



LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

CENTRUM FÜR INFORMATIONS- UND SPRACHVERARBEITUNG
STUDIENGANG COMPUTERLINGUISTIK



Bachelor's Thesis

in Computational Linguistics

at the Ludwig-Maximilians-Universität München

Faculty of Languages and Literatures

Evaluation of Existing Plagiarism Research for the Optimisation of NLP-based Similarity Detection using Ludwig Wittgenstein's Remarks

Sabine Ullrich



LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

CENTRUM FÜR INFORMATIONS- UND SPRACHVERARBEITUNG
STUDIENGANG COMPUTERLINGUISTIK



Bachelor's Thesis

in Computational Linguistics

at the Ludwig-Maximilians-Universität München

Faculty of Languages and Literatures

Evaluation of Existing Plagiarism Research for the Optimisation of NLP-based Similarity Detection using Ludwig Wittgenstein's Remarks

Evaluation bestehender Plagiatsforschung zur Optimierung
der NLP basierten Ähnlichkeitssuche am Beispiel von Ludwig
Wittgensteins Bemerkungen

Author:	Sabine Ullrich
Advisor:	Daniel Bruder, M.Sc.
Supervisor and examiner:	Dr. Maximilian Hadersbeck
Work period:	03 April - 12 June 2017

Declaration

I hereby declare that this bachelor's thesis is my own work,
I have marked all citations and I have documented all
sources and materials used.

Munich, 12 June 2017

.....
Sabine Ullrich

Abstract

Similarity detection in documents seems to be a well-elaborated field of study. Scientists develop different approaches, ranging from fingerprinting to methods of Natural Language Processing (NLP) and graph-based detection. However, none of the existing works have tried to develop a hybrid approach that combines syntactic NLP methods, such as word position or Part-of-Speech (POS) tags, with the retrieval of semantic information. This thesis has focused on syntactic as well as semantic information, extending the state of the art in document similarity detection. Vector Space Models (VSMs) have been used to store the document information in vectors and to calculate the cosine similarity between two documents. Results show that this approach is of high relevance, improving the F-score of existing results by 0.15, achieving an F-score of 0.76.

Die Ähnlichkeitssuche in Dokumenten scheint auf den ersten Blick ein weit erforschtes Gebiet zu sein. Verschiedenste Ansätze wurden bereits entwickelt, welche von Fingerprinting bis zu auf Natural Language Processing (NLP) beruhenden und Graph-basierten Ansätzen reichen. Dennoch wurde bis heute nicht versucht mit einem hybriden Ansatz syntaktische Teile der NLP Methoden, wie zum Beispiel Wortposition und Part-of-Speech Tags, mit semantischen Informationen zu vereinen. Der Fokus dieser Arbeit liegt auf der Kombination von syntaktischer und semantischer Information, womit der State of the Art bei der Ähnlichkeitssuche in Dokumenten erweitert wird. Für das Speichern der Information in Vektoren wurden Vector Space Models verwendet. Zwischen den Vektoren wird dann der Kosinusabstand zweier Dokumente berechnet. Ergebnisse zeigen, dass dieser hybride Ansatz von hoher Bedeutung ist, da er den F-Score der bestehenden Ergebnisse um 0,15 verbessert und damit einen F-Score von 0,76 erreicht.

Acknowledgements

I would like to thank my supervisor Dr. Maximilian Hadersbeck for providing this interesting and challenging topic, the great chance to develop a new approach in the field of similarity research, and for the possibility to write this thesis at two places simultaneously during my time abroad. I really appreciate the opportunity to gain insight and contribute to the Wittgenstein Advanced Search Tool and to be part of this sophisticated project.

A big thanks to Daniel Bruder who not only listened to all my problems and questions, but always took his time to discuss current topics and helped me to develop and implement new ideas during my time in Cambridge. I consider myself very lucky to have had him as an advisor for this thesis, enhancing it with inspiring ideas and pushing me forward to where I am now. I cannot thank him enough.

Last but not least, I want to thank my family and friends for their unlimited support during my undergraduate studies in general, but especially during the year abroad, and for their encouraging words and their patience when I was busy working on this thesis.

Contents

Abstract	I
1 Introduction	1
1.1 Motivation	1
1.2 Outline	2
2 Related Work	3
2.1 NLP-based Approaches	3
2.2 Synonym-based Approaches	5
2.3 Other Approaches	5
3 Methods in Plagiarism Research	7
3.1 Types of Plagiarism	7
3.2 Plagiarism Detection Approaches	8
3.2.1 Intrinsic Plagiarism Detection	8
3.2.2 Extrinsic Plagiarism Detection	9
3.3 Overview of Plagiarism Detection Systems	12
3.3.1 Introduction of the Tools	12
3.3.2 Evaluation Methods	13
3.4 Experiments and Evaluation	14
3.4.1 Exact Copy Evaluation	15
3.4.2 Paraphrased Evaluation	15
3.4.3 Translated Evaluation	16
3.4.4 Small SEO Rewritten Evaluation	16
3.4.5 Evaluation of Document Similarity	17
3.5 Discussion	18
4 NLP-based Document Similarity	21
4.1 Methodology	21
4.1.1 Text Preprocessing	22
4.1.2 Feature Design	23
4.1.3 Similarity Measure	24
4.2 Implementation	26
4.2.1 Structure of the Feature Vector	26
4.2.2 Feature Extraction	27
4.2.3 Feature Weighting	31
4.3 Evaluation	32
4.3.1 Expected Results	32
4.3.2 Results and Evaluation	32
5 Conclusions and Future Work	35
5.1 Conclusions	35
5.2 Future Work	35

List of Abbreviations	37
Appendices	39
A Used Texts	41
B Detailed Results	45
C Lists of Stop Words	47
D Sample Program Output	49
List of Figures	53
List of Tables	55
CD Content	57
Bibliography	59

1 Introduction

The rapid growth of the World Wide Web during the last two decades has granted access to digital libraries, search engines and millions of web pages across the world. From the very first active website `info.cern.ch` in August 1991 [23], the number of webpages has increased extraordinarily quickly, until in 2014, Tim Berners-Lee, director of the World Wide Web Consortium (W3C), announced the milestone of one billion websites [2]. But it is not only different hostnames that are crucial in determining the growth of the web. In 2009, more than 50 million science papers had been published since 1665, in about 28,100 active peer-reviewed scholarly journals. Each year this number grows by 2.5 million [3]. With the Horizon 2020 programme, which was announced in a press release from the EU Presidency [35], all scientific articles in Europe will be freely accessible online to everyone by 2020, unless there are profound reasons for not doing so, such as intellectual property rights or security and privacy issues. This research programme will boost innovations and accelerate the development of new technologies and products, but simultaneously it may also facilitate plagiarism in academic work. Luckily, new technologies are being developed to detect and track plagiarism as it becomes easier to make unrecognised use of academic papers.

This chapter is divided into two sections. Section 1.1 will expose the incentive of developing a system that identifies document similarities. Further, Section 1.2 will give an overview of the structure of this work.

1.1 Motivation

Based on different plagiarism detection systems, documents can be compared with content on the World Wide Web, or with reference documents. Systems then detect possibly copied passages using syntactic or semantic features and yield results with the percentage of similar content.

The sentence similarity detection used in plagiarism detection systems can also be relevant to reveal similarities within a single document. The philosopher Ludwig Wittgenstein for instance, used to reuse information from his former works in later publications. However, he did not give reference to these recycled ideas. Nevertheless, it would be interesting to see which ideas he considered crucial enough to integrate them again in his works. For researchers in the field of philosophy it can therefore still be of importance to get an overview of Wittgenstein's reused ideas.

As will be shown in Chapter 2, existing research to date focuses either on syntactic or semantic similarity. No attempt has been made to combine these two approaches to improve results. This is why this work will not only take copied sentences or phrases into consideration, but also aim to detect paraphrased similarities and synonyms. Therefore, a wider range of features will be considered for the detection of similarities. The NLP-based approach will include lemma information, POS tags, word position and synonyms. The files to be analysed will be pre-processed in order to yield even more accurate results.

Stop words that are similar across all kinds of documents will be removed, as they do not contribute to the meaning of the document. It will be shown that this hybrid approach will extend the state of the art and that existing results will be improved.

1.2 Outline

The remainder of the thesis is organised as follows: Chapter 2 will discuss related work with different existing approaches, pointing out the lack of a hybrid approach that combines syntactic with semantic features. Chapter 3 will explain definitions that are used in this work, as well as structuring methods, providing an overview and evaluation of existing plagiarism detection tools. The proposed hybrid approach will then be presented in Chapter 4, describing the utilised methodology, implementation details and results. The work concludes in Chapter 5 with some suggestions for future work. A list of all introduced abbreviations can be reviewed on page 37.

2 Related Work

Many approaches have already been made with regard to similarity detection in documents. Some of them are based on features [1, 8], others on the hash structure of files [17, 28]. This chapter will briefly give an overview of related work in this field and then propose a new method for similarity detection, which combines existing methods and therefore yields more accurate results.

The remaining chapter is subdivided as follows: Section 2.1 will describe NLP-based approaches, while Section 2.2 will outline techniques that consider synonyms. Finally, Section 2.3 will survey miscellaneous methods in the field of similarity detection.

2.1 NLP-based Approaches

Several similarity detection techniques incorporate either syntactic or semantic information in their algorithms. While some of them tokenise the input files before comparing them, others include Machine Translation (MT) or Word Sense Disambiguation (WSD) systems. All these tasks form part of NLP. Some of the major NLP tasks are presented in Figure 2.1.

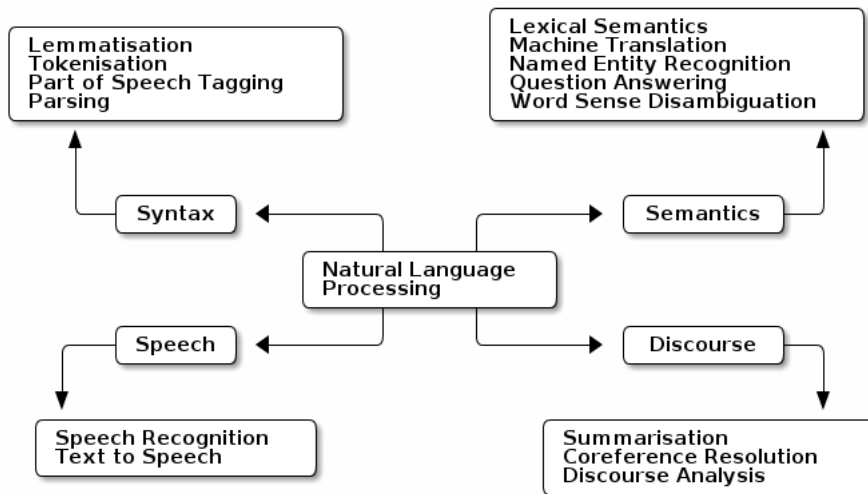


Figure 2.1: Natural Language Processing tasks

Ekbal et al. [8], for instance, perform text pre-processing on the input data to filter out irrelevant information before measuring the similarity of two files. Their pre-processing steps include splitting sentences, tokenising the document, retrieving lemmas, finding character offsets and determining POS classes with the Stanford NLP tool¹. Furthermore, they detect the file language and discard all non-English documents. Stop words are removed as well as lemmas belonging to certain POS classes. After the pre-processing steps, a subset of documents is selected, by means of methods from Information Retrieval (IR),

¹This NLP software is provided by the University of Stanford: <https://nlp.stanford.edu/software/>

using VSMs for document selection. More precisely, the documents are represented as vectors and the most similar documents are determined, calculating the smallest cosine angle between the vectors. The most similar documents are possible source documents.

However, Ekbal et al. point out that with this method, all terms are considered equally important, which is why some terms are assigned a higher importance than desirable. For example, a collection of documents about Machine Learning is very likely to have the terms *Machine* and *Learning* in almost every document. For that reason, they reduce the Term Frequency weight by a factor that increases with the number of occurrences in a document. This so-called Term Frequency-Inverse Document Frequency (TF-IDF) can now be combined to produce a composite weight for each term in a document as

$$\text{tf-idf}_{t,d} = \text{df}_{t,d} \times \log \frac{N}{\text{df}_t} \quad (2.1)$$

where df_t is the number of documents in the collection that contain a term t , and N is the total number of documents in the collection [18]. The TF-IDF weighting scheme assigns to a term t a weight in document d . A term receives the highest weight if it occurs many times in a relatively small number of documents. The assigned weight is lower when a term occurs fewer times in a document or occurs in many documents, and lowest when the term occurs in nearly all documents. The overlap score measure of a query q and a document d ,

$$\text{Score}(q, d) = \sum_{t \in q} \text{tf-idf}_{t,d} \quad (2.2)$$

is then the sum of the TF-IDF weight of each term in a document d . Lastly, the plagiarised passages are retrieved using a graph-based technique, with the depth first search algorithm, and false detections are filtered out. The algorithm thereby expands the left-most node until the deepest node is reached before expanding nodes to its right, see Figure 2.2. In doing so, nodes with a low score are discarded.

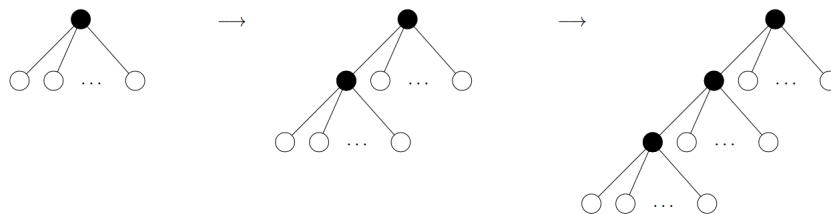


Figure 2.2: Depth-first tree-search algorithm, taken from Holden [16]

For the evaluation of this method to detect pair-wise similarity between documents, three obfuscation strategies were applied: Firstly, random text operations have been made, which insert or replace words and short phrases at random. Then semantic word variation have been carried out, i.e. words are replaced by one of their synonyms, antonyms, hyponyms, or hypernyms. Finally, words were shuffled at random, while retaining the original POS sequence. Ekbal et al. yield moderate results with their approach, attaining a recall value of 0.19 and precision of 0.66, on a scale of 0 to 1. Although addressing the point that one part of plagiarism is replacing words with their synonyms, and obfuscating their texts with semantic word variation, there was no incorporation of word synonyms. However, they leave this open to future work, proposing to integrate the lexical database WordNet to deal with semantic variations.

2.2 Synonym-based Approaches

In order to involve semantic word information, Abdalgader et al. [1] take synonyms into consideration. Rather than assigning a value of 0 or 1 for the vector entry, they insert a non-zero value if two words are semantically related. For a precise calculation of these “semantic vectors”, they first perform the Lesk algorithm to disambiguate words, that is, calculate the lexical overlap of the glossaries of two words. Lastly, they assign the correct expanded synonym set to a word. This enriched semantic context allows a more accurate estimate of the semantic similarity between two texts to be obtained. For the synonym extraction they use WordNet and measure the shortest path between two words, in order to determine how close they are. They store these word-sense pairs for all words but stop words in two sets S1 and S2, and combine them in the reduced vector space U, before calculating the similarity between the two vectors. An illustration of their method is given in Figure 2.3. With this combination of WSD and synonym expansion they achieve results with an accuracy of 0.70.

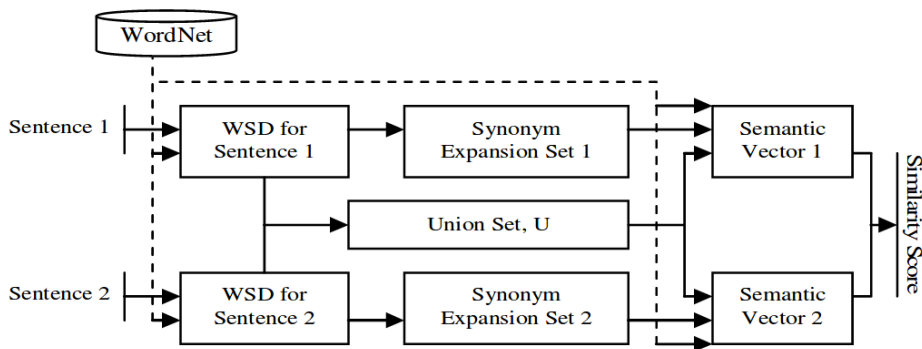


Figure 2.3: Synonyms and word sense disambiguation in similarity detection, taken from Abdalgader et al. [1]

2.3 Other Approaches

Rather than incorporating vectors, Tomita et al. [36] build subject graphs in order to calculate the similarity between documents. Subject graphs are an extension of the conventionally used term vectors. Each node in the graph has a weight that corresponds to the respective significance of the term association. For the similarity detection, they first extract all terms from the text, then calculate the significance of each term and create a term vector. Finally, the significance of each term-term association is calculated and an association vector is built. An example can be seen in Figure 2.4. The nodes in the graph represent the subjects, while every edge between two vertices is assigned a weight (e_{ij}).

The term-term association is calculated from the frequency of two terms in a unit, where the unit can be a sentence, clause or word window. Tomita et al. found that while sentence units slightly outperform clause units, the latter are still more precise, because a larger unit would yield excessively large associations and therefore fail to offset the increase in computational complexity. They conclude that subject graphs simplify the analysis steps from the knowledge discovery of a large volume of texts.

Hash-based or fingerprint methods are used by Sadowski and Levin [28]. They determine the similarity between two files by storing a set of hash keys and auxiliary data per file. Then they compare the pre-sorted hash key values (see Figure 2.5). To improve

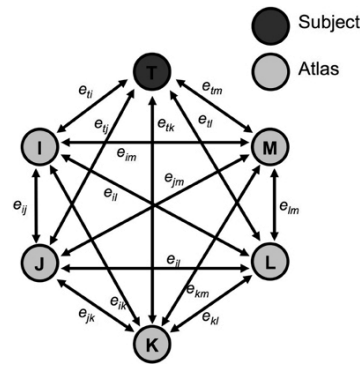


Figure 2.4: Example of a subject graph, as presented by Wang et al. [37]

performance, they preselect a set of strings to search for and then only compute the distance between the sum tables of two documents. In their experiments they show that an unbalanced weighting scheme works best on a realistic file set, while a more uniform weighting scheme performs better on artificial data sets. This method outperforms the others in terms of storage and performance. A major drawback remains, as this method does not take the ordering of the tag matches into account when creating key values and sum tables, that is, two files with rearranged content would yield a similarity value of 1. In other words, they would achieve the same similarity value as two identical files.

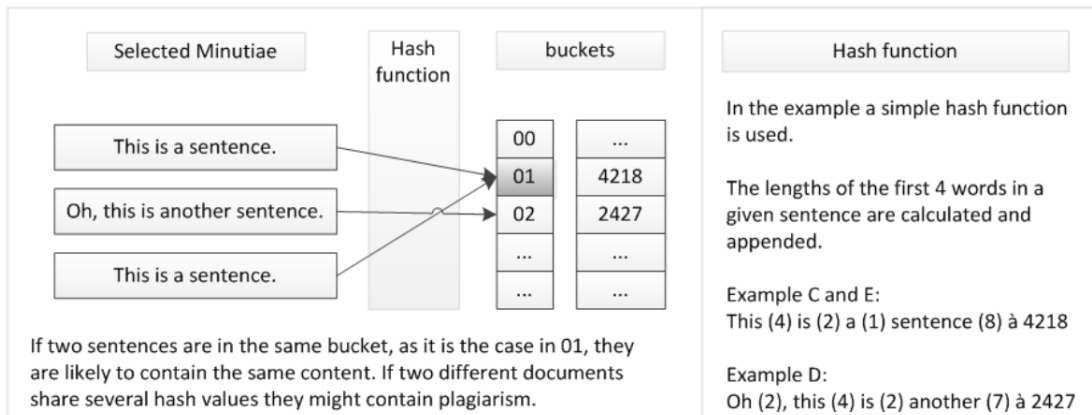


Figure 2.5: Similarity detection by means of a hash function, as presented by Gipp [11]

Kent et al. [17] combine fingerprint matching with four other features in the similarity detection process. They consider the top keyword feature, which is the word in the document with the highest frequency and should represent the overall idea of an article. Next, they take the first sentence into consideration, along with query phrases, which refers to the clause after certain words, such as “in conclusion” or “the experiment shows that”. Lastly, they take the longest common subsequence into consideration. The combination of these features seems to improve the accuracy of the results, however, they did not obtain a favourable result: The most frequent words in articles do not necessarily refer to the overall ideas, but might be frequent due to the author’s writing style. The first sentence feature is only effective in particular types of articles, such as news articles, but rather less effective in scientific papers. Also the number of query phrases should be increased in order to yield accurate results.

3 Methods in Plagiarism Research

As seen in Chapter 2, lots of approaches exist to detect plagiarism automatically [7, 11, 34]. As the field of Plagiarism Detection (PD) software becomes a major importance, scientists try to develop tools to improve the accuracy of such software. In 2009, the yearly workshop and evaluation lab, Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection (PAN) was initiated, which works on uncovering plagiarism, authorship, and social software misuse [27]. PAN organises yearly competitions and benchmark activities where scientists work on different shared tasks, including source retrieval, text alignment, cross-language text reuse detection, and external and intrinsic PD.

The following chapter will first give an overview of different types of plagiarism in Section 3.1. Then existing PD approaches will be explained in Section 3.2 before going into detail and looking at six specific systems in Section 3.3. Experiments and evaluation will be addressed in Section 3.4. Finally, results and research gaps will be discussed in Section 3.5.

3.1 Types of Plagiarism

There are several different types of plagiarism, some of which are more difficult to detect than others. The coarsest way would be to categorise plagiarism into intended and unintended plagiarism. However, it is not possible to prove whether a part of a document is plagiarised intentionally or by coincidence. If two documents share identical text however, we speak of exact copy plagiarism, as it is very unlikely to use the exact formulations and vocabulary as found in an existing source. Direct copying is the most obvious and provable type of plagiarism, manually or automatically, and refers to word-by-word copies without using quotation marks or any acknowledgement to the source [19].

In general, the following distinction can be made, which summarises the key types as found in the works of Eisa et al., Martin, and Mustofa et al. [7, 19, 22].

1. *Exact copy plagiarism*: copying phrases or the whole content of a document without reference to the source
2. *Paraphrasing plagiarism*: rewriting sentences or using synonyms, while the source text can still be recognised
3. *Translated plagiarism*: translating phrases or the whole content of a document from a language to another without reference to the source
4. *Plagiarism of secondary sources*: reference to the original source exists, but obtained from a secondary source without looking up the original
5. *Style plagiarism*: copying an author's style or concept
6. *Idea plagiarism*: copying someone's idea and reuse it as one's own
7. *Plagiarism of authorship*: putting one's name under someone else's work
8. *Self-plagiarism*: recycling or translating one's own ideas without reference [20, 21]

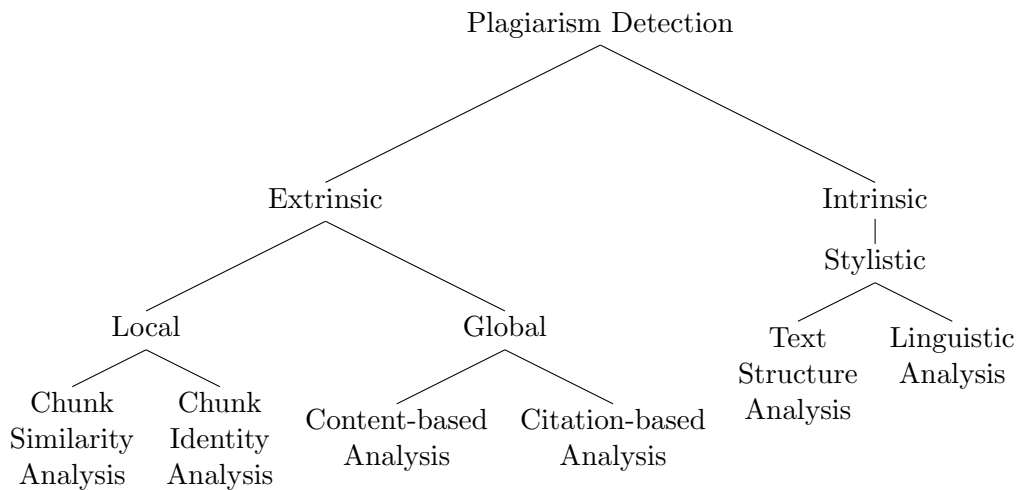


Figure 3.1: Plagiarism Detection Approaches

Note that some of the types are fundamentally difficult to detect, such as style or idea plagiarism, while others are easier to identify. Translated plagiarism for instance is extremely computationally expensive, as a bilingual dictionary is necessary for the translation the suspicious document. Mustafa et. al [22], for instance, work with translations from documents written in Bahasa Indonesia to English. Not only is the software restricted to these two languages, but also it is inherently difficult, if not impossible, to determine the language of the source document in the first place.

3.2 Plagiarism Detection Approaches

This section will present existing approaches to detect plagiarism. The hierarchy of all presented methods is displayed in Figure 3.1.

Subsection 3.2.1 will shortly outline intrinsic plagiarism techniques that focus on the assumption that only one document is provided without any given source documents. Next, Subsection 3.2.2 will present extrinsic methods, which are further divided into local and global methods. Then, global methods are again divided into content-based methods, including techniques based on n-grams, structure, semantics, and into citation-based methods that detect plagiarism based on language-independent markers, such as citations or formulas.

3.2.1 Intrinsic Plagiarism Detection

In some cases, reference documents cannot be provided or it is too time expensive to crawl through web content. In that case, we can draw back on intrinsic PD methods, which try to find plagiarised passages within a single document. Approaches are style based, that is, automatic systems try to detect stylistic changes and thus can determine whether a document is written by a single author or whether some of the content is copied from another document. Intrinsic PD is therefore closely related to the problem of author verification [33]. Stylistic changes can either be detected through text structure analysis, which determines if the structure of a paragraph deviates significantly from the remaining text, including vocabulary and average word length. On the other hand, a

linguistic analysis can be performed, focusing on the used grammar, word types and so forth. However, this method is not crucial for the current study, which is why it will be no further explained in detail.

3.2.2 Extrinsic Plagiarism Detection

As soon as reference documents are provided or a web-based search can be granted, PD systems can work with extrinsic or external methods. These methods can yield more accurate results, as they do not exclusively focus on stylistic changes but can directly compare the input with existing scholarly articles and work out similarities. Extrinsic PD can further be divided into local and global similarity assessment [34] as explained below.

Local vs. Global

Methods that are founded on local similarity directly link the number of identical regions in contiguous matching sequences. One way to do this is by matching sequences with length n , which retrieves word-for-word plagiarism. The most common local approach is fingerprinting, which is explained below. Global and local similarity detection are contrasted in Figure 3.2.

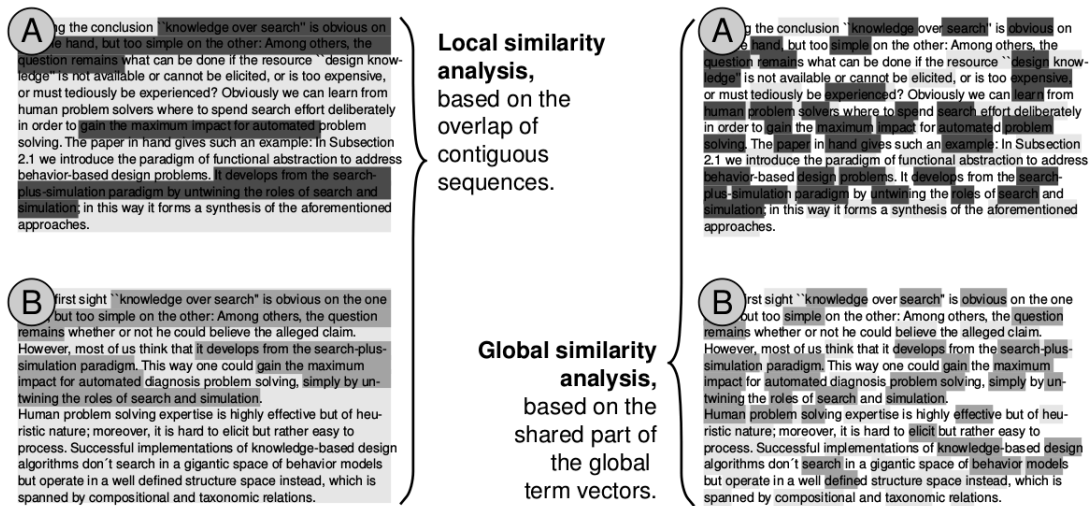


Figure 3.2: Local and global similarity detection, as presented by Stein [34]. The left side illustrates all identical n -grams with a length ≥ 5 between two documents A and B, while the right side shows global similarity, where common word stems without stop words of A and B are highlighted.

In order to get the fingerprint of a file, each document is divided into contiguous chunks of tokens. Instead of copying long passages of a document, a plagiariser would try to change sentence structure of copied material, so the longest chunks can be discarded. Similarly, short chunks can be disregarded, because the possibility of retrieving false positives is relatively high. For example, two general passages that are not representative can be found in several documents and are hence incorrectly identified as plagiarism. Next, the chunks are broken up into short byte strings, which helps to disguise documents and prevents the program from storing intellectual property of the files. This set of byte strings from a single document is called its signature [10]. The resulting byte strings are then stored in a hash table. If two document candidates share byte strings in their signatures, they are related and one of the documents possibly plagiarised the other. Formally, consider documents d_1

and d_2 with their set of their respective digested chunks, signatures $h(d_1)$ and $h(d_2)$ and their resolutions are $|h(d_1)|$ and $|h(d_2)|$. The local similarity is the intersection of the two fingerprints divided by their unification, which is described as [10, 34]:

$$sim_{loc}(d_1, d_2) = \frac{|h(d_1) \cap h(d_2)|}{|h(d_1) \cup h(d_2)|} \quad (3.1)$$

In case document d_1 is an excerpt of a much larger document d_2 , then the similarity $sim_{loc}(d_1, d_2) = 1.0$, which means the two documents share the same signatures. However if the similarity is 0, then there is no overlap of the two documents and very likely no case of plagiarism.

By contrast, global similarity handles longer text sections or even the complete document. As opposed to local similarity approaches, word order is neglected, and stop words removed. That is, two documents may have a strong similarity, although they may not share a single bigram [34]. Commonly used assessment methods are, for instance, VSMs. VSMs do not take order into consideration, but compare vector representations of a text as an unordered set [11]. A weight vector assigns weights to these terms according to the importance of the features. Finally, a similarity function computes “how matching terms of documents contribute to the calculation of a similarity score” [11, p. 29], for example with the standard cosine similarity measure. Finkel [10] calculates global similarity as

$$sim_{glob}(d_1) = \frac{|h(d_1) \cap (\bigcup_{d_2} h(d_2))|}{|h(d_1)|} \quad (3.2)$$

which is the intersection of the signature of d_1 and the unification of all chunks of d_2 , divided by the signature of d_1 .

This thesis will also be built around global similarity detection, more precisely around VSMs. Before going into detail, content-based and citation-based approaches will be presented in the next paragraphs, which both derive from global similarity, which has been illustrated in Figure 3.1.

Content-based vs. Citation-based

Methods in global similarity are based either on content or citations. The latter indicates that a suspicious document refers to the exact same documents as a source document [11]. On the other hand, content-based techniques include [7, 12, 24]:

- String-based detection techniques
- Structure-based techniques
- Semantic-based approaches
- Hybrid approaches

The most straightforward PD systems compare n-grams at character or word level. These n-grams can then be filtered, retrieving only stop word n-grams, n-grams with at least one named entity, or the most frequently used n-grams [12]. Gupta et al. [12] call these methods *string-based detection techniques*, as they solely refer to strings and therefore detect only simple cases of word-for-word plagiarism, or cases that contain small random shuffling. They point out that although string-based techniques are less efficient when it comes to complex types of plagiarism, they yield good results concerning precision

and can therefore be combined in hybrid approaches.

On the other hand, in *structure-based techniques*, units are not necessarily n-grams, but can be sentences, chunks or units that are based on POS Tagging or any other syntactic feature. These techniques can provide more detailed information on a document and reveal deeper manipulations [12]. Other techniques that include syntax are for instance duplication-gram, re-ordering, and alignment of words and phrase tags. Similarity scores are then calculated by considering the number of similar POS tags of the suspicious and the source document [7]. Structure-based approaches utilise representations with graphs and trees, where the nodes of a graph represent the sentences and the edges illustrate the attributes of the sentence. Each node is uniquely defined which allows a computation of the degree of similarity between two documents. Multilayer self-organising map-based approaches include additional structure-based features such as pages and paragraphs to detect plagiarism [7].

However, none of the methods described so far are able to detect paraphrasing plagiarism that includes word synonyms. *Semantic-based approaches* include the meaning of words and can thus detect sentences where words are replaced by their synonyms. These techniques include Semantic Role Labelling (SRL), Machine Learning (ML), and soft computing techniques [12]. In order to take the meaning of a document into consideration, models need dictionaries or thesauruses where they can look up synonyms and hypernyms. Fuzzy-based methods also include the semantics of a suspicious document while finding equivalent original texts through fuzzy numbers or sets [7].

Unlike content-focused methods, those based on citations do not only check language-dependent markers, such as n-grams, but also consider language-independent markers, i.e. citations, formulas or dates [11]. To quantify similarity, these markers are then inspected for overlap, distinctiveness, order and proximity. It is then possible to determine the bibliographic coupling strength, that is the absolute number of references that two documents share, as well as retrieving documents that share the same order of citations. The bigger the overlap of these features, the more likely it is that one document copied the citation structure of another. Besides detecting plagiarism, citation-based techniques can also be used as a filter, for example in verifying if the citation is giving credit to the original source or whether it only refers to a secondary source [25]. The citation-based approach is to date still rare, but a good example is for instance the software CitePlag¹ which was introduced by the University of Konstanz and is described in more detail in Gipp [11].

The following work will focus on a hybrid approach, which combines semantic and syntactic techniques. Before going into detail in Chapter 4, Section 3.3 will first give an overview of existing free plagiarism detection tools, and evaluate how well they perform.

¹<http://citeplag.org/>

3.3 Overview of Plagiarism Detection Systems

To see how existing tools perform, the following section will compare six freely available PD systems. The evaluated tools are Academic Plagiarism², CopyLeaks³, Plagiarisma⁴, Plagium⁵, PlagTracker⁶ and Small SEO Plagiarism checker⁷.

First, these systems will be tested for the identification of exact copy plagiarism and then analysed how they cope with manipulated paragraphs. This includes paraphrasing, translating, and replacing words with one of their synonyms. Then, two excerpts from Wittgenstein’s work will be taken and examined to what extent these systems manage to detect similarities.

3.3.1 Introduction of the Tools

Before outlining some of the tools’ advantages and drawbacks, each of them will be introduced shortly in the following paragraphs. It should be said that all six tools to be analysed are freely available online and that results are based on the free version if there exists a paid version of the software. All advantages and disadvantages of the presented software are summarised in Table 3.1.

	Advantages	Disadvantages
Academic Plagiarism	Queries can be saved	Limited number of queries Finds only exact matches
CopyLeaks	Open Source Optical Character Recognition Text comparison possible	Limited number of queries Slow performance
Plagiarisma	Fuzzy logic and Levenshtein distance	Separate queries required Limited number of queries
Plagium	Paragraph-wise Comparison	Low priority for free version
PlagTracker	Unlimited number of queries Text comparison possible	Limited number of queries
Small SEO	Unlimited number of queries	Poor results

Table 3.1: Comparison of existing plagiarism detection tools.

According to their website, Academic Plagiarism checker uses advanced algorithms to scan for and categorise plagiarised content. To find plagiarism, the tool compares the document to billions of websites as well as a large collection of essays gathered online. However, this is only included in the premium version. The free software compares documents with the web only. Also, only exact sentences are matched, while the paid version allows a dynamic match, which should then find paraphrased sentences and synonyms as well. The algorithm automatically detects and ignores quoted text. While the paid version allows five scans a day, free users can only submit one document every three days.

²<https://academicplagiarism.com/>

³<https://copleaks.com/>

⁴<http://plagiarisma.net/>

⁵<http://www.plagium.com/>

⁶<http://www.plagtracker.com/>

⁷<http://smallseotools.com/plagiarism-checker/>

Copyleaks works with cloud computing, and scans more than 60 trillion webpages and databases. Their algorithm works with various levels of duplicate content, which should detect word-for-word plagiarism, paraphrasing, and sentence restructuring. The free version of Copyleaks grants ten pages (250 words per page) every month. It supports a vast number of file types, all Unicode languages including Asian languages, and even Optical Character Recognition (OCR) to compare pictures and graphics that contain textual content. The latter distinguishes this tool from the others, along with the possibility to compare two documents against each other.

Plagiarisma is split into four different plagiarism detectors, namely systems that search Google, Yahoo, Books and Scholar. In order to get accurate results, four different queries should be started, one for each search engine. Each query must contain between 30 and 2000 characters. With the creation of a free account, the tool allows the use of their similarity checker, which compares text and files using fuzzy logic and Levenshtein distance.

Plagium advances paragraph-wise, then applies grammar and word usage rules to break up each paragraph into chunks of text. These chunks are then compared to the web. Following that, the retrieved documents and webpages that correspond to each of those paragraphs are displayed. Queries are limited to 5000 characters, with the option to get the premium version to analyse longer text.

PlagTracker, as the other tools, scans papers and checks them against a huge database of millions of published works. For the free version, it is not necessary to create an account. Also, there is no limited number of queries, however, each query is limited to 5000 words. In order to check longer documents, users can upgrade to premium, which will also enable the function to exclude a list of references from the text. Similarly to the tools CopyLeaks and Plagiarisma, PlagTracker includes the feature to compare two files against each other.

Finally, Small SEO (Search Engine Optimisation) checks sentences one by one and compares them with already indexed content retrieved by various search engines. As opposed to the other tools, Small SEO has no limitations and therefore does not need a premium version.

3.3.2 Evaluation Methods

The tools described in the previous subsection are evaluated as follows: All sentences are labelled manually either “plagiarised” or “not plagiarised” (P or \bar{P}). These labels serve as references for the system’s output. The hand-labelled data is referred to as the *gold standard* (Table 3.2 uses the short term *gold*). Sentences that are considered plagiarism in the gold standard and are at the same time detected as plagiarism by the detection system (referred to as *sys* in Table 3.2) are called True Positives (TP). Sentences that do not contain plagiarised parts and are correctly ascribed are True Negatives (TN). Equally, plagiarised sentences are either missed, that is, False Negatives (FN), or incorrectly marked as “not plagiarised”, namely False Positives (FP). An illustration of this evaluation method is given in Table 3.2.

To make systems comparable, the established measures *precision* and *recall* are determined. While the recall value reflects what proportion of sentences in the gold standard were detected, precision shows what proportion of the output is correct. If there are no

	P_{sys}	\bar{P}_{sys}
P_{gold}	TP	FN
\bar{P}_{gold}	FP	TN

Table 3.2: Fourfold table reflecting how values are determined

possible correct answers in the gold standard, $recall = 1$; if there are no answers given by the system, then $precision = 1$. Equation (3.3) shows how the values in Table 3.2 are utilised to calculate precision and recall.

$$Recall = \frac{TP}{TP+FN} \quad Precision = \frac{TP}{TP+FP} \quad (3.3)$$

In general, precision and recall oppose each other, that is, as precision goes up, recall usually goes down. This is why F-measure is used to combine the two values and compare systems according to both measures, as can be seen in Equation (3.4). If $\beta = 1$, we talk about F_1 .

$$F\text{-measure} = \frac{(\beta^2 + 1)PR}{\beta^2P+R} \Rightarrow F_1 = \frac{2PR}{P+R} \quad (3.4)$$

The factor β is used to favour either precision or recall, if $\beta > 1$ precision is favoured, while recall is favoured for all $\beta < 1$. However, as in most cases there is no particular reason to give preference to either of them, F_1 is used to receive a balanced value. Results in this thesis therefore also consider F_1 only.

3.4 Experiments and Evaluation

According to the methods presented in Subsection 3.3.2, the existing tools are evaluated. To test the systems for different degrees of obfuscation, the first remark from Wittgenstein’s Typescript (TS) 207 [26] is taken and modified. The abstract used in these experiments comprises the introductory words of Wittgenstein’s Lecture on Ethics and is composed of 16 sentences.

In order to test for different degrees of disguise, the remark has been changed in three different ways. First of all, the remark has been copied and pasted, and tested for similarity (Subsection 3.4.1), then it has been slightly paraphrased and some words have been replaced by synonyms (Subsection 3.4.2). The tools have been tested for translated plagiarism in Subsection 3.4.3. Lastly, Subsection 3.4.4 evaluates how well the tools deal with text where the remark has been completely modified by replacing words with one of their synonyms with the Small SEO Article Rewriter.

The complete texts can be found in the Appendix, Tables A.1 to A.4. Similarly, detailed results can be found there, Tables B.1 to B.5. All results are summarised in Table 3.3.

Additionally, the three tools CopyLeaks, Plagiarisma and Plagium, support the feature to compare two documents against each other. The manipulated texts from TS 207 are also tested against the original with these tools. Furthermore, Wittgenstein’s TS 213 [26]

⁸This result could not be determined, as PlagTracker was unavailable online in the middle of the experiments. The overall result will therefore be calculated using the other three factors.

F ₁ value	exact	paraphrased	translated	SEO Rewriter	overall
Academic Plagiarism	0.90	0.12	0.12	0.12	0.38
CopyLeaks	0.97	0.77	0.00	0.00	0.43
Plagiarisma	1.00	0.67	0.55	0.22	0.61
Plagium	1.00	1.00	0.12	0.00	0.53
PlagTracker	1.00	0.00	0.00	n/a ⁸	0.33
Small SEO	0.81	0.48	0.00	0.00	0.32

Table 3.3: Evaluation of existing plagiarism detection tools.

is compared to Manuscript (MS) 114 [26], showing how the author reused his ideas in different workings. Results will be presented in Subsection 3.4.5.

3.4.1 Exact Copy Evaluation

In the first experiment, the systems are tested for exact copy plagiarism as described in Section 3.1. The extract is directly taken from Wittgenstein’s TS 207 and pasted into the different PD systems. Plagiarisma, Plagium and PlagTracker search the web successfully and define all 16 sentences as plagiarised. Academic Plagiarism, CopyLeaks and Small SEO fail to determine the first part in the TS “Ladies and Gentlemen”, and therefore yield slightly lower recall values. However, this copy-and-paste approach is the easiest one to detect, and web-based tools were able to reveal most of the plagiarised parts, achieving high F₁ values ranging from 0.81 to the maximum of 1.00.

3.4.2 Paraphrased Evaluation

Second, the same text excerpt is slightly changed: The syntax has been adapted and some of the words have been replaced by synonyms. Plagium yields the best result, classifying the text to be 100% plagiarised. CopyLeaks and Plagiarisma still attain accurate results with F₁ values of 0.77 and 0.67, respectively. PlagTracker and Academic Plagiarism do not find any similar results. The latter can be explained because searching for rewritten passages is not part of the freely available version of the tool. An excerpt of the paraphrased remark is shown in Table 3.4.

Original	Paraphrased
Before I begin to speak about my subject proper let me make a few introductory remarks. I feel I shall have great difficulties in communicating my thoughts to you and I think some of them may be diminished by mentioning them to you beforehand.	Before I begin to talk about my subject proper, I want to make a few introductory remarks. I feel I will have big problems in communicating my thoughts to you and I think some of them may be diminished by mentioning them to you in advance.

Table 3.4: Example of the paraphrased text

3.4.3 Translated Evaluation

In order to find translated plagiarism, the 16 sentences mentioned above have been translated into German with Google Translate and subsequently been corrected manually. An example can be seen in Table 3.5. This seems to be the most challenging disguise, as almost every tool yields an F_1 value of 0.00. Although Plagium detects plagiarised sentences, yielding an F_1 of 0.11, the retrieved web pages with apparent plagiarism do not contain contents of Wittgenstein’s remarks. They rather share short random phrases in common, such as “that means” or “he either believes”. The only successful software among the tools to be tested is Plagiarisma, finding a website that cites a part of Wittgenstein’s TS 207 in German, which is why 6 out of 16 sentences are correctly classified.

Original	Translated
Before I begin to speak about my subject proper let me make a few introductory remarks. I feel I shall have great difficulties in communicating my thoughts to you and I think some of them may be diminished by mentioning them to you beforehand.	Bevor ich anfangen, über mein Thema zu sprechen, lassen Sie mich ein paar einleitende Bemerkungen machen. Ich glaube, ich werde große Schwierigkeiten haben, meine Gedanken an Sie zu vermitteln, und ich denke, dass einige von ihnen vermieden werden können, indem ich sie Ihnen gegenüber im Voraus erwähne.

Table 3.5: Example of the translated text

3.4.4 Small SEO Rewritten Evaluation

Finally, the short text has been transformed with the Small SEO Article Rewriter⁹. This freely available tool replaces words with their synonyms, which helps to disguise plagiarism, making it inaccessible to detection systems that do not include the semantics of the words. Systems that are solely web-based are not able to retrieve sentences where content words are replaced with synonyms which shows that they are unable to uncover transformed passages. Table 3.6 gives an impression of how the transformed text looks. Academic Plagiarism and Plagiarisma yield low F_1 values of 0.12 and 0.22, respectively, while none of the other tools are able to detect similarities between the original and the disguised text.

Original	Small SEO rewritten
Before I begin to speak about my subject proper let me make a few introductory remarks. I feel I shall have great difficulties in communicating my thoughts to you and I think some of them may be diminished by mentioning them to you beforehand.	Before I begin to talk concerning my subject proper let me build a couple of introductory remarks. I feel I shall have great difficulties in communicating my thoughts to you and that I assume a number of them could also be diminished by mentioning them to you beforehand.

Table 3.6: Example of the Small SEO rewritten text

⁹<http://smallseotools.com/article-rewriter/>

3.4.5 Evaluation of Document Similarity

Mainly, the PD tools compare text against web pages. The three tools CopyLeaks, Plagiarisma, and Plagium also allow to compare two documents against each other. The first document is the original text, again taken from Wittgenstein’s TS 207, as in the subsections before. Then the document to be compared is the paraphrased, translated, or rewritten text, respectively. The systems output the percentage of similar content of the two texts. Table 3.7 shows how well each of the tools performed.

	Paraphrased	Translated	Small SEO	Overall
CopyLeaks	65.40%	0.00%	18.10%	27.83%
Plagiarisma	81.43%	45.03%	60.92%	62.46%
Plagium	55.40%	0.00%	55.20%	36.87%

Table 3.7: Results for different degrees of disguise

As can be seen, CopyLeaks and Plagium fail to detect any kind of translated plagiarism, which reveals they do not include dictionaries in their algorithms. Plagiarisma stands out detecting 45.03% similarity for the translated part. As for the other results, the tools perform quite moderately. Plagiarisma outperforms the other tools slightly, detecting 81.43% similarity for the paraphrased passage and 62.46% similarity for the rewritten text.

TS 213	MS 114
Augustinus beschreibt wirklich einen Kalkül; nur ist nicht alles, was wir Sprache nennen, dieser Kalkül. (Und das muß man in einer großen Anzahl von Fällen sagen, wo es sich fragt: ist diese Darstellung brauchbar oder unbrauchbar. Die Antwort ist dann: “ja, brauchbar; aber nur dafür, nicht für das ganze Gebiet, das Du darzustellen vorgabst”.)	Augustinus beschreibt einen Kalkül unserer Sprache, nur ist nicht alles, was wir Sprache nennen, dieser Kalkül. (Und das muß man in vielen Fällen sagen, wo die Frage vor uns steht: “ist diese Darstellung brauchbar, oder unbrauchbar”. Die Antwort ist: “ja, brauchbar, – aber nur dafür; nicht für das ganze Gebiet, das Du darzustellen vorgabst”.) S. 179 A

Table 3.8: Wittgenstein’s TS 213 compared to MS 114 [26]

So far, only artificially created similarity has been compared. This means that all compared texts were adapted for research purposes. In order to identify non-artificial similarity, Wittgenstein’s TS 213, and MS 114 [26] will be compared. The two remarks are similar to a human reader, but the variations might be more challenging for an automatic detection system. The two remarks contain the same content, while one of them has different filler words than the other, as can be seen in Table 3.8.

It is not obvious to the tools, however, that these documents share the same content. CopyLeaks, for instance, fails to detect any similarities where sentence structure has been changed. Table 3.9 shows the percentages of similar content that the tools detect.

Plagiarisma again outperforms the other tools, detecting 85% similar content, followed by Plagium which retrieves 75% similarity and CopyLeaks only revealing 23% of parallel content. All results will be discussed in the next section.

	Similarity
CopyLeaks	23%
Plagiarisma	85%
Plagium	75%

Table 3.9: Results for the comparison of TS 213 and MS 114

3.5 Discussion



Most of the free plagiarism tools, namely CopyLeaks, Plagium, PlagTracker, and Small SEO Plagiarism Checker, start a Google or Yahoo search and determine the number of web pages with the same content, as well as a percentage of plagiarised or unique phrases. They split the text to be analysed into chunks of different sizes, depending on the tool, and start several queries searching for the content on the internet. As for document comparison, the tools split the documents into similar parts and compare them successively.

Summarising the results, the tools yield good outcomes for direct plagiarism, where the content has been copied and pasted the way it is; achieving results that range from 0.81 to 1.00. The F-measure decreases as the paragraph is adapted and sentence structures changed. However, some tools still perform well. Plagium scores an F_1 of 1.00, followed by CopyLeaks and Plagiarisma with 0.77 and 0.67, respectively. Already at this stage, Academic Plagiarism and PlagTracker fail to make a connection to the original document. All tools fail to reach satisfying results for translated texts as well as texts that were transformed with the Small SEO Article Rewriter. In a nutshell, precision and recall values are high for direct plagiarism, varying from high F_1 values from 0.8 to 1.0, while F_1 decreases with the complexity of disguise.

It should be noted that the existing tools fail to make a distinction between *similarity* and *plagiarism*. The systems claim to detect plagiarism, but in fact they do not consider whether a document cites another source properly or not. In other words, they try to identify similarities across texts, while not examining if a document plagiarises another or cites it adequately. A tool might rate a student's paper, therefore, with a plagiarism amount of 60%, while in fact it is much lower, such as 2%, because the student has marked all citations [5].

Moreover, it is important to note that the outcome of these tools cannot be accepted without searching manually for plagiarised content. This is because the tools are only able to detect similarities in files that are not manipulated. To help students to disguise plagiarism, there are blogs online that outline tips on how to cheat PD systems. One of those blogs is for example "How to cheat Turnitin" [5]. According to the blogger, the easiest way to cheat the detection systems is to manipulate a document in such a way that the systems fail to extract its content properly. A possibility is to add a tilde (\sim) after every letter 'a', and then enable the macros in Microsoft Word before the paper is submitted. The tilde becomes invisible to the examiner and the system fails to find similarities with the source documents. Heather [14] adds that another way to manipulate a PDF file is by modifying the Character map (Cmap), which maps character identifiers to character codes. These entries are needed if different fonts store their glyphs in another order. However, if the Cmap is modified by swapping two entries (see Figure 3.3), this reassignment will

produce completely garbled outcome. Modifying the glyphs themselves, such that they no longer correspond to the alphabet yields equally incoherent results. Lastly, each character in the document to be analysed could be replaced with the Bézier curve that defines the character glyph. The curve basically paints the letters without storing any machine-readable letters behind them. As systems reject input if a file is empty, random letters and words must be added and made invisible, such that the detection system cannot determine that the file has been manipulated. Heather [14] proposes a solution to fix this problem by employing OCR in detection systems, which will check hidden text for plagiarism. Of the systems that have been tested, CopyLeaks is the only one that incorporates OCR in its search.

Character	e	x
Glyph		



Character	e	x
Glyph		

Figure 3.3: Glyphs in an unmodified (left) and modified (right) font [14]

Furthermore, none of the systems appear to draw on NLP methods. As a consequence they detect unoriginality in short chunks in paper headers because papers share content words at several points. Retrieving irrelevant similarities then falsifies the final score of the plagiarism percentage. This problem could be dealt with by removing stop words before analysing the documents.

In the present work, this last issue should be addressed in order to improve results. On the other hand, intentional cheating should not be considered, because including OCR would go beyond the scope of this work, as it is too time-consuming. Also, the following work will focus on document similarity detection rather than plagiarism detection, without regard to citations used. This is because accurate plagiarism detection is dependent on human interaction to see if similar text passages are indeed copied without reference to the source. Similarly to experiments in Subsection 3.4.5, two texts will be compared, as opposed to searching the internet, and a degree of similarity measured.

4 NLP-based Document Similarity

As could be seen in Chapter 2, research in plagiarism and similarity detection is still to be improved. The analysed work focuses on various different approaches instead of combining them and yielding better results. In the previous chapter, these isolated methods were reflected in the results, as free plagiarism tools are not as accurate as we would like them to be. It is therefore questionable why so few hybrid approaches in the field of similarity detection exist. In this thesis, not only methods of the syntactic part of NLP are considered, including word position and POS tags, but also the semantic part, including synonyms, is taken into account. Combining these syntactic and semantic features, this new hybrid approach considers more factors, leading to preciser results than existing methods. The remaining chapter is further subdivided as follows. Section 4.1 will present the applied methodology of the hybrid approach, Section 4.2 will show how these methods were implemented and finally, Section 4.3 will show the evaluation of the presented approach.

4.1 Methodology

As mentioned before, none of the existing papers have so far dealt with the combination of NLP methods and semantic similarity. Although Abdalgader et al. [1] take synonyms into consideration, they do not deal with the position of a word in a given file, which leads to a higher similarity score than expected. Ekbal et al. [8] take all NLP steps into account but disregard semantic similarity of sentences. Subsequently, a new methodology will be proposed. Subsection 4.1.1 will outline text pre-processing methods and Subsection 4.1.2 will explain the included features in the new approach. The section concludes in Subsection 4.1.3, proposing a similarity measure for the experiments. Figure 4.1 outlines the procedure of the new hybrid approach.

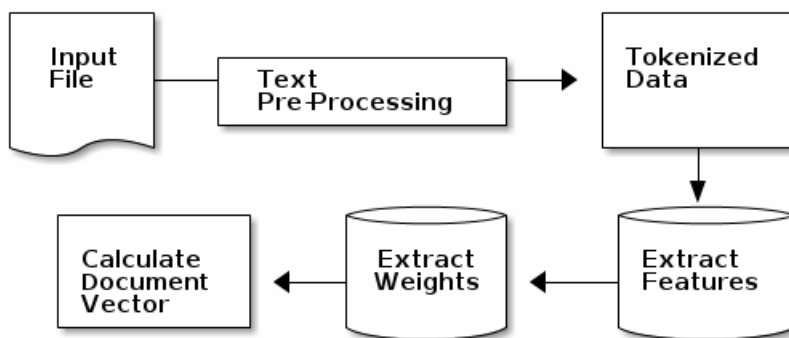


Figure 4.1: Illustration of the methodology

4.1.1 Text Preprocessing

Before features can be extracted from the documents, the texts should be adjusted by pre-processing. Several existing methods calculate document similarity based on words themselves, sometimes including semantic information such as synonyms. In any case, it is indispensable to process files before comparing them.

In short, the used text pre-processing steps include:

- Tokenisation
- Lowercasing (English)
- Punctuation removal
- Stop word removal
- Number Replacement

Documents cannot be compared as they are, but rather segments should be compared. In this work, these segments should be tokens that were separated by whitespace. Splitting by newline is not necessary because the input is read in line by line. Note the difference between tokens, types and terms pointed out by Manning et al. [18]. While a token is an instance of a sequence of characters, a type is the class of all tokens containing the same character sequence. After removing common expressions without meaning, normalised types are left, so-called terms. In this case, as the texts are split by whitespace only, tokens still contain punctuation.

After the tokenisation step, two identical words may still not share the same token. This is because words at the start of a sentence begin with an uppercase letter, while words at all other positions have a lowercase letter as their initial character. This is why all words should be lowercased, such that they can be compared with the words at every other position. This step is however unreliable for German, where all nouns are capitalised. This information is needed for POS tagging. If all words are lowercased, the tagger will not recognise German nouns any more and categorise them incorrectly. For this reason, only English texts will be uncapitalised, while German files will maintain their case.

The lowercased tokens still comprise punctuation marks, which have to be removed from every token. If this step is left out, two words “little” and “little,” are not identified as the same word. After the removal of punctuation, the documents consist of “clean” data.

As has been discussed in Section 3.5, results of current tools are falsified due to an overlap of common words in various papers. These so-called “stop words” do not contribute to the meaning of a document and can thus be disregarded. In the pre-processing steps, English and German stop words are removed, where the language depends on the settings. English stop words are removed by default if no language is specified. An extract of the removed stop words can be seen in Table 4.1; full lists are specified the Appendix, Tables C.1 and C.2.

Lastly, documents may contain references to other papers and scientific work. However, the numbering in these references is very likely to vary from the document to be compared. Equally, papers refer to tables and figures with varying numbers. As a consequence, these numbers would falsify the similarity results [6]. Instead of deleting them, but still to indicate that there is a number, all digits are replaced with the dummy string “[NUM]”.

English	German
a, about, above, after, again, against, all, am, an, and, any, are, aren't, as, at, be, because, been, before, being, below, between, both, but, by, can't, cannot, could, couldn't, did, didn't, do, does, doesn't, doing, don't, down, during, each, few, for, from, further, had, (...), you've, your, yours, yourself, yourselves	aber, alle, allem, allen, aller, alles, als, also, am, an, ander, andere, anderem, anderen, anderer, anderes, anderm, andern, anders, auch, auf, aus, bei, bin, bis, bist, da, damit, dann, das, dass, dasselbe, dazu, (...), wo, wollen, wollte, während, würde, würden, zu, zum, zur, zwar, zwischen, über

Table 4.1: Extract of English and German stop words

4.1.2 Feature Design

After pre-processing the two files, the features that should be included in the vector can be extracted. The experiments have been carried out with five different features, which are as follows:

1. Word itself
2. Lemma
3. POS tag
4. Position
5. Synonyms

Primarily, all words should be stored in the feature vector, in order to find exact copies of text and to distinguish them from rewritten text. Along with the words themselves, stems or lemmas should be taken into account. Take two words with the same lemma, such as *imagine* and *imagination*. Needless to say, they should receive a higher similarity value than words that do not share the same word stem or lemma. Especially in highly inflected languages, words with the same lemma vary in suffixes in different positions of a sentence. This is why extracting the words' lemmas is useful for detecting their similarity, as they would have a similarity value of zero otherwise.

It is also crucial to take the POS tag into account, which helps to consider sentence structure and meaning of the words. POS can help to disambiguate two words and does not yield an inappropriately high similarity value for two different words that share the same spelling. This syntactic and semantic information is crucial to obtaining accurate results.

Imagine two sentences s_1 and s_2 , where s_1 is the original one. In sentence s_2 , every word in s_1 has been replaced by its synonym without referencing s_1 . This is considered paraphrased plagiarism, as described in Section 3.1. In order to detect this, words that share the same meaning must be included in the feature vector. Further, without including POS, the word *fly* in Example 4.1 is considered the same and the two sentences yield an inappropriate similarity value.

- (a) They let him *fly* back to Virginia.
- (b) A *fly* is a small insect. (4.1)

In terms of lexical generalisation, the synonyms of each lemma are considered. For the English language, synonyms are retrieved from WordNet, a large lexical database of English [9]. In WordNet, nouns, verbs, adjectives, and adverbs are grouped into sets of cognitive synonyms, so-called synsets. These synsets are used to compare the semantic similarity of two sentences. WordNet includes 155,287 lexical units and 117,659 synsets.

Similar to WordNet is the lexical-semantic database GermaNet, which retrieves German synonyms accordingly. Lexical units that express the same concepts are also grouped into synsets [13, 15]. GermaNet was developed and maintained within various projects at the research group for General and Computational Linguistics Division of Computational Linguistics of the Linguistics Department, University of Tübingen since 1997. It comprises 110,167 synsets and 142,814 lexical units and is thereby just a little less extensive as its English counterpart.

Consider two documents d_1 and d_2 consisting of one word each, $d_1 = \textit{small}$ and $d_2 = \textit{little}$. Without including synonyms and POS tags in the vector, the $\textit{sim}(d_1, d_2) = 0$, meaning they do not share anything in common. With regard to the features mentioned above, d_1 and d_2 may not share the same word and lemma, but they are synonyms and share the same POS tag ($\textit{POS} = \textit{adjective}$). The similarity of the two vectors is $\textit{sim}(d_1, d_2) = 0.5$, provided every feature is assigned the same weight. This shows that by including additional information about the words, even paraphrased or other types of disguised plagiarism can be revealed.

Finally, related work has shown [1] that including semantic information is an important factor in determining document similarity. Nevertheless, they yield higher similarity scores than desired, as they only refer to synonyms without regard to word position in a text. Considering this feature is a crucial point however, as two documents may share the same words, but may not share any semantic similarity due to a completely different ordering of the words. For that reason, the position of a word will be considered and included in the feature vector. This will be especially important for low inflected languages, such as English, where the position of a word reflects its function in the sentence.

4.1.3 Similarity Measure

After designing the feature vector, an appropriate similarity measure must be chosen to compare the vectors of two documents. There are several different approaches to calculate the similarity between two vectors. A detailed description can be found in Cha [4]. The distance of two vectors can be calculated for example using cosine similarity, Euclidian distance, or the Jaccard similarity index.

The latter compares members for two sets and determines if these sets are shared or distinct. These members can either be symmetric, i.e. of equal importance (gender, marital status, etc), or asymmetric, meaning the members have different levels of importance (testing positive for a disease). Jaccard similarity is easy to interpret, however, it is sensitive to small document sizes and could give erroneous results.

On the other hand, Euclidean distance measures the distance between particular points of interest along the vector. It may be less useful in determining document similarities as long documents and short documents could yield a high similarity value just because they share a high number of common words in a topic.

Cosine similarity normalises the length of documents and is therefore a popular similarity measure in the context of text mining for comparing documents. For this reason, cosine similarity will also be used in this work. Manning et al. [18] present Cosine Similarity in terms of IR (see Figure 4.2), where a query q is compared to various documents d_1 to d_n . The documents and the query are displayed as vectors, and the cosine similarity measured. The smaller the angle between two vectors, the more similar they are. Like this, documents that are similar to the query are retrieved. With regard to similarity detection between two documents, only the two documents should be compared, without reference to a query, and the cosine similarity is calculated.

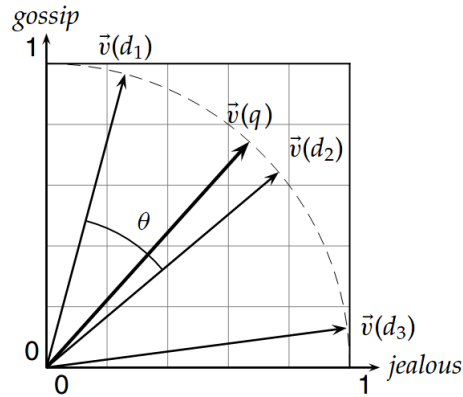


Figure 4.2: Cosine similarity between three documents vectors d_1 to d_3 and a query vector q as presented by Manning et al. [18], where q comprises the words *jealous* and *gossip*

This measure is especially useful when the length of two vectors varies significantly. For example, two documents d_1 and d_2 share very similar content, but d_1 is considerably longer than d_2 . This means they have significant vector difference simply because of the varying document length. In order to normalise the vectors, and to treat them as if they were all of the same size, the cosine similarity of the two document vectors should be computed. In this thesis however, the vectors are not simple Bag of Words (BOW) vectors, that store simply information about word occurrences in a document disregarding their word position. Instead, the vectors store additional word information, which means they are all artificially of the same length. Like this, the normalisation of cosine similarity is no longer required, but still selected to measure the percentage of similar content of the documents. Equation (4.2) shows how cosine similarity is calculated.

$$\text{sim}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)| |\vec{V}(d_2)|} \quad (4.2)$$

Another part of the similarity measurement is the weight vector. Depending on the weights, document similarities change. After creating a weight vector which is of the same length as the feature vector, it is multiplied with the feature vector of the document, as can be seen in Figure 4.3.

Obviously, there is no clear guideline for the value of the weight for each feature. Also, the tool will be implemented for two different languages, English and German, where it is likely that weightings will vary according to the selected language. Highly inflected languages, such as German, will probably need lower weights for the position feature, while

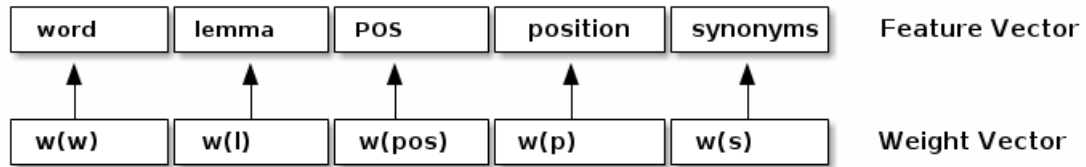


Figure 4.3: Applying the weight vector to the feature vector

word position will play a major role in low inflected languages. Therefore, two different trainings would have to be carried out, until the optimal weight is found for each of the features in the respective language.

4.2 Implementation

Following the methods explained in the previous section, text pre-processing has been applied on the two files. The following section is structured as follows. Subsection 4.2.1 will demonstrate how the feature vector has been built up, Subsection 4.2.2 explains how the features then were extracted to fill the vector. Last, Subsection 4.2.3 will show how the weight vector has been implemented.

4.2.1 Structure of the Feature Vector

As most of the existing tools fail to detect similarities as soon as the word order is slightly changed, this thesis will expand on a BOW representation of the documents. In the BOW representation, the word order is neglected, while the number of occurrences of a word is preserved¹ [18]. Equation (4.3) shows the simplest vector representation of a collection of documents. The words of all documents are stored in the columns, each column refers to one word (w_1 to w_n) and reflects the number of occurrences, while each column represents one document for all documents d_1 to d_m .

$$\text{BOW} = \begin{matrix} & w_1 & w_2 & w_3 & \dots & w_n \\ \begin{matrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_m \end{matrix} & \begin{pmatrix} 1 & 1 & 1 & \dots & 5 \\ 0 & 3 & 0 & \dots & 1 \\ 0 & 0 & 2 & \dots & 1 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \end{matrix} \quad (4.3)$$

However, we will have to extend the matrix in Equation (4.3), such that all the selected features can be stored. Equation (4.4) shows how the matrix of a document d_1 would look, storing all the relevant features. The first part of the equation represents the verbose matrix, while the second part expresses the binary transformation of the information. The first digit in the word vector stands for “nicer”, while the binary value represents the presence of absence of this word in this document. The length of the word vector

¹As opposed to Boolean Retrieval, where the vector stores only 0 or 1, depending if a word exists in a document or not

is therefore equal to the number of different words in the two documents. The second part represents the lemmas in the texts. Similarly to the word vector, each digit stands for the presence of a certain lemma, and the length of the vector is again the number of lemmas in both documents. However, this vector is probably shorter than the other one, as multiple words share the same lemma. The remaining digits for POS tag, word position, and synonyms work in the same manner.

$$d_1 = \begin{matrix} & \begin{matrix} word & lemma & pos & \dots & syn \end{matrix} \\ \begin{matrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{matrix} & \begin{pmatrix} nicer & nice & JJ & \dots & kind \\ grass & grass & NN & \dots & green \\ \vdots & \vdots & \vdots & \dots & \vdots \\ walker & walk & NN & \dots & go \end{pmatrix} \end{matrix} = \begin{matrix} & \begin{matrix} word & lemma & pos & syn \end{matrix} \\ & \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \end{matrix} \quad (4.4)$$

The problem with these extended word information is that, if we nest them into a matrix that stores the information of each of the documents, the vector is not at one level any more. This means the document feature vectors cannot be compared.

The idea is to calculate an average over all words from the document vector, such that the dimension is decreased and the two vectors can be compared. The example of an average vector can be seen in Equation (4.5). The decimal number shows the proportion in which a certain word, lemma, POS tag etc. occurs in a document. Here again the length of each feature reflects the representation of this feature in the document. The documents in the equation consist therefore of three different words, three different lemmas, two POS tags, and so on.

$$\begin{matrix} & \begin{matrix} word & lemma & pos & position & syn \end{matrix} \\ d_1 & \left(\begin{matrix} 0.33 & 0.33 & 0.33 & 0.33 & 0.33 & 0.33 & 0.33 & 0.66 & \dots & 0.33 & 0.33 & 0.33 \end{matrix} \right) \\ d_2 & \left(\begin{matrix} 0.66 & 0.00 & 0.33 & 0.66 & 0.00 & 0.33 & 0.66 & 0.33 & \dots & 0.66 & 0.00 & 0.33 \end{matrix} \right) \end{matrix} \quad (4.5)$$

Provided that there are two documents in Equation (4.5), each consisting of three words w_1 to w_3 , we can deduce from Vector 1 that each of the three words in the corpus occurs exactly once. Vector 2 shows that w_1 occurs twice, w_2 is absent in this document, and w_3 occurs once. This works equivalently for the remaining features.

4.2.2 Feature Extraction

The last subsection discussed the structure of storing the features. Next, these features will have to be extracted from the two input files. After the pre-processing steps presented in Subsection 4.1.1, tokens have already been extracted and can be stored in the feature vector. The following paragraphs will explain, how the remaining features, POS tag, lemma, synonyms, and position, have been extracted and stored.

As discussed in Section 4.1.2, it is of major interest to include the POS tag in the feature vector, as this helps to distinguish words that have the same spelling but a different meaning and POS tag. To extract these lemmas, along with their respective POS tag, the *TreeTagger* will be used. Other than a standard trigram tagger, the *TreeTagger* achieves a higher accuracy, even with a lower document size. Furthermore, the tagger does not influence the time efficiency of the program, as it is able to tag up to 10,000 tokens per second [30]. A sample output can be seen in Table 4.2, taken from the *TreeTagger* website².

²<http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/>

word	pos	lemma
The	DT	the
TreeTagger	NP	TreeTagger
is	VBZ	be
easy	JJ	easy
to	TO	to
use	VB	use
.	SENT	.

Table 4.2: Sample TreeTagger output, taken from the TreeTagger website

The TreeTagger was developed by Helmut Schmid in the Text Corpora project at the Institute for Computational Linguistics of the University of Stuttgart [29, 30]. The TreeTagger is a n-gram tagger that models the probability of a tagged sequence of words. The transition probabilities are estimated with a binary decision tree. In each step, the test yielding the most information is attached to the current node of the tree. This tree is expanded recursively until the node with the highest probability reveals the category of the word [30]. Figure 4.4 shows an example of an excerpt of a binary decision tree. Suffix and prefix lexicons are used to classify unknown words.

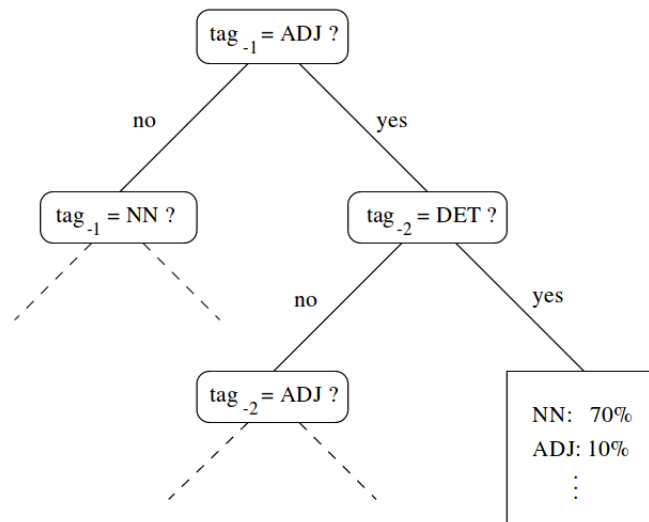


Figure 4.4: Sample tree of the TreeTagger, as illustrated by Schmid [30]

For the retrieval of synonyms, the lexical databases GermaNet and WordNet are used to store the German and English synonyms, respectively. Storing synonyms in the feature vector turned out to be not as straightforward as the other features. The first idea was to store each synonym as a feature, such that for the word *small*, the synonyms are $Syn_1 = little$, $Syn_2 = minor$ etc. Also, the word itself must be stored as a synonym as this is crucial for comparing the vectors: $Syn_0 = small$. It must also be taken into consideration that words have different sets of synonyms in WordNet and GermaNet, even if they are synonyms of each other. The synonym vectors for the two words above are given in Equation (4.6) and displayed in Figure 4.5 .

$$\begin{array}{l}
 \begin{array}{ccc}
 & \textit{syn}_0 & \textit{syn}_1 & \textit{syn}_2 \\
 \text{small} & (\text{small} & \text{little} & \text{minor}) \\
 \text{little} & (\text{little} & \text{small} & \text{humble})
 \end{array}
 \rightarrow w_2 \begin{array}{ccc}
 & \textit{syn}_0 & \textit{syn}_1 & \textit{syn}_2 \\
 (1 & 0 & 1 & 0) \\
 (0 & 1 & 0 & 1)
 \end{array}
 \end{array}
 \quad (4.6)$$

As can be seen, the vectors do not share any common parts, because of their different word ordering. This is why synonyms cannot be stored in this way but must be implemented differently.

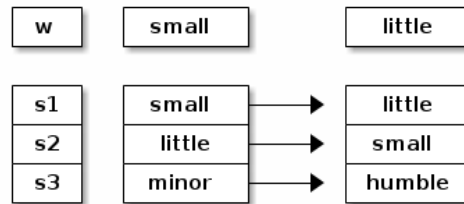


Figure 4.5: Storing synonyms in separate list entries

As the vectors are built from dictionaries, another idea would be to give the dictionary an ordered list of synonyms as a value, and *synonyms* as a key. This would be implemented as follows:

```
dictionary =
{word: little, pos: JJ, ..., synonyms: [humble, little, small...]}
```

This would work if the tool that is used to transform the dictionaries into vectors had the possibility to transform arrays into vectors. Unfortunately, with the used tool, DictVectorizer from scikit-learn³, only non-embedded structures can be transformed to binary values.

The idea of the algorithm is to store all synonyms that belong together in a list and embed these small lists into a bigger one, that then stores all synonyms of all words in the documents. The lists themselves are sorted alphabetically, such that the first element remains the same if we search these lists. The function checks the words in the existing synonym list first, before creating a new synonym set. When creating the synonym feature for the vector, the algorithm then searches in which list the current word appears and returns the first element of the synonym set. This word represents the whole list, that is, it is unambiguous to which list it refers and there is no need to store the whole list in the dictionary.

The transformation with DictVectorizer is now possible, because the list is not embedded any more. The first document in the example below contains the words *small* and *nice*, while the second one only contains the word *little*. The synonyms have been calculated as above, and the first synonym is stored as representative. The following sample code shows the transformation of two documents:

³scikit-learn is a ML tool for Python, which provides useful modules for data mining and data analysis. http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.DictVectorizer.html

```

>>> Document = 'nice small minor'
>>> Synonyms = [
...   [courteous, dainty, decent, nice],
...   [belittled, humble, little, minor, modest, small]
... ]
>>> create_dictionary()
[ {word: small, pos: JJ, lemma: small, Syns: belittled},
  {word: nice, pos: JJ, lemma: nice, Syns: courteous},
  {word: minor, pos: JJ, lemma: minor, Syns: belittled} ]

```

The synonym list has a nested structure, as it contains lists of all synonyms of the documents to be compared. Every time a new word is read in, these subordinate lists will have to be searched in order to find if the word or one of its synonyms already exists. Then for each vector, the algorithm will have to iterate again over all lists in order to identify the correct synonym list. However, this search is too time-expensive, especially as the documents exceed a small number of elements. Time was not a problem during the experiments, as most of the example documents contained only one or a couple of short sentences. As the two files get longer, searching a list of lists becomes too time complex. To iterate over a list of lists has the time complexity of $\mathcal{O}(n^2)$, compared to a hash (dictionary) structure with a complexity of $\mathcal{O}(n)$. Figure 4.6 shows the comparison of the two structures (adapted from: <https://stackoverflow.com/questions/487258/>).

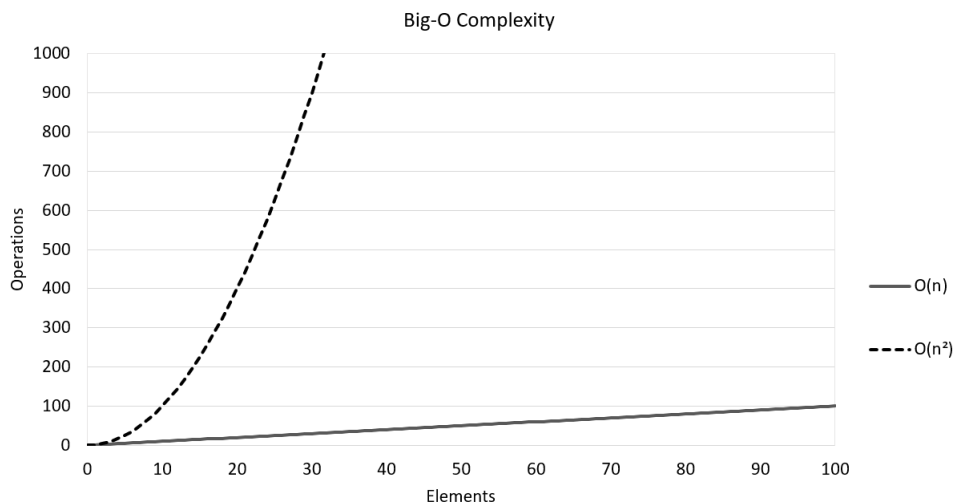


Figure 4.6: Time complexity of $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$

Now that all synonyms are stored in a less complex structure, time complexity is reduced to $\mathcal{O}(n)$. The first element in each hash entry can then be stored as the synonym in the feature vector (see Figure 4.7). Then DictVectorizer transforms the dictionary into a vector as can be seen below:

```

def vectorize(dictionary):
    vec = DictVectorizer()
    pos_vectorized = vec.fit_transform(dictionary).toarray()
    return pos_vectorized
>>>vectorize(dictionary)
doc1 [

```

```
[ 1.  0.  0.  0.  1.  1.  0.  0.  1.]
[ 0.  1.  0.  1.  0.  1.  0.  1.  0.] ]
doc2 [
[ 1.  0.  1.  0.  0.  1.  1.  0.  0.] ]
```

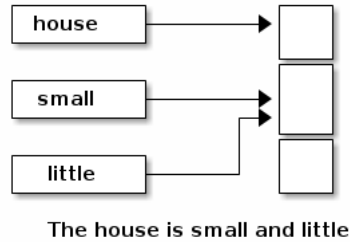


Figure 4.7: Incorporating words in a hash structure, where synonyms point to the same entry.

Finally, to integrate the position in the feature vector, it is not possible to include only a dictionary entry that has a key “position” and the value of the word position in the sentence. As with the used method, the positions would all have the value $\frac{1}{N}$ where N is the total number of positions in all documents. The solution is not only to store the position number, but to concatenate it with the word on that position in the format *word.position*. The output in Example 4.7 indicates that, in the first sentence, the lemma *this* occurs at the first position and stores a different word position feature (*this.0*) than the same lemma in the second sentence (*this.1*).

- (a) this is a small house
 this.0 is.1 a.2 small.3 house.4
- (b) is this a small house
 is.0 this.1 a.2 small.3 house.4 (4.7)

4.2.3 Feature Weighting

As explained in Section 4.1.3, not all features are equally important. Two texts may share the same POS tags, while their content is completely different. This indicates that POS should be assigned a lower weight than the word itself. On the other hand, words in paraphrased texts may have another position in the sentence, while still expressing the same meaning.

To create a weight vector, the same weight for “word” has been stored according to the number of digits that represent the words themselves in the word vector. Then a weight for all lemmas has been added to the weight vector, and so forth. The weight vector for a file containing five different words, four lemmas, and four POS tags, could look as follows:

```
[ 2. , 2. , 2. , 2. , 1. , 1. , 1. , 1. , 0.5, 0.5, 0.5,
  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 1. , 1. , 1. , 1. , 1. ]
```

So far, weights have to be adjusted manually with default values lemma=2, POS=1, position=0.5, synonyms=0.5, word=1. The values sum up to five, in agreement with the number of features. However, it is possible to assign higher values to shift the result and

give more weight to considerable features. In the developed tool, it is possible to assign these weights over the command line in order to test if new weights perform better than the default values. The weight vector is multiplied with the feature vector before calculating the cosine similarity between the two document vectors.

Due to the time limitation of this thesis, weights will have to be adjusted manually and an accurate weight found through human intuition. It is left for future work (see Chapter 5) to train a ML system in order to get a perfectly accurate feature vector.

4.3 Evaluation

After processing the documents and retrieving the two feature vectors and the weight vector, the program can be evaluated. Therefore, the same data that has already been utilised for the evaluation of the existing tools in Chapter 3 will be used and the outcome evaluated. Subsection 4.3.1 will discuss the expected results, while Subsection 4.3.2 will evaluate actual results and compare them to those of existing software.

4.3.1 Expected Results

The new hybrid approach combines features and methods from existing work discussed in Chapters 2 and 3. This is why we would expect the tool to outperform existing methods, improving their attained F-score. In general, two files with the same meaning should have a high similarity measure, while files that share nothing in common should score a measure near zero. The new system should moreover be resistant to transformed sentences, a slight change in sentence structure and replacement of words with their synonyms. We would also expect different results for different languages. This is because the wrong language setting will remove the stop words belonging to another language and POS tagging will not work. For this reason, inappropriate language settings should yield poorer results. Lastly, translated plagiarism will very likely not be detected, as the tool neither works with bilingual dictionaries nor with other translation systems. Actual results will be discussed in Subsection 4.3.2.

4.3.2 Results and Evaluation

The hybrid approach was able to significantly expand the state of the art compared to web-based detection systems. Overall results could be improved by an F-measure of 0.15. The system has been compared to the three best performing tools, CopyLeaks, Plagiarisma, and Plagium, that were evaluated in Section 3.4. Evaluation results, including the new approach, are shown in Table 4.3.

F1 value	Exact	Paraphrased	Translated	SEO Rewriter	Overall
CopyLeaks	0.97	0.77	0.00	0.00	0.43
Plagiarisma	1.00	0.67	0.55	0.22	0.61
Plagium	1.00	1.00	0.12	0.00	0.53
Hybrid Approach	1.00	0.97	0.11	0.95	0.76

Table 4.3: Comparison of the new approach with the best performing tools

These results show that in order to detect any kind of disguised similarity, the consideration of several features is reasonable. Including word position, the new approach detects changes in sentence structure and still yields high similarity values. Considering word

synonyms, the system is able to reveal a high overlap between the original and the text that has been transformed with the Small SEO Article Rewriter. Without this feature, the tool would not have attained as accurate results as the outcomes of existing tools demonstrate.

Exact Copy Evaluation

For exact copy similarity, the developed tool manages to achieve a F-score of 1.00, that means, all the sentences have been classified correctly. The system creates two identical vectors which have a cosine distance of zero and hence a similarity value $sim(d_1, d_2) = 1$. Table 4.4 illustrates an extract of the two equivalent vectors, one of TS 207 and the other of its copy.

Document 1	Document 2
[[1., 0., 0., ..., 0., 0., 0.],	[[1., 0., 0., ..., 0., 0., 0.],
[0., 1., 0., ..., 0., 0., 0.],	[0., 1., 0., ..., 0., 0., 0.],
[0., 0., 1., ..., 0., 0., 0.],	[0., 0., 1., ..., 0., 0., 0.],
...	...
[0., 0., 0., ..., 1., 0., 0.],	[0., 0., 0., ..., 1., 0., 0.],
[0., 0., 0., ..., 0., 1., 0.],	[0., 0., 0., ..., 0., 1., 0.],
[0., 0., 0., ..., 0., 0., 1.]]	[0., 0., 0., ..., 0., 0., 1.]]

Table 4.4: Vector comparison for two identical documents

Paraphrased Evaluation

The extract that has been paraphrased or slightly adapted still yields an accurate similarity score of 0.97. The result refers to the percentage of similar content, which means it is even desirable not to achieve 1.00, as the two files are not identical and the score would be incorrect. The tool still assigns a considerably high similarity value, because it includes the synonyms and lemmas of the words. Table 4.5 shows the two document vectors of the original (Document 1) and paraphrased (Document 2) extract.

Document 1	Document 2
[[0., 1., 0., ..., 0., 0., 0.],	[[1., 0., 0., ..., 0., 0., 0.],
[0., 0., 1., ..., 0., 0., 0.],	[0., 0., 1., ..., 0., 0., 0.],
[0., 0., 0., ..., 0., 0., 0.],	[0., 0., 0., ..., 0., 0., 0.],
...	...
[0., 0., 0., ..., 0., 0., 0.],	[0., 0., 0., ..., 1., 0., 0.],
[0., 0., 0., ..., 0., 0., 0.],	[0., 0., 0., ..., 0., 1., 0.],
[0., 0., 0., ..., 1., 0., 0.]]	[0., 0., 0., ..., 0., 0., 1.]]

Table 4.5: Vector comparison with the paraphrased document

Translated Evaluation

As expected, the new approach yields poor results in terms of translated document similarity. Results rate the translated and the original remark to be 11% similar. This can be explained through the fact that most of the POS tags may be similar in both languages, while the other four features, word, lemma, synonyms, and word position, are different. Without including any dictionary or translation systems, translated similarities can obviously not be detected.

Small SEO Evaluation

There are significant improvements in revealing parallelism between the original text and the text that has been transformed with the Small SEO Article Rewriter. While existing systems yield scores between 0.00 and 0.22, the new tool detects a similarity of 0.95, improving results by 0.73. This can be explained by the fact that the SEO tool replaces words with their synonyms. Current systems find similar structures in function words other than content words, and fail to draw a parallel between the two documents. The hybrid approach, however, takes synonyms into account and can detect the similarities between the two texts accordingly.

Evaluation of Document Similarity

The tool improves results a little, surpassing Plagiarisma by 5.21% and Plagium and CopyLeaks by 30.8% and 39.84%, respectively. Again, the most significant changes can be found in the text rewritten by Small SEO, 34.08%. This can again be explained through the fact that freely available tools do not include semantic word information in their algorithms.

	Paraphrased	Translated	SEO Rewriter	Overall
CopyLeaks	65.40%	0.00%	18.10%	27.83%
Plagiarisma	81.43%	45.03%	60.92%	62.46%
Plagium	55.40%	0.00%	55.20%	36.87%
Hybrid Approach	97.00%	11.00%	95.00%	67.67%

Table 4.6: Comparison with tools that compare two documents

Comparing again two documents that are not adapted for research purposes, Wittgenstein’s TS 213, and MS 114 (see again Table 3.8 on page 17) will be analysed with the new tool. Table 4.7 shows that the tool outperforms existing systems.

	% detected
CopyLeaks	23
Plagiarisma	85
Plagium	75
Hybrid Approach	94

Table 4.7: Similarity between Wittgenstein’s TS 213 and MS 114 [26]

These results show a similarity of the files of 94%, which is an expected result. The files are slightly different and should not have scored a similarity of 100%. This would be the case if two identical files were compared. This means the tool could provide accurate results and outperform existing tools that compare two documents against each other. A complete sample output of the two files can be found in the Appendix, Chapter D.

5 Conclusions and Future Work

In this work, a new hybrid approach in the field of similarity detection has been presented. Semantic-based and syntactic-based approaches have been combined together with common NLP-methods. Two documents have been pre-processed, before extracting five different features and creating a weight vector to be multiplied with the feature vectors. Combining all features, including synonyms and word position, expands the state of the art methods in similarity detection.

A summary of the new approach will be presented in Section 5.1. However, similarity detection remains a complex field, and the synonym hash, as well as the weight vector are still to be improved. Section 5.2 will give an overview of future work.

5.1 Conclusions

A new hybrid approach has been developed, combining methods that depend on words and word positions, and methods that only use synonyms. Five different features have been extracted, namely the word itself, lemma, POS-tag, word position, and synonyms. These features have been stored in a feature vector for each document and been multiplied with a weight vector. Last, the cosine similarity between the two documents has been calculated. Results have shown that the combination of existing methods leads to more accurate results, improving the F-measure by 0.15 compared to existing tools. The tool has performed well with exactly copied similarity, paraphrased similarity, and a text that has been modified with the Small SEO Article Rewriter, where all words have been replaced by their synonyms. The hybrid method yields a similarity score of 0.97 in these three subjects. The overall result of 0.76 can be explained by the fact that no dictionaries or translation systems have been used. Translated passages are hence not classified as similar, yielding a similarity of 0.11.

Overall, the tool could improve the state of the art significantly, improving the F-score from 0.67 to 0.97 in three of the categories. To conclude, hybrid approaches in the field of similarity detection yield better results, as they consider more different features than approaches that solely focus on either syntactic or semantic features.

5.2 Future Work

A lot has already been done, as summarised in Section 5.1. However, due to time limitations, the weights of the features still have to be adapted manually and optimised with experiments and subjective rating of the similarity of two files. It remains to optimise the weight vector by means of ML methods, where the weight vector is trained on hand-labelled training data. The ML algorithm should then learn the weights automatically and further yield more precise results.

Secondly, the synonym hash has to be optimised. Currently, synonyms are stored in a hash, and words with the same meaning point to the same entry. These synonym entries can be extended, such that all synonyms of a word are classified correctly. WSD methods

would also help to retrieve the correct synonyms of a word.

Furthermore, the feature set could be expanded. Incorporating MT systems will help to identify similarities across documents of different languages. A possibility would be to include the Operation Sequence Model, as presented by Schütze et al. [31]. This model combines n-gram and phrase-bases Statistical MT. By combining the two approaches, the system considers all possible reorderings and achieves to correctly reorder words across large distances. This translation system yields better accuracy and translation quality compared to other systems, which could be useful in determining similarities as precisely as possible.

Moreover, the relation between words could be taken into account. A combination of convolutional and recurrent neural networks for relation classification is described in Schütze et al. [32]. Recurrent neural networks compute a weighted combination of all words in a sentence, while convolutional neural networks only extract the most informative n-grams for the relation. Their features complement each other, which is why combining the two networks improves existing techniques. Integrating this information about the relation of the words will then further achieve more accurate results in the field of document similarity extraction.

Finally, the tool has only been tested on Wittgenstein's remarks and variations of it. Therefore, it would be interesting to see how the tool performs on another type of data. It is left to future work to apply the system to different data. A collaborative documentation of plagiarism that could be used for this is, for instance, the GuttenPlag Wiki¹. The Wiki deals critically with the plagiarised content of the doctoral dissertation of the former German Minister of Defence, Karl-Theodor Freiherr zu Guttenberg. The dissertation along with original sources could be taken as new evaluation data of the similarity detection system.

¹http://de.guttenplag.wikia.com/wiki/GuttenPlag_Wiki

List of Abbreviations

BOW	Bag of Words. 25, 26
Cmap	Character map. 18
IR	Information Retrieval. 3, 25
ML	Machine Learning. 11, 29, 32, 35
MS	Manuscript. 15, 17, 34, 55
MT	Machine Translation. 3, 36
NLP	Natural Language Processing. I, 1, 3, 19, 21, 35
OCR	Optical Character Recognition. 13, 19
PAN	Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection. 7
PD	Plagiarism Detection. 7–10, 12, 15, 17, 18
POS	Part-of-Speech. I, 1, 3, 4, 11, 21–24, 27, 31–33, 35
TF-IDF	Term Frequency-Inverse Document Frequency. 4
TS	Typescript. 14–17, 33, 34, 41–44, 55
VSM	Vector Space Model. I, 4, 10
WSD	Word Sense Disambiguation. 3, 5, 35

Appendices

A Used Texts

Ladies and Gentlemen, Before I begin to speak about my subject proper let me make a few introductory remarks. I feel I shall have great difficulties in communicating my thoughts to you and I think some of them may be diminished by mentioning them to you beforehand. The first one, which almost I need not mention, is, that English is not my native tongue and my expression therefore often lacks that precision and subtlety which would be desirable if one talks about a difficult subject. All I can do is to ask you to make my task easier by trying to get at my meaning in spite of the faults which I will constantly be committing against the English grammar. The second difficulty I will mention is this, that probably many of you come up to this lecture of mine with slightly wrong expectations. And to set you right in this point I will say a few words about the reason for choosing the subject I have chosen: When your former secretary honoured me by asking me to read a paper to your society, my first thought was that I would certainly do it and my second thought was that if I was to have the opportunity to speak to you I should speak about something which I am keen on communicating to you and that I should not misuse this opportunity to give you a lecture about, say, logic. I call this a misuse for to explain a scientific matter to you it would need a course of lectures and not an hour's paper. An other alternative would have been to give you what's called a popularscientific lecture, that is a lecture intended to make you believe that you understand a thing which actually you don't understand, and to gratify what I believe to be one of the lowest desires of modern people, namely the superficial curiosity about the latest discoveries of science. I rejected these alternatives and decided to talk to you about a subject which seems to me to be of general importance, hoping that it may help to clear up your thoughts about this subject (even if you should entirely disagree with what I will say about it). My third and last difficulty is one which, in fact, adheres to most lengthy philosophical lectures and it is this, that the hearer is incapable of seeing both the road he is led and the goal which it leads to. That is to say: he either thinks: "I understand all he says, but what on earth is he driving at" or else he thinks "I see what he's driving at, but how on earth is he going to get there". All I can do is again to ask you to be patient and to hope that in the end you may see both the way and where it leads to.

Table A.1: Wittgenstein's TS 207 - original version

Dear Ladies and Gentlemen, Before I begin to talk about my subject proper, I want to make a few introductory remarks. I feel I will have big problems in communicating my thoughts to you and I think some of them may be diminished by mentioning them to you in advance. First of all, which almost I need not mention, is, that English is not my mother language and my expression therefore often lacks that precision and subtlety which would be of importance if one talks about a difficult subject. All I can do is to kindly ask you to make my task easier by trying to get at my point in spite of the faults which I will constantly be committing against the English grammar. The second difficulty I will mention is this, that probably many of you come up to this lecture of mine with false expectations. And to make it clear at this point I will say a few words about the reason for choosing the chosen subject: When your former secretary honoured me by asking me to read a paper to you, my first thought was that I would surely do it and my second thought was that if I was to have the chance to talk to you I should speak about something which I am keen on telling to you and that I should not misuse this chance to give you a lecture about, for instance, logic. I call this a misuse for to explain a scientific topic to you it would need a couple of lectures and not an hour's paper. Another alternative would have been to give you what is called a popularscientific talk, that is a lecture intended to make you think that you comprehend a thing which actually you don't comprehend, and to gratify what I think to be one of the lowest desires of modern people, namely the superficial curiosity about the latest discoveries of science. I rejected these alternatives and decided to give a lecture about a subject which seems to me to be of general importance, hoping that it may help to understand this subject (even if you should entirely disagree with what I will say about it). My last problem is one which, to be honest, adheres to most lengthy philosophical lectures and it is this, that the hearer is incapable of seeing both the road he is led and the goal which it leads to. That means: he either believes: "I comprehend all he says, but what in blazes is he driving at" or else he thinks "I see what he's driving at, but how in blazes is he going to get to this". All I can do is once again to ask you to be patient and to hope that in the end you may see both the way and where it leads to.

Table A.2: Wittgenstein's TS 207 - paraphrased version

Meine Damen und Herren, bevor ich anfangen, über mein Thema zu sprechen, lassen Sie mich ein paar einleitende Bemerkungen machen. Ich glaube, ich werde große Schwierigkeiten haben, meine Gedanken an Sie zu vermitteln, und ich denke, dass einige von ihnen vermieden werden können, indem ich sie Ihnen gegenüber im Voraus erwähne. Das erste, das ich fast nicht erwähnen muss, ist, dass Englisch nicht meine Muttersprache ist und es meinem Ausdruck also oft der Präzision und Feinheit fehlt, welche wünschenswert wäre, wenn man über ein schwieriges Thema spricht. Alles, was ich tun kann, ist Sie zu bitten, meine Aufgabe zu erleichtern, indem Sie versuchen, die Bedeutung trotz der Fehler zu verstehen, die ich ständig gegen die englische Grammatik begehen werde. Die zweite Schwierigkeit, die ich erwähnen werde, ist, dass wahrscheinlich viele von Ihnen zu dieser Vorlesung von mir mit etwas falschen Erwartungen kommen. Und um Sie in diesem Punkt richtig zu stellen, werde ich ein paar Worte über den Grund für die Wahl des Themas, das ich gewählt habe, sagen: Als Ihre ehemalige Sekretärin mich geehrt hat, indem Sie mich bat, eine Vorlesung in Ihrer Gesellschaft zu halten, war mein erster Gedanke, dass ich sicherlich tun würde und mein zweiter Gedanke war, dass ich, wenn ich die Gelegenheit hätte, mit Ihnen zu sprechen, über etwas sprechen sollte, von dem ich begeistert bin Ihnen zu vermitteln, und dass ich diese Gelegenheit nicht missbrauchen sollte, um einen Vortrag über etwa die Logik zu halten. Ich nenne dies einen Missbrauch, um Ihnen eine wissenschaftliche Angelegenheit zu erