CitePlag: A Citation-based Plagiarism Detection System Prototype

Norman Meuschke
UC Berkeley, USA
meuschke@berkeley.edu

Bela Gipp
UC Berkeley, USA / OvGU, Germany
gipp@berkeley.edu

Corinna Breitinger
UC Berkeley, USA
breitinger@berkeley.edu

ABSTRACT

This paper presents an open-source prototype of a citation-based plagiarism detection system called CitePlag. The underlying idea of the system is to evaluate the citations of academic documents as language independent markers to detect plagiarism. CitePlag uses three different detection algorithms that analyze the citation sequence of academic documents for similar patterns that may indicate unduly used foreign text or ideas. The algorithms consider multiple citation-related factors such as proximity and order of citations within the text, or their probability of co-occurrence in order to compute document similarity scores. We present technical details of CitePlag's detection algorithms and the acquisition of test data from the PubMed Central Open Access Subset. Future advancement of the prototype lies in increasing the reference database by enabling the system to process more document and citation formats. Improving CitePlag's detection algorithms and scoring functions to reduce the number of false positives is another major goal. Eventually, we plan to integrate text-based detection algorithms in addition to the citation-based detection algorithms within CitePlag.

Keywords

Plagiarism Detection System, Prototype, Open Source, Detection Algorithms, Citation Analysis, CbPD

1. INTRODUCTION

Studies show that existing plagiarism detection systems (PDS) can accurately identify copies or modestly obfuscated plagiarism. Strongly reworded paraphrases, translations, and idea plagiarism, however, lack the lexical text similarities that existing PDS require to discover plagiarism. Therefore, current PDS always almost fail to detect these plagiarism forms [16, 17, 18, 25, 26, 27, 36].

For academic texts, citation pattern analysis allows us to assess document similarity independent of lexical text matches. We define citation patterns as sequences of citations in two texts, A and B, which partially or entirely link to shared references of A and B. We use the term "citation" for referring to strings in the body of academic texts that link to sources in the bibliography and "references" for denoting entries in the bibliography. Figure 1 shows the concept of citation pattern analysis.

Citation pattern analysis evaluates the number of shared citations, their order of appearance, their proximity to each other in the text, and their probability of co-occurrence for computing a document similarity score. We approximate the co-occurrence probability of citations based on their usage frequencies within the collection [14].

Plagiarism detection (PD) is one possible application of citation pattern analysis. The approach can also serve other information retrieval tasks, such as recommending related literature. We use the term citation-based plagiarism detection (CbPD) for distinguishing the application of citation pattern analysis for PD purposes.

5th International Plagiarism Conference '12,
July 16–18, 2012, Newcastle upon Tyne, UK.

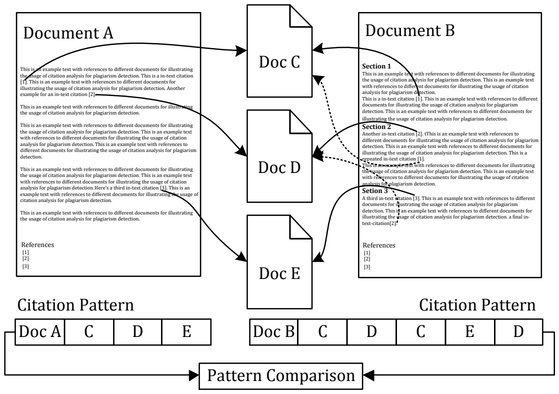

Figure 1: Concept of citation pattern analysis

To allow for a CbPD analysis, the suspicious documents and the documents in the reference collection must possess the required input data for a citation pattern analysis. The requirements on input data are twofold.

First, the documents must contain citations within their accessible full text, because, knowing the position of citations within the text is crucial for identifying citation patterns.

Second, accurate citation information must be obtainable for identifying matching citations between documents. If this data is not readily accessible, tools able to extract citation information from the original documents must be applicable to the collection. This second requirement is of importance because citation extraction tools can only interpret a fraction of the many style conventions that exist for citing sources in academic texts. Depending on the citation style of an academic field, the data to include in citation and reference strings, as well as its order, and formatting varies. Tools for automated citation and reference extraction are either knowledge- or rule-based or employ machine learning [7]. To achieve good recognition performance, a tool's system of rules or its training set must be tailored to specific citation styles.

In prior studies, we demonstrated that the CbPD concept is a valuable enhancement, not a substitution, to existing text-based PD approaches [13, 15]. CbPD can partially identify forms of plagiarism that existing PDS cannot detect. For example, CbPD identified 13 of 16 translated plagiarisms in the dissertation of Mr. Guttenberg, Germany's former minister of defense [15].

The setups of these earlier studies were limited and did not entirely reflect real-world PD scenarios. In [15], we conducted the citation-based plagiarism check subsequent to the manual verification of plagiarism in Guttenberg's thesis and limited the reference collection to proven sources of plagiarism. Furthermore, the uncommon citation formats of the analyzed documents, which originate from the legal domain, prevented a fully automated citation extraction and required manual rework. In [13], we artificially plagiarized documents and inserted them into the test collection. The goal of our current research is a large-scale evaluation of our citation-based PDS prototype.

The requirements for a CbPD analysis restrain us from using existing collections for testing CitePlag and comparing it to text-based PDS. Although test collections containing verified cases of plagiarism exist from prior PDS comparisons, e.g. [4, 23, 35], none of these collections offers academic full texts that include citations and references. Given the unsuitability of existing test collections, we chose the PubMed Central Open Access Subset (PMC OAS) for implementing a prototype that can analyze a large, real-world document collection.

We will not present the individual similarities we detected in the PMC OAS in this paper, because we are convinced no suspicion of potential academic misconduct should be expressed prior to a thorough expert investigation. Only domain experts can fully comprehend and judge the scientific contribution of certain works, especially, since the understanding of what constitutes undue publication behavior and/or self-plagiarism varies between academic fields [1, 3, 5]. Experts often require many weeks, even months, to properly analyze plagiarism allegations. The recent investigations into more than 20 alleged plagiarism offenses, mainly involving German politicians, made clear this immense time commitment [33].

During plagiarism investigations, affected authors must have the opportunity to make their opinion heard, since suspicious similarities between documents sometimes occur without any wrongdoing of authors. We construct a fictitious, yet possible scenario to illustrate this necessity. Imagine an accidental omission of an author name on a journal article written by several authors. Parts of the journal article may be legitimately similar to a previous conference article of the omitted author. A PDS may flag the journal article as being suspiciously similar to the conference article. The accidentally omitted author name may cause a reviewer to suspect plagiarism. Giving authors the opportunity to comment on findings before making public any allegations is thus a simple way to prevent undue and unwarranted damage to the reputation of authors.

We are not experts in the life sciences and are therefore more likely to fail in appropriately classifying detected similarities in the PMC OAS. Furthermore, we are unable to perform careful investigations, especially because we compare all ~234,000 documents the PMC OAS to all other documents in the corpus.

Any automated PD procedure is subject to an error rate that can cause false positives. For CbPD, the main sources of error are the extraction and disambiguation of citation and reference data from within the text. Currently, CitePlag is a PDS prototype. Major functionalities of CitePlag, such as the document import and the detection process (see section 3) are the focus of our current research and subject to continuous enhance- and improvement. Therefore, we advise users of the system to treat its detection results with an appropriate skepticism.

This paper is a research-in-progress report rather than a final description and evaluation of CitePlag. We demonstrated the capabilities of CbPD using the Guttenberg plagiarism case verified by experts prior to the current study [15]. This paper presents the current state of the prototype, which we make freely available under an open-source license. Please consult the CbPD project website at *www.sciplore.org* for latest technical details about CitePlag, source code download, and related information. By offering CitePlag as an open-source system, we invite everyone who is interested, to analyze the PMC OAS and draw his or her own conclusions from the indicated similarities.

The structure of this report is as follows. Section 2 introduces the PubMed Central Open Access Subset. Section 3 presents technical details of the CitePlag prototype, while section 4 outlines future research concerning the CbPD approach and the CitePlag prototype.

2. PUBMED CENTRAL OA SUBSET

The PMC OAS comprises ~234,000 Open Access full-text articles from the life sciences. Citation and reference data for articles is available in XML format, which simplifies data extraction.

The U.S. National Center for Biotechnology Information (NCBI), a subunit of the U.S. National Library of Medicine (NLM), provides the PMC OAS together with numerous other information systems. Figure 2 presents an overview of systems and data sources and their relation to the PMC OAS. This overview will clarify some technical terminology used in section 3.

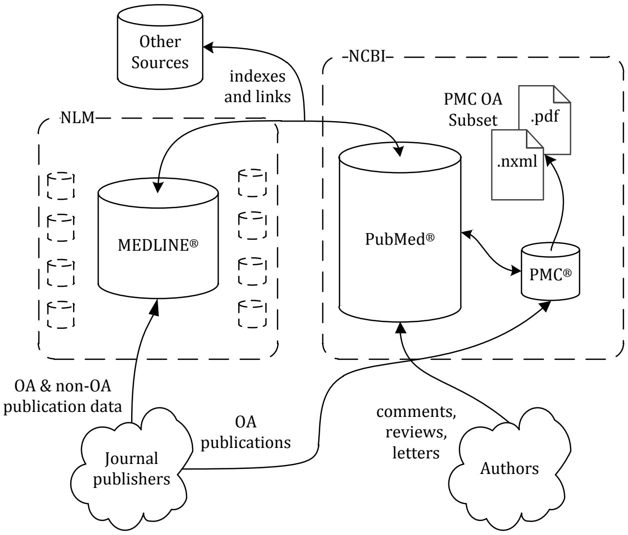

Figure 2: Information systems related to the PMC OAS

MEDLINE® is a comprehensive index of ~19M articles from ~5,600 peer-reviewed journals in the life sciences. MEDLINE does not offer full texts, but it provides structured records of bibliographic data. PubMed® is an information system that offers free access to MEDLINE records, and additional content such as articles outside of MEDLINE's scope. PubMed comprises ~21M records for which it lists outgoing links to full text sources if available. Most records in PubMed Central® (PMC) are also part of MEDLINE and/or PubMed. PMC offers additional content that does not necessarily meet the formal criteria for being included in MEDLINE or PubMed, such as editorial letters, comments, book reviews, conference summaries and non-journal manuscripts.

The NLM and NCBI assign system-wide unique, numerical IDs to records upon inclusion in MEDLINE, PubMed, or PMC. We will refer to these IDs as MEDID for MEDLINE, PMID for PubMed and PMCID for PMC entry keys.

2.1.1 *The PMC OAS in Prior PD Studies*

In an earlier study, researchers from the Garner Lab analyzed a subset of ~72,000 articles from the PMC OAS using their self-developed, text-based similarity algorithm eTBLAST [28].

eTBLAST is an adaption of biomedical sequence alignment algorithms to text [21]. The Garner Lab group could not identify cases of plagiarism in texts originating from the PMC OAS.

In comparable examinations, the same research group used eTBLAST to check abstracts of research articles contained in MEDLINE records for suspicious similarities [10, 11, 19]. During these studies, the team manually acquired and checked 4,515 full texts for articles with highly similar abstracts. Investigating these articles yielded 252 documents with a full text similarity score above 50% and non-shared author sets, which made them likely candidates for plagiarism. An additional 89 cases with the same similarity criteria had common authors, making them likely self-plagiarisms. The Garner Lab published the results of all studies in a database called Déjà vu [8]. Data in Déjà vu is openly accessible via a web front-end and is available for bulk download.

The Garner Lab team stated three possible reasons for the apparent absence of plagiarism in the PMC OAS. First, the researchers only manually checked 34 highly similar full-text article pairs. Second, the team analyzed less full texts from the PMC OAS than MEDLINE abstracts. Third, the group found that plagiarized works appear more often in journals with low impact factors, possibly because the plagiarists intended to reduce the risk of discovery by publishing in less popular journals. However, the PMC OAS covers mostly high-impact journals [28].

We hypothesize that plagiarism is more strongly obfuscated in high impact journals like those covered by the PMC OAS to avoid detection. Furthermore, we assume that such strongly disguised cases of plagiarism are much harder to detect than, for example, plagiarism committed by students. If these assumptions prove true, this provides another reason the Garner Lab PDS could not detect any plagiarized articles. Therefore, analyzing the PMC OAS with a citation-based PD approach appears promising to test whether CbPD can identify strongly obfuscated plagiarism.

2.1.2 PMC OAS Document Format

For including articles in PubMed Central, the NCBI requires content providers to submit texts in Extensible Markup Language (XML) and in conformance with a Document Type Definition (DTD) called the Journal Archiving and Interchange Tag Suite (JAITS) [31]. We refer to JAITS-conformant documents as NXML-texts because they carry the file extension .nxml. Content providers, mainly professional publishers, can additionally include articles as a PDF.

We expect the obligation of content providers to supply JAITS-conformant texts to increase the accuracy of citation and reference data. Publishers commonly impose predefined formatting rules on documents to be included in their volumes. We assume that most publishers providing articles to PMC use automated tools for the conversion of texts to the NXML format. It is likely that publishers incorporate this knowledge about specific formatting conventions into knowledge- or rule-based citation extraction tools. The precision and recall of such customized citation extraction tools are likely to be higher than for unspecialized tools.

The JAITS DTD provides markup for most document data that is necessary for a CbPD analysis. Metadata required for CbPD, such as: authors, title, publication dates, identifiers (IDs) and the partitioning of documents into sections, subsections and paragraphs is marked-up in NXML-texts. The DTD also enforces markup and linkage of citations and references in the NXML-texts. The availability of IDs facilitates the identification of shared references between articles.

The JAITS DTD does not provide markup for sentence and word boundaries. However, our CbPD algorithms require this information to analyze how many characters, words or sentences separate individual citations in the text for computing a similarity score. Section 3.1 describes details about how the CitePlag prototype acquires this information from NXML-texts.

In summary, the major advantages that made us choose the PMC OAS as a collection for demonstrating CitePlag are:

- the inclusion of high quality journal articles;
- the XML markup of most information we require;
- the accuracy of citation and reference data;
- the wide availability of IDs for documents and references;
- the partial analysis by a prior text-based PD study.

3. CITEPLAG PROTOTYPE

CitePlag is an Open Source prototype of a citation-based PDS developed in Java. We provide CitePlag for download at *www.sciplore.org*.

Figure 3 illustrates CitePlag's system architecture, which consists of four components - the parser, the database, the detector, and the report generator. The parser extracts bibliographic data, such as citations, references, authors, and titles from documents and stores it in the database. The relational database provides this document data to the detector. The detector runs the analysis algorithms and feeds the results back to the database for storage. The report generator retrieves detection results from the database and summarizes them for human inspection. The following subsections present the four main components in more detail.

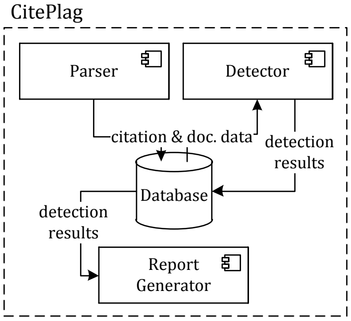

Figure 3: CitePlag system architecture

3.1 Document Parser

The document parser's task is to extract all necessary meta-, citation- and reference data from input documents and import it into the database. We fitted the parser of the current prototype to the PMC OAS. Therefore, the current version of CitePlag available for download is only able to process NXML-texts. In the future, we plan to replace the prototypic parser with a component that uses the open-source citation extraction tool ParsCit [6]. The future version of the parsers will be able to process more citation styles and document formats, e.g. PDF files.

A major subtask in parsing is determining the exact positions of citations within the document's full text. The detection process of similar citation patterns requires knowing the exact position of citations in the text. We measure this position in terms of the character, word, sentence, paragraph, and section count where citations appear. The parser applies standard Java text processing methods for acquiring character counts and evaluates the corresponding tags in the NXML-texts for obtaining the paragraph and section position of citations. NXML-texts do not provide markup for sentences and words. Hence, identifying the boundaries of these elements requires pre-processing prior to data extraction.

The goal of the pre-processing step is to include delimiters for sentences and words without compromising the existing XML markup. For performing this task, we developed an independent sub-component to the parser, the Sentence-Word-Tagger (SW-Tagger). After the SW-Tagger has identified sentence and word boundaries, a second sub-component, the Data Parser, extracts all relevant data and imports it into the database. Figure 4 illustrates the two-stage parsing process of the CitePlag document parser. The following two subsections describe the SW-Tagger and the Data Parser in more detail.

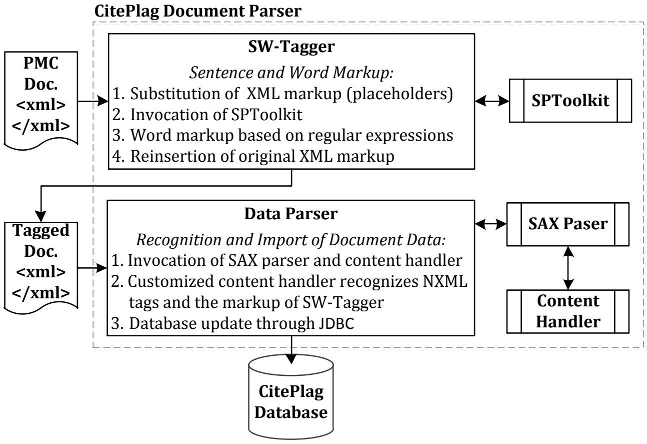
CitePlag Document Parser
PMC Doc. <xml> </xml>
SW-Tagger
Sentence and Word Markup:
1. Substitution of XML markup (placeholders)
2. Invocation of SPToolkit
3. Word markup based on regular expressions
4. Reinsertion of original XML markup
SPToolkit
Tagged Doc. <xml> </xml>
Data Parser
Recognition and Import of Document Data:
1. Invocation of SAX parser and content handler
2. Customized content handler recognizes NXML tags and the markup of SW-Tagger
3. Database update through JDBC
SAX Paser
Content Handler
CitePlag Database

Figure 4: Two-stage parsing process for NXML-documents

3.1.1 Sentence-Word-Tagger (SW-Tagger)

SW-Tagger identifies individual sentences and words in NXML-texts and marks them with characteristic delimiters that do not impair the validity of the original XML markup. The ambiguity of natural language renders the recognition of sentences, words, and other parts of speech (POS) challenging. An example of a highly ambiguous grapheme in natural languages is the period. Besides indicating the end of a sentence, a period can, for example, be a decimal point or a delimiter within an email address. Specific terms used in the life sciences pose additional challenges to sentence and word boundary detection. Articles in this field frequently refer to chemical substances, abbreviations, or other domain-specific named entities that are difficult to match to ordinary sentence structures.

We chose to incorporate an existing sentence tagger into the CitePlag document parser. Approaches to POS tagging comprise knowledge-based systems, heuristics, or machine learning [34]. The peculiarities of life science texts force researchers to adjust POS taggers specifically for this field, in order to achieve a good POS detection performance.

Several POS taggers exist for the life sciences, e.g. [2, 9, 22, 30]. We evaluated OpenNLP [29] in combination with the biomedical extensions of [2], StanfordCoreNLP [30], and SPToolkit [22] regarding their suitability for integration into the CitePlag document parser. We manually inspected five annotated documents for each tool. Although our test was too small to be statistically significant, results seemed to reproduce the precision and recall values of ~99% for word and sentence boundary detection that earlier studies reported [2, 22]. OpenNLP and StanfordCoreNLP required ~1.5s, and SPToolkit required ~30ms of processing time per document in our tests. We attribute this difference in runtime complexity to the different approaches of the systems. While OpenNLP and StanfordCoreNLP employ sophisticated machine learning procedures, SPToolkit relies on comparably less complicated heuristic rule sets.

Aside from its superior runtime performance, the sentence detector of SPToolkit offers an output format that is easier to integrate with the other sub-components of the document parser than that of OpenNLP or StanfordCoreNLP. The two later tools can process XML texts. However, both tools discard the original XML markup and create individually formatted output files. This tagging behavior would require changes to the tools' source codes for producing an output that includes sentence and word markup in addition to the original XML tags. SPToolkit provides its output as a plain Java string object that is universally usable.

We decided to incorporate SPToolkit into the document parser, because the tested tools have practically identical sentence detection performance, yet SPToolkit offers both better runtime performance and a favorable output format. By default, SPToolkit is not able to process XML texts. Therefore, we substitute all XML tags in the original documents with unique placeholder strings of the form `Z\*§`*running no.*`/§` and store the tag content in an index for later re-insertion. After the substitution, the parser runs the sentence detection procedures of SPToolkit.

SPToolkit misses the functionality of word boundary detection. To avoid using a runtime-intensive POS tagger based on machine learning, we adapted and incorporated word markup heuristics that are common in word split-up tools. We designed the tagger to markup words with plain text annotations similar to the ones we use for tagging sentences, so they do not interfere with the original XML markup. The tagger restores the original markup after the detection of sentences and words by re-substituting the placeholder strings with the original tag content from the stored index.

To check the quality of the markup procedure, we randomly sampled four documents from four journals and inspected the markup for three paragraphs in each document. For words, we found 2,092 correctly identified instances, six incorrect separations and no misses. Five of the six errors originated from one document that states the names of places and tribes in native African languages. These words contain unusual combinations of diacritics and hyphens that caused the word split-up heuristics to fail. The word markup procedure achieved a precision of 99% and a recall of 100%. The detection for sentences was error-free in our sample. Overall, we are confident that the implemented markup procedure works sufficiently accurate.

3.1.2 Data Parser

The data parser extracts all information necessary for a CbPD analysis from NXML-texts. This task requires evaluating the original XML markup and the plain text markup for sentences and words that the SW-Tagger introduced to the documents during the pre-processing step. The parser must process all documents in their entirety, because it must read and extract data from all parts of the corresponding texts. For example, documents generally list metadata, such as author names and journals, at the beginning of the text. Citation information occurs throughout the text, while references occur at the end of the text.

The given extraction task requires sequential read-only parsing of ~234,000 documents. We implemented the Data Parser according to the Simple API for XML (SAX) [24], because the functionality of

SAX meets the task’s requirements, while offering high processing speed. The Java programming language offers several frameworks for XML processing besides SAX, for example, the Java API for XML Processing (JAXP) or the Streaming API for XML (StAX). SAX is the most basic, because opposed to the other two frameworks it imposes strictly sequential reading of documents without interruption and does not offer functionality for manipulating documents. The functional restrictions allow SAX to be very efficient in the use of computational resources, which results in a high processing speed [32, p. 36].

SAX follows a push approach for accessing data in XML documents. This means a parser implementing the SAX API reads and triggers (“pushes”) a notification when it detects one of five predefined events. Encountering the start or end tag of the whole document or arbitrary elements represents one event each, thus totaling four events. The fifth event is the encountering of literal character data. Only the application that invokes the SAX parser defines reactions for events that the SAX parser reports. For this purpose, the invoking application must provide callback handlers to the SAX parser. These handlers contain and execute programming logic dependent on the event they receive from the SAX parser.

The content handler (see Figure 4) is the callback handler of the data parser that extracts document metadata, citations, and references. For most data elements, such as document IDs, author names, and references, this extraction is straightforward. Likewise, citations are easy to parse when the respective NXML text contains individual tags for every citation.

However, some texts state several citations in an abbreviated fashion, for example, “[3 – 8]” without offering XML markup for all citations in the range. To recognize these notations, we implemented an additional check to see if citations occur within a range of 13 or less characters. We chose thirteen characters by assuming that a notation similar to this: “[110] – [115]” is the likely maximum length of an abbreviated citation range. If citations occur within the 13-character-interval, the content handler uses regular expressions to check whether the literal character data between the citation tags actually represents a citation range.

For keeping track of sentence and word counts, we adapted the method of the callback handler that reacts to event notifications for literal character data. We use regular expressions to recognize the sentence and word markup introduced in the pre-processing step. After gathering all data for an element, such as a citation or reference, the content handler submits the element to the database.

We limited the documents in the PMC OAS to articles that have the document types “research-article”, “review-article”, “case-report”, “other”, “brief-report” and “report”. We also exclude documents containing more than one text body or no text body at all. Samples indicated that documents without a text body are mostly scanned versions of older articles that only express metadata in NXML. Documents with multiple text body parts were usually conference reviews that list summaries of proceeding articles. Both of these document types are not relevant for a plagiarism analysis. The exclusions affected ~13,000 documents. The total number of documents imported to the CitePlag database was 221,220.

3.2 Database

We chose the Open Source Software MySQL for managing CitePlag’s database. Figure 5 depicts CitePlag's data model in a special Entity Relationship Model (ERM) notation. This ERM variant states the data type of attributes in capital letters after their name. A diamond shape in front of attribute names indicates the permissibility of null values for that attribute. If the diamond is unfilled, the attribute can hold null values. A light blue fill for normal attributes and a red fill for foreign key attributes indicate that null values are prohibited. Connectors symbolize relationships and link to those attributes that participate in the relationship from a technical perspective. The diagram omits relationship names due to its technical nature.

Entities are documents, authors, citations, references, matching citation patterns (“CitPatMatches”) and the citations that form the patterns (“CitPatMembers”). The partition of entities into tables and the relationships between those tables follow common database design practices. Most table and attribute names are self-descriptive. We will explain names that may not be as intuitive.

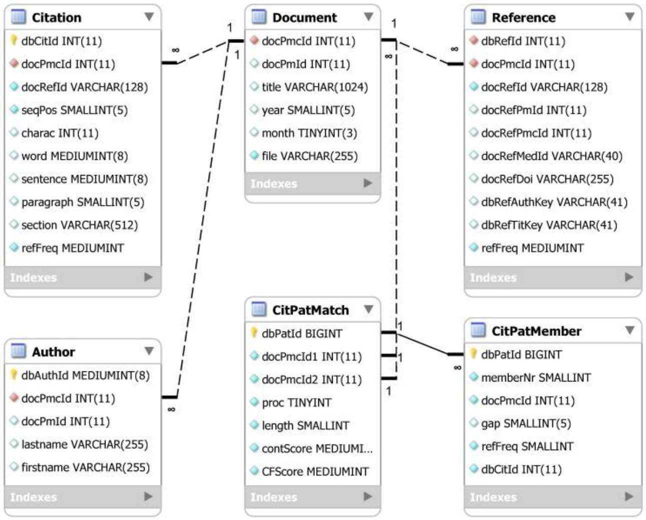

Figure 5: ER data model for the CitePlag Database

Attributes with the prefix “db” represent CitePlag-internal IDs that we assigned, while attributes with the prefix “doc” are IDs contained in the original NXML texts. The attributes “dbRefAuthKey” and “dbRefTitKey” in the “Reference” table are keys that we created artificially, based on the author names and the title of references. We use these keys for approximate reference matching, in case the document does not state other IDs such as a PMCID or DOI for the specific reference.

The “proc” attribute in the “CitPatMatch” table identifies the pattern analysis procedure (see section 3.3.1) that identified the match. The attribute “length” in the same table states the number of citations that are part of the match. The attributes “contScore” and “CFScore” are similarity factors for ranking citation patterns according to their degree of suspicion (see section 3.3.2).

3.2.1 Consolidation of Reference Identifiers

Documents in the PMC OAS commonly state different document identifiers such as PMIDs, MEDIDs or DOIs for references. However, different authors do not use identifiers consistently. For example, some authors state no identifiers, some use a PMID, others prefer a DOI or vice versa.

The best possible identification of matching references is a prerequisite for a CbPD analysis. For this purpose, we consolidated available document identifiers for references after we imported the respective data into the CitePlag database. Our aim was to assign all identifiers that are available for a document in the corpus to all references that likely point to the respective document. To achieve this, we had to identify valid relationships between identifiers and documents.

By visually examining examples from the dataset, we verified that all document identifiers available for references are subject to a certain error rate. Examples we found encompassed references with PMIDs that matched neither the simultaneously given DOIs nor the documents in general. Human errors or OCR recognition errors are likely causes for these kinds of erroneous assignments.

For completing or correcting reference records that did not state a certain identifier or likely stated an incorrect one, we applied the following procedure.

First, we individually selected all PMIDs, DOIs, MEDIDs and RefTitKeys.

Second, we took each of these identifiers as a seed for building all combinations with other identifiers that appear in two or more documents. For example, when we took PMIDs as the seed, we selected all pairwise combinations of PMID-DOI, PMID-MEDID, PMID-RefAuthKey and PMID-RefTitKey that authors had stated for actual references in at least two different documents. If we encountered non-unique combinations of identifiers, we chose the combination that most authors used and ignored all others. By assuming that the most commonly used combination of e.g. a PMID and a DOI is likely to be the correct mapping, we consolidated all ambiguous pairwise mappings of document identifiers to unique mappings. During this process, we recognized that RefAuthKey is to error-prone for using it as a seed.

Third, we joined the consolidated pairwise-unique mappings of document identifiers using the respective seed identifier in the mappings as the join criterion. This step yielded the following four combined mappings for the respective seed identifiers.

1 PMID-DOI-MEDID-RefAuthKey-RefTitKey

2 DOI-PMID-MEDID-RefAuthKey-RefTitKey

3 MEDID-PMID-DOI-RefAuthKey-RefTitKey

4 RefTitKey-PMID-DOI MEDID-RefAuthKey

Fourth, we joined the mappings 1 through 4 consecutively in this order to the table of all references using the respective seed identifier of the mappings as the join criterion. If reference records matched one of the mappings in at least one more identifier besides the seed identifier, which we used for the join, we updated all data fields of the reference record to equal the mapping. Mapping 4, which uses the artificially computed RefTitKey as the seed identifier, is potentially more error-prone than the other mappings. Therefore, we used mapping 4 only to alter records that do not offer any other document identifier.

Table 1 displays the availability of document identifiers for references before and after the consolidation. The table states the number of references for which the respective type of document identifier is available. Most frequently, authors state PMIDs when citing sources, DOIs and MEDIDs follow in second and third rank. In the table, we count the reference quantities per identifier category according to the most popular identifier that individual references offer. If a reference states a PMID and a DOI for example, we count it for the PMID category only. The table also lists the number of distinct identifiers from each group in the corpus.

During the consolidation, we could assign a PMID to ~100,000 reference records that did not have one prior to the consolidation. We were able to reduce the number of references that do not offer any numeric identifier by ~58,000 records. Additionally, we reduced the number of distinct PMIDs by ~3,000 and the number of distinct DOIs by ~17,000. These decreases in distinct identifiers suggest that we were able to eliminate the respective quantities of non-unique identifiers.

	Before Consolidation		**After Consolidation**	
	No. of Ref.	**No. of dist. IDs**	**No. of Ref.**	**No. of dist. IDs**
Total	6,921,249			
PMID	5,470,266	2,367,554	5,572,531	2,364,433
no PMID, DOI	195,359	158,652	192,705	141,357
no PMID, no DOI, MEDID	84	81	82	79
No identifiers, authors and title	831,899	655,841	733,183	597,220
No title and/or authors	423,641	-	422,748	-

Table 1: Reference identifier consolidation

3.3 Detector

The detection component performs pairwise comparisons between citation sequences. The component applies three different pattern analysis algorithms, which we developed to cover common forms of plagiarism, and which we will explain in the next sub-section.

3.3.1 Citation Pattern Analysis Algorithms

In [14], we proposed three pattern analysis algorithms for citation sequences termed Longest Common Citation Sequence, Citation Tiling and Citation Chunking. This section will provide a summary.

The *Longest Common Citation Sequence* (LCCS) is an adaption of a traditional similarity measure for text strings. The LCCS consists of the maximum number of citations that one can take from a citation sequence without changing their order, but allows skipping over non-matching citations. For instance, the sequence (3, 4, 5) is a subsequence of (2, 3, 1, 4, 6, 8, 5, 9). The LCCS measure recognizes the order of citations, but offers flexibility to cope with slight transpositions or gaps of non-matching citations.

Measuring the LCCS yields high similarity scores if a plagiarist uses longer parts of another text without alterations or only with minor changes of the source's citations. These patterns characterize copy&paste plagiarism that potentially underwent minor obfuscations such as rewording through synonym replacements.

Greedy Citation Tiling (GCT) is also an adaption of a well-known similarity measure for text strings, which its inventor specifically designed for PD purposes [37]. GCT identifies the longest individual patterns of consecutive, matching citations. The algorithm permanently links individual longest matches in the compared citation sequences and stores them as a so-called tile.

GCT focuses on exact matches in the citation sequences. Such matches are strong indicators for potentially suspicious text similarity. GCT is able to deal with transpositions in the citation sequence that result from rearranging longer text segments, which is typical in shake&paste plagiarism.

Citation Chunking is a set of heuristic procedures that identify local citation patterns regardless of potential transpositions, i.e. rearranging citations, or scaling, i.e. using identical citations multiple times, although the source only stated them once. We define three strategies for delimiting citation chunks. The first strategy considers only consecutive matching citations for forming chunks. The second includes matching citations in a chunk if $n \leq 1$ or $1 > n \leq s$ non-matching citations separate it from the last preceding matching citation. The variable s denotes the number of citations in the chunk under construction. The third strategy includes citations that occur within a user-defined interval of text.

We believe that the first chunking strategy is likely to reproduce and detect citation patterns that result from copy&paste plagiarism or weak paraphrases, which are achieved, for example, through synonym replacements. The second chunking strategy discovers shake&paste plagiarism, which results from interweaving text segments from different sources. When analyzing longer bodies of text, the third chunking strategy may detect idea plagiarism. After any of the procedures have delimited citation chunks, they compare all chunks pairwise while neglecting the order of citations within the chunks. The number of matching citations is the main similarity criterion.

Identifying citation patterns is the first of two subtasks in the citation-based similarity assessment. The second is to rank patterns according to their likelihood of resulting from undue text usage. We determined two main factors that increase this likelihood and derived two corresponding ranking functions to analyze them. We describe these ranking functions in the next section.

3.3.2 Scoring Functions for Citation Patterns

Citing Frequency-Score (CF-Score)

We regard the citation counts of individual documents in the collection to be valuable for indicating potentially suspicious citation patterns. Intuitively, two documents A and B that both received, for example, two hundred citations, are more likely to appear in a matching citation pattern than two documents C and D that received three citations each, for example. Therefore, we consider citation patterns containing highly cited documents to be less likely a result of undue practices, but rather represent commonly cited standard literature of a field.

If a document is already highly cited, its likelihood of gathering additional citations increases. Merton analyzed this phenomenon and termed it the Matthew effect in science [20]. Over time, highly cited documents tend to form a body of standard literature in a field. Authors frequently cite standard literature when providing context or referring to a base of established knowledge, which is relevant to their own research. Therefore, standard literature commonly does not indicate a specific similarity in the content of two works, but rather a rough topical relatedness of the research in several works. On the contrary, we regard shared citations to rarely cited sources to be a comparatively stronger indicator for potentially suspicious similarities between two works.

To derive a scoring function from this hypothesis, we make the simplified assumption that authors cite sources independently of each other. That is, the choice to cite one source does not affect the choice of citing another. This assumption does not accurately reflect real citing behavior, because topical similarity of sources, their academic quality and other factors can influence an author's choice of document citations, hence making citations statistically dependent. Incorporating these complex and interrelated factors into a model is difficult and beyond the scope of our preliminary evaluation. Therefore, we assume independent citations to derive a simplified, and easy to work with approximation of co-occurrence probability for our current prototype. We plan to devise more complex and realistic ranking models in the future.

Assuming statistical independence for references, the probability of a reference r pointing to a document X equals the count of all references to X in the corpus divided by the corpus size N as follows: $P(r_X) = \frac{|r_X|}{N}$. Because rarely cited documents are more predictive and should receive a higher score, we inverse the ratio of the probability assessment to equal $\frac{N}{|r_X|}$. We expect that the value of more frequently cited sources in predicting uncommon, highly specific content similarities does not decrease in direct proportion to the number of citations these sources gather. Therefore, we consider the square root of the total number of references to a source $\sqrt{|r_X|}$ to be the denominator for our score.

Because we derive our score from analyzing **c**iting **f**requencies, we name it CF-score. The CF-score for a citation c_i that links to a reference r_j, which represents the source document X, computes as $CF\big(c_i(r_j)\big) = \frac{N}{\sqrt{|r_X|}}$.

To compute a CF-score for a citation pattern p_k that consists of n citations $c_1 \dots c_n$ that link to m references r_j we accumulate the CF-scores of all citations in the pattern: $CF(p_k) = \sum_1^n CF(c_i(r_j))$. Analogously, we compute the CF-score for a pair of documents d_1, d_2 that share q matching citation patterns p_k by accumulating the CF-scores of the matching patterns: $CF(d_1, d_2) = \sum_1^q CF(p_k)$.

To exemplify the computation of CF-scores for ranking citation patterns, we assume a corpus of $1{,}000$ documents. In this corpus four documents A, B, C and D have the following citation counts: $|r_A| = 100, |r_B| = 50, |r_C| = 10, |r_D| = 5$. Furthermore, we imagine two document pairs X, Y and X, Z that share the following citation patterns: X,Y: (A,B) (A,C) and X,Z: (CD). The resulting CF-scores for the document pairs compute as:

$$\begin{aligned} CF(X,Y) &= CF(p_1(A,B)) + CF(p_2(A,C)) + CF(p_3(B,B)) \\ &= \left(\frac{1{,}000}{\sqrt{100}} + \frac{1{,}000}{\sqrt{50}}\right) + \left(\frac{1{,}000}{\sqrt{100}} + \frac{1{,}000}{\sqrt{10}}\right) = 657.65 \end{aligned}$$

$$\mathrm{CF(X,Z)} = \mathrm{P}(p_1(\mathrm{C,D})) = \left(\frac{1{,}000}{\sqrt{10}} + \frac{1{,}000}{\sqrt{5}}\right) = 763.44$$

The example shows that although the document pair X, Y shares more citation patterns, the single pattern that document X shares with document Z scores higher because it consists of comparably rarely cited sources.

Continuity-Score (Cont.-Score)

The number and proximity of matching citations within shared citation patterns are major factors that determine the similarity and the degree of suspicion for individual patterns. Our previous analysis of real-world plagiarism cases, such as the one of Mr. Guttenberg [15], consistently confirmed this relationship.

To incorporate this knowledge as part of the CbPD analysis, we developed a scheme for computing what we call a continuity score for citation patterns. Based on our experience from prior plagiarism investigations, we devised the following scoring heuristic that weighs matching citations higher, if they occur in close proximity.

Within a citation pattern, each matching citation that follows another matching citation, after three or less intermediate, non-matching citations have occurred, should increase the continuity-score of the pattern. The score increase in this case should be larger than 1 for not reflecting a simple count of matching citations, which we record separately. Furthermore, the score increase should be larger if fewer non-matching citations separate two subsequent matching citations. Lastly, the score should increase in proportion to the number of previous matching citations that fulfill the criterion of having a maximum of three intermediate, non-matching citations separating them from the preceding matching citation. This characteristic of the function reflects our observation that the similarity of citation patterns progressively increases, if longer sequences of matching citations occur in the pattern. The following formula presents the formal definition of the continuity-score:

$$Cont.-Score = \sum_{p=1}^{n} \max\{a - 0.25 \cdot \left(p(c_M^i) - p(c_M^{i-1}) - 1\right), 1\}$$

$$a = \left\{ \begin{array}{c|c} 1 & c_M^1 \\ a+1 & p(c_M^i) - p(c_M^{i-1}) \leq 4\,, i > 1 \\ 1 & p(c_M^i) - p(c_M^{i-1}) > 4\,, i > 1 \end{array} \right\}$$

The formula considers a base score a for matching citations. For the first matching citation in the pattern c_M^1, we set the base score to 1. We increment the base score for each subsequent matching citation $c_M^i \mid i > 1$ in the pattern if less than four non-matching citations separate c_M^i from the previous matching citation c_M^{i-1}. We express this condition in terms of the sequential position $p(c)$ of citations. To penalize non-matching citations between matching citations, we subtract a penalty value of 0.25 for each non-matching citation. If four or more non-matching citations separate two matching citations, the base score is set to 1 again. In this case, the summand would become 0 if four intermitted non-matching citations separate the matching citations or negative if more than four non-matching citations exist in between. We chose to disallow the possibility that the continuity score of a pattern can become less than the count of matching citations in the pattern. We achieve this behavior through the application of the $max()$ operator, which ensures that the minimum score increase for each matching citation is 1.

Figure 6 exemplifies the computation of continuity scores for two citation patterns. In the figure, Arabic numerals represent matching citations and the letter x symbolizes non-matching citations. In the example, both citation patterns contain eight matching citations. This comparatively high number of matching citations allows both patterns to receive a continuity score that exceeds the length of the pattern, which equals the count of matching citations. The continuity score of the second pattern equals about 1.7 times the score of the first pattern. In this example, the higher score could signal that the second pattern is more likely to be a comparatively long match, and hence suspicious. The first pattern is somewhat likely to represent three smaller matches, which is less suspicious.

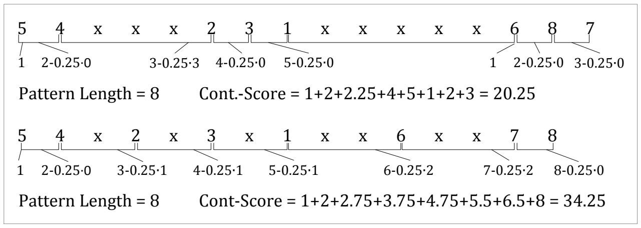
5 4 x x x 2 3 1 x x x x x 6 8 7
1
2-0.25·0
3-0.25·3
4-0.25·0
5-0.25·0
1
2-0.25·0
3-0.25·0
Pattern Length = 8
Cont.-Score = 1+2+2.25+4+5+1+2+3 = 20.25
5 4 x 2 x 3 x 1 x x 6 x x 7 8
1
2-0.25·0
3-0.25·1
4-0.25·1
5-0.25·1
6-0.25·2
7-0.25·2
8-0.25·0
Pattern Length = 8
Cont-Score = 1+2+2.75+3.75+4.75+5.5+6.5+8 = 34.25

Figure 6: Cont.-Score computation for citation patterns

3.3.3 Detector Implementation

Figure 7 depicts the main components of the detector using a class diagram notation of the Unified Modelling Language (UML). We implemented each pattern analysis algorithm as a stand-alone Java class. The class "CitationPatternChecker" is a central hub that instantiates the different analysis classes according to selectable parameters and bundles functionality, which all pattern analysis algorithms require, e.g. determining the set of shared references. The other classes are multi-threaded implementations for subtasks related to input and output operations on the CitePlag database.

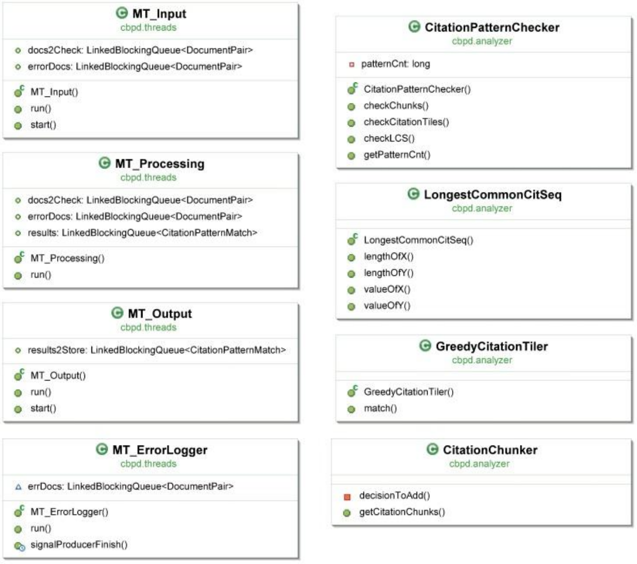

Figure 7: UML class diagram for CitePlag Detector

3.4 Report Generator

Currently, CitePlag only has basic report functionalities due to its prototypical nature. The CitePlag report generator retrieves detection results from the CitePlag database and creates plain text files for every document pair that has matching citation patterns above a user entered threshold. Figure 8 shows an example report.

```
Doc1. 59651, Brown: Debate: "How low should LDL
cholesterol be lowered for optimum prevention of
vascular disease?" Viewpoint: "Below 100 mg/dl"

Doc2. 524501, Al Shaer, Choueiri, Suleiman: The pivotal
role of cholesterol absorption inhibitors in the
management of dyslipidemia


Match 1, l= 5

D1, Sec. 1, Sent. 6: [B9] _(0)_ [B10] _(0)_ [B11] _(0)_
[B12] _(0)_ [B13]

D2, Sec. 1, Sent. 9: [B4] _(0)_ [B5] _(0)_ [B6] _(0)_
[B7] _(0)_ [B8]


Match 2, l= 5

D1, Sec. 1, Sent. 6: [B9] _(0)_ [B10] _(0)_ [B11] _(0)_
[B12] _(0)_ [B13]

D2, Sec. 5, Sent. 53: [B4] _(0)_ [B5] _(0)_ [B6] _(0)_
[B7] _(0)_ [B8]
```

Figure 8: Example results report

Result reports contain the main document metadata (title, authors, PMCID) for both documents and all matching citation patterns. For every matching citation pattern, the report states the length l of the match, the section ("Sec.") and sentence ("Sent.") of each document in which the match begins, and the citations that make up the match. The report lists matching citations enclosed in square brackets using the citation ID of the original NXML text. Additionally, the report states the number of non-matching citations that separate matching citations within a citation pattern enclosed in round brackets.

The report generator includes the overall score for each document pair as the first characters of the file name for the respective report file. This naming convention allows us to use the operating system's alphabetical file sort for ranking files according to their importance.

4. OUTLOOK

This paper describes the current state of the CitePlag prototype. We are still in the process of improving and advancing CitePlag's functionalities. We publish changes and updates on the project's website *www.sciplore.org*.

The focus of our current research and development is CitePlag's document parser and detection module. Regarding the document parser, we are working on integrating the open-source citation extraction tool ParsCit. Because ParsCit does not offer sentence and word detection, we must add this functionality to the tool to make it suitable for our extraction task. Once the new document parser is in operation, CitePlag will be able to process PDF files and recognize numerous citation formats. The improved document parser will enable us to significantly increase CitePlag's database, beyond its current coverage of the PMC OAS.

Improvements of CitePlag's detection module include a suitable combination of detection algorithms and consideration of further similarity factors for the scoring of detected citation patterns. Through further empirical research on verified plagiarism cases, we seek to discover more characteristics of citation patterns that can help to detect suspicious document segments. We will use these characteristics for constructing a comprehensive citation-based similarity model.

We are also working on devising additional scoring functions for ranking citation patterns according to their degree of suspicion. This is necessary to prevent false positives. Currently, CitePlag considers scores for citing frequency and continuity of citation patterns (see section 3.3.2). In the future, we will change the scoring function for citing frequency to employ Co-Citation Proximity Analysis (CPA) [12]. CPA is an enhancement of the popular Co-Citation similarity measure. Co-Citation considers documents as similar if documents that are more recent cite them together. CPA additionally evaluates the distance between the citations in the more recent texts. We showed that considering this additional information improves the similarity assessment of Co-Citation. Documents have a high CPA measure if many authors cite them together in close proximity. Document pairs that fulfill this condition are less suspicious from a CbPD perspective because they likely represent standard literature.

Decreasing the score of citation patterns in related work sections of a document is another strategy to reduce the impact of standard literature on the CbPD assessment. Reducing the score of matching citation patterns in documents with shared author sets can be desirable for focusing the analysis on detecting potential plagiarism instead of potential self-plagiarism. We plan to incorporate scoring functions that reflect these two strategies.

In the long run, we plan to combine our citation-based detection algorithms with text-based PD methods. One combination of the two approaches would be to employ the computationally less intensive citation-based methods as a preliminary filter to limit the number of documents subject to a computationally demanding text-based analysis.

REFERENCES

[1] T. Bretag and S. Mahmud. Self-Plagiarism or Appropriate Textual Re-use? *Journal of Academic Ethics*, 7: 193–205, 2009. doi: 10.1007/s10805-009-9092-1.

[2] E. Buyko, J. Wermter, M. Poprat, and U. Hahn. Automatically Adapting an NLP Core Engine to the Biology Domain. In *Proceedings of the Joint BioLINK-Bio-Ontologies Meeting*, pages 65–68, 2006.

[3] Christian Collberg, Stephen Kobourov, Joshua Louie, and Thomas Slattery. SPlAT: A System for Self-Plagiarism Detection. In *Proceedings of IADIS International Conference WWW/INTERNET 2003*, pages 5–8, 2003. URL splat.cs.arizona.edu/icwi_plag.pdf.

[4] P. Clough and M. Stevenson. Creating a corpus of plagiarised academic texts. In *Proceedings of the Corpus Linguistics Conference 2009, University of Liverpool, UK*, July 2009. URL http://ir.shef.ac.uk/cloughie/papers/CL2009.pdf.

[5] C. Collberg and S. Kobourov. Self-plagiarism in computer science. *Commun. ACM*, 48 (4): 88–94, 2005. doi: 10.1145/1053291.1053293.

[6] I. G. Councill, C. L. Giles, and M.-Y. Kan. ParsCit: An open-source CRF Reference String Parsing Package. In *Proceedings of LREC 2008*, number 3, pages 661–667. European Language Resources Association (ELRA), 2008. URL http://aye.comp.nus.edu.sg/parsCit/.

[7] M.-Y. Day, R. T.-H. Tsai, C.-L. Sung, C.-C. Hsieh, C.-W. Lee, S.-H. Wu, K.-P. Wu, C.-S. Ong, and W.-L. Hsu. Reference Metadata Extraction using a Hierarchical Knowledge Representation Framework. *Decis. Support Syst.*, 43: 152–167, February 2007. doi: 10.1016/j.dss.2006.08.006. URL http://portal.acm.org/citation.cfm?id=1223916.1223947.

[8] Deja Vu. A study of scientific publication ethics. Online Source, Dec. 2011. Retrieved May 29, 2012 from: http://dejavu.vbi.vt.edu/dejavu/.

[9] G. Divita, A. Browne, and R. Loane. dTagger: a POS tagger. In *AMIA 2006 Symposium Proceedings*, pages 200–203, 2006.

[10] M. Errami, J. M. Hicks, W. Fisher, D. Trusty, J. D. Wren, T. C. Long, and H. R. Garner. Déjà vu—A study of duplicate citations in Medline. *Bioinformatics*, 24 (2): 243–249, 2008. doi: 10.1093/bioinformatics/btm574.

[11] M. Errami, Z. Sun, T. C. Long, A. C. George, and H. R. Garner. Déjà vu: a database of highly similar citations in the scientific literature. *Nucleic Acids Research*, 37 (suppl 1): D921–D924, 2009. doi: 10.1093/nar/gkn546.

[12] B. Gipp and J. Beel. Citation Proximity Analysis (CPA) - A new approach for identifying related work based on Co-Citation Analysis. In *Proceedings of the 12th International Conference on Scientometrics and Informetrics (ISSI'09)*, volume 2, pages 571–575, Rio de Janeiro (Brazil), July 2009. International Society for Scientometrics and Informetrics. Available at: http://sciplore.org/pub.

[13] B. Gipp and J. Beel. Citation Based Plagiarism Detection - A New Approach to Identify Plagiarized Work Language Independently. In *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia (HT'10)*, pages 273–274. ACM, June 2010. doi: 10.1145/1810617.1810671. Available at: http://sciplore.org/pub.

[14] B. Gipp and N. Meuschke. Citation Pattern Matching Algorithms for Citation-based Plagiarism Detection: Greedy Citation Tiling, Citation Chunking and Longest Common Citation Sequence. In *Proceedings of the 11th ACM Symposium on Document Engineering (DocEng2011)*, pages 249–258. ACM New York, NY, USA, September 2011. doi: 10.1145/2034691.2034741. Available at: http://sciplore.org/pub.

[15] B. Gipp, N. Meuschke, and J. Beel. Comparative Evaluation of Text- and Citation-based Plagiarism Detection Approaches using GuttenPlag. In *Proceedings of 11th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'11)*, pages 255–258, Ottawa, Canada, June 2011. ACM New York, NY, USA. doi: 10.1145/1998076.1998124. Available at: http://sciplore.org/pub.

[16] HTW Berlin. Portal Plagiat - Softwaretest 2004. Online Source, 2004. Retrieved May 29, 2012 from: http://-plagiat.htw-berlin.de/ff-alt/05hilfen/programme.html.

[17] HTW Berlin. Portal Plagiat - Softwaretest 2008. Online Source, 2008. Retrieved May 29, 2012 from: http://plagiat.htw-berlin.de/software/2008/.

[18] HTW Berlin. Portal Plagiat - Softwaretest 2010. Online Source, 2010. Retrieved May 29, 2012 from: http://plagiat.htw-berlin.de/software/2010-2/.

[19] T. C. Long, M. Errami, A. C. George, Z. Sun, and H. R. Garner. Responding to Possible Plagiarism. *Science*, 323 (5919): 1293–1294, 2009. doi: 10.1126/science.1167408.

[20] R. K. Merton. The Matthew Effect in Science. *Science*, 159 (3810): 56–63, January 1968. doi: 10.1126/science.159.3810.56.

[21] A. Pertsemlidis and H. Garner. Engineering in genomics: text comparison based on dynamic programming. *Engineering in Medicine and Biology Magazine, IEEE*, 23 (6): 66 – 71, Nov.-Dec. 2004. ISSN 0739-5175. doi: 10.1109/MEMB.2004.1378640.

[22] S. Piao and Y. Tsuruoka. A Highly Accurate Sentence and Paragraph Breaker. Online Source, June 2008. Retrieved Jan. 28, 2011 from: http://text0.mib.man.ac.uk:8080/scottpiao/-sent_detector.

[23] M. Potthast, B. Stein, A. Barrón Cedeño, and P. Rosso. An Evaluation Framework for Plagiarism Detection. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 997–1005, Beijing, China, Aug. 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/C/C10/-C10-2115.pdf.

[24] Project SAX. Simple API for XML (SAX). Online Source, Apr. 2004. Retrieved May 29, 2012 from: http://www.saxproject.org/.

[25] B. Stein, M. Koppel, and E. Stamatatos. Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection PAN'07. *SIGIR Forum*, 41 (2): 68–71, December 2007. doi: 10.1145/1328964.1328976.

[26] B. Stein, E. Stamatatos, and M. Koppel, editors. *Proceedings of the ECAI08 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse, Patras, Greece, July 22, 2008*, volume 377 of *CEUR Workshop Proceedings*, 2008. CEUR-WS.org. URL http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-377/pan08-proceedings.pdf.

[27] B. Stein, P. Rosso, E. Stamatatos, M. Koppel, and A. Eneko, editors. *Proceedings of the 3rd PAN Workshop. Uncovering Plagiarism, Authorship and Social Software Misuse*, 2009. URL http://ftp.informatik.rwth-aachen.de/Publications/-CEUR-WS/Vol-502/pan09-proceedings.pdf.

[28] Z. Sun, M. Errami, T. Long, C. Renard, N. Choradia, and H. Garner. Systematic Characterizations of Text Similarity in Full Text Biomedical Publications. *PLoS ONE*, 5 (9): e12704, Sept. 2010. doi: 10.1371/journal.pone.0012704.

[29] The Apache Software Foundation. Apache OpenNLP. Online Source, 2010. Retrieved May 29, 2012 from: http://incubator.apache.org/opennlp/.

[30] The Stanford Natural Language Processing Group . Stanford CoreNLP - A Suite of Core NLP Tools. Online Source, Nov. 2010. Retrieved May 29, 2012 from: http://nlp.stanford.edu/software/corenlp.shtml.

[31] U.S. National Library of Medicine. Journal Archiving and Interchange Tag Suite. Online Source, Nov. 2008. Retrieved May 29, 2012 from: http://dtd.nlm.nih.gov/.

[32] A. Vohra and D. Vohra. *Pro XML Development with Java Technology*. Apress, Berkely, CA, USA, 2006. ISBN 1590597060.

[33] VroniPlag Wiki. Vroniplag - collaborative documentation of plagiarism. Online Source, Apr. 2012. Retrieved May 29, 2012 from: http://de.vroniplag.wikia.com.

[34] D. Walker, D. Clements, D. Maki, and J. Amtrup. Sentence boundary detection: A comparison of paradigms for improving MT quality. In *MT Summit Proceedings VIII*, 2001.

[35] D. Weber Wulff. Test cases for plagiarism detection software. In *Proceedings of the 4th International Plagiarism Conference*, Newcastle Upon Tyne, 2010. URL http://-www.plagiarismadvice.org/documents/conference2010/-papers/4IPC_0017_final.pdf.

[36] Webis Group, University Weimar. PAN 2011 Lab - Uncovering Plagiarism, Authorship, and Social Software Misuse. Online Source, 2011. Retrieved May 29, 2012 from: http://pan.webis.de/.

[37] M. J. Wise. String Similarity via Greedy String Tiling and Running Karp-Rabin Matching. Online Preprint, Dec. 1993. Retrieved May 29, 2012 from: http://vernix.org/marcel/share/RKR_GST.ps.

Citation for this Paper

Meuschke, N. & Gipp, B. & Breitinger, C., “CitePlag: A Citation-based Plagiarism Detection System Prototype,” in *Proceedings of the 5th International Plagiarism Conference*, 2012, DOI: 10.5281/zenodo.3483088.

BibTeX:

```
@inproceedings{Meuschke2012,
  title     = {{{CitePlag}}: {{A Citation}}-Based {{Plagiarism Detection System
              Prototype}}},
  booktitle = {Proceedings of the 5th {{International Plagiarism Conference}}},
  doi       = {10.5281/zenodo.3483088},
  author    = {Meuschke, Norman and Gipp, Bela and Breitinger, Corinna},
  year      = {2012}
}
```

RIS:

```
  TY  - CONF
  TI  - CitePlag: A Citation-based Plagiarism Detection System Prototype
  AU  - Meuschke, Norman
  AU  - Gipp, Bela
  AU  - Breitinger, Corinna
  C3  - Proceedings of the 5th International Plagiarism Conference
  PY  - 2012
  ER  -
```

Related Publications: www.gipp.com/pub