the characteristics of the Word Aligned Hybrid (WAH) algorithm, which needs at least a gap size of a multiple of 31 (or 63), this leads to a number of fill words at the beginning of the bitmap, followed by 0-fills with growing capacity, interrupted by single literals. This results in a better compression ratio as when the set bits are more evenly distributed over the whole bitmap. Figure 4 compares the memory consumption for the different sorting orders. On the

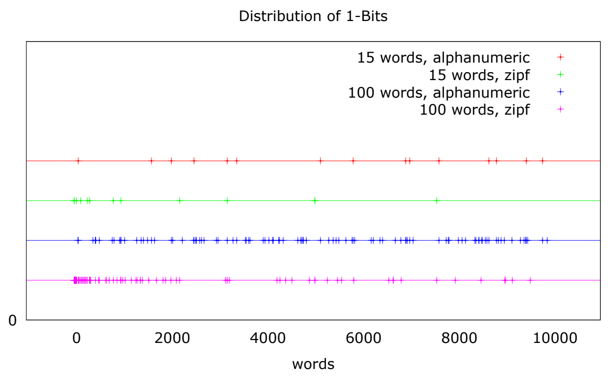

Figure 3. Comparison of distributions using alphabetic or word frequency-based ordering of words in bitmap

sentence (15 words) and paragraph (200 words) level, the amount of memory for the Zipf-based sorting is about 75% compared to alphabetic ordering. The gain of memory for the 5000 words case is smaller (about 2%).

2) As discussed before, words with high frequency are not very usable to identify documents. As these words are now grouped together at the beginning of the bitmap, they can simply be ignored by skipping the first n bits.

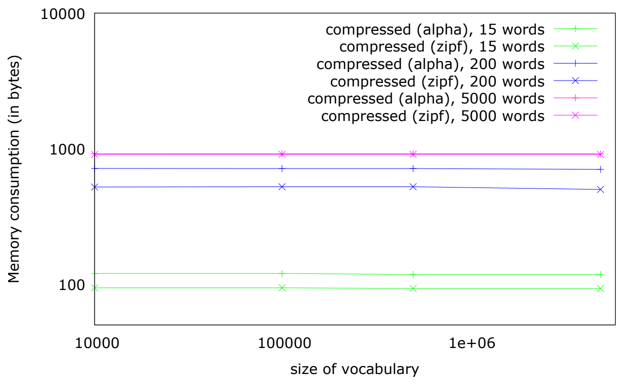

Figure 4. Comparison of distribution using alphabetic or word frequency based ordering of words in bitmap

D. Comparison of Execution Time for Measuring Similarity

In a last experiment, we compare the time for the computation of the similarity measure using different measure functions and text representations. Figure 5 shows the difference in execution time between the cosine measure based on the vector representation and the Jaccard measure based on compressed and uncompressed bitmaps. The main findings of this experiment are:

- The computation of the similarity measure using compressed bitmaps and the Jaccard coefficient is much more independent of the chunk size (sentence, paragraph, whole paper) compared to the cosine similarity measure. The time difference using the bitmaps is in a range of a factor of 2 compared to three orders of magnitude for using the vectors with the cosine similarity measure.
- The performance of the cosine similarity is slightly better (max. factor of 2) for the sentence case. In the case of considering a whole paragraph or paper, the bitmaps solution is at least superior by a factor of 4, growing up to a factor of 50.

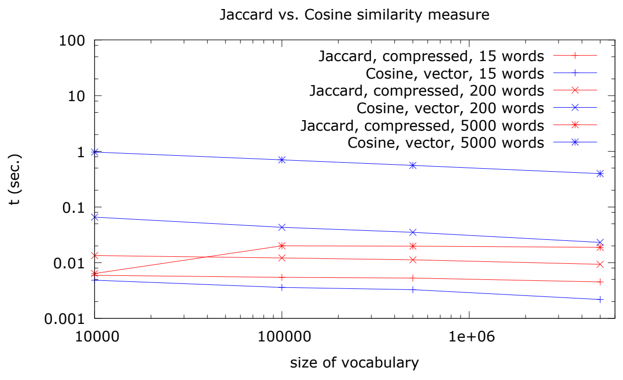

Figure 5. Comparison of execution time for similarity measure based on compressed bitmaps (Jaccard) and vector (cosine measure)

Based on the findings of the last two experiments, the usage of bitmaps seems an appropriate choice for the *source retrieval* step, to find the “candidate set”. In contrast to the much more expensive cosine measure which typically only allows to use one vector per document, it is possible to subdivide a document into a number of smaller parts which can be examined separately and, hence, to minimize the chance to overlook smaller parts, which have been plagiarized.

But also for the *text alignment* step, compressed bitmaps seem to be suitable. The reason for this is, that compared to the search for n-grams which require that the order of the words is the same in the suspicious document as well as in the candidate documents, the order of the word is irrelevant. This makes this approach insensitive to obfuscation approaches like paraphrasing single sentences. Additionally, the obfuscation approach by replacing single words by synonyms or hypernyms can be handled easily. In this case, not only the bit for a concrete word has to be set, but also for possible synonyms and hypernyms. These words can be provided automatically using Wordnet [10].

V. Algorithm

The algorithm for the identification of the candidate set is as follows: In a preprocessing step, all documents which form the comparison document set are fragmented into a small number of chunks. For each of these text-fragments the bitmap representation is built. Additionally, the amount of 1-bits (the number of words) is stored. To handle obfuscations, taxonomies from Wordnet are used and for every word where Wordnet offers a synonym or one or more hypernyms, the bits for these words are also set (the number of words determined previously is not incremented). After this enrichment step, the bitmaps are compressed using the WAH algorithm [11]). Parallel to the fragmentation of the document into a small number of text fragments for the candidate search, a sentence-wise

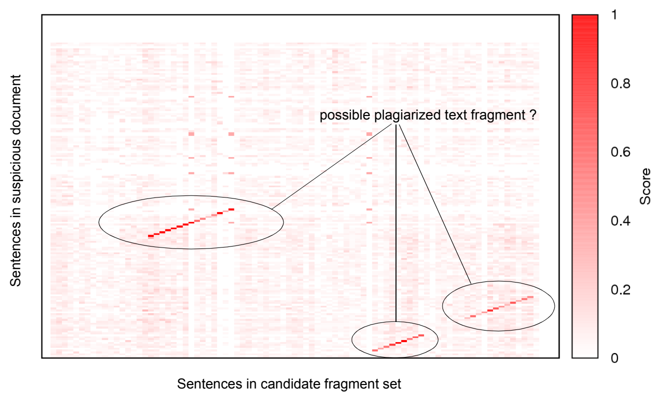

Figure 6. Plagiat Matrix

fragmentation and transformation into compressed bitmaps is performed for the later text alignment step. In the same way, but without the enrichment of the Wordnet taxonomies, the suspicious document is preprocessed.

In the next step, every compressed bitmap of the suspicious document is compared with every bitmap of the document base using the Jaccard measure coefficient. As we are only interested in finding the candidate set, the most leftmost bits (which represent the most irrelevant words) can be ignored. If the Jaccard measure is above a threshold value, the fragment the bitmap belongs to is included into the candidate set.

In the text alignment step, the bitmaps representing the sentences are considered. Every compressed bitmap representing a sentence in the suspicious document is compared using the Jaccard measure with all sentences from the fragments identified in the first step. As a result, we get a matrix, where each row represents a sentence in the suspicous document and each column represents a sentence from the qualified fragments. Every cell in the matrix has a value between zero and one, representing the similarity between two sentences. This matrix can easily be represented graphically using a heatmap, as it is shown in Figure 6. Using a color gradient for the values in the interval $[0, 1]$ from white to red, we can easily identify fragments with similar or alike content. The lines originate from a number of consecutive sentences with high similarity and, therefore, are probably plagiarisms. The representation of plagiarized fragments as lines is also shown in [12].

VI. Conclusion

We presented a new approach to plagiarism detection using compressed bitmaps. As we have shown, the bitmap approach can be used for the candidate retrieval as well as for the text alignment process. At the beginning of the paper, we show that from the memory consumption aspect and the performance aspect, a compressed bitmap with the Jaccard measure is superior to the vector representation using cosine similarity measure. To cover both steps, we build compressed bitmaps based on different aggregation levels. An additional enrichment step using semantic taxonomies from Wordnet allows us to also cover obfuscation techniques like renaming words. Obfuscation by paraphrasing is also covered by our approach, based on the set characteristic (no order of words). The final visual representation using heatmaps shows plagiarized text fragments as lines.

As a next step, we have to finely tune our process, find appropriate threshold values, and compare our results with others (see PAN competition mentioned in Section II). Another interesting aspect for our future research is the parallelization of the whole process using a framework like Hadoop [13].

References

[1] B. Stein, S. M. zu Eissen, and M. Potthast, “Strategies for retrieving plagiarized documents,” in Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ser. SIGIR ’07. New York, NY, USA: ACM, 2007, pp. 825–826.

[2] M. Potthast, M. Hagen, T. Gollub, M. Tippmann, J. Kiesel, P. Rosso, E. Stamatatos, and B. Stein, “Overview of the 5th international competition on plagiarism detection,” in CLEF 2013 Evaluation Labs and Workshop Working Notes Papers, 2013.

[3] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. New York, NY, USA: Cambridge University Press, 2008.

[4] G. Zipf, Human behavior and the principle of least effort: an introduction to human ecology. Addison-Wesley Press, 1949. [Online]. Available: http://books.google.de/books?id=1tx9AAAAIAAJ

[5] IARIA, “ThinkMind // DBKDA 2013, The Fifth International Conference on Advances in Databases, Knowledge, and Data Applications,” 2013, [accessed 24-Feb-2014]. [Online]. Available: http://www.thinkmind.org/index.php?view=instance&instance=DBKDA+2013

[6] M. F. Porter, “Readings in information retrieval,” K. Sparck Jones and P. Willett, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, ch. An Algorithm for Suffix Stripping, pp. 313–316.

[7] Wikipedia, “Wortschatz — Wikipedia, the free encyclopedia,” 2013, [accessed 24-Feb-2014]. [Online]. Available: http://de.wikipedia.org/wiki/Wortschatz

[8] N. Meuschke and B. Gipp, “State of the Art in Detecting Academic Plagiarism,” International Journal for Educational Integrity, vol. 9, no. 1, Jun. 2013, pp. 50–71.

[9] M. Nelson, The Data Compression Book. New York, NY, USA: Henry Holt and Co., Inc., 1991.

[10] G. A. Miller, “Wordnet: A lexical database for english,” Communications of the ACM, vol. 38, 1995, pp. 39–41.

[11] K. Wu, E. J. Otoo, and A. Shoshani, “Optimizing bitmap indices with efficient compression,” ACM Trans. Database Syst., vol. 31, no. 1, 2006, pp. 1–38.

[12] T. Gottron, “External plagiarism detection based on standard ir technology and fast recognition of common subsequences - lab report for pan at clef 2010.” in CLEF (Notebook Papers/LABs/Workshops), M. Braschler, D. Harman, and E. Pianta, Eds., 2010.

[13] T. White, Hadoop: The Definitive Guide, 1st ed. O’Reilly Media, Inc., 2009.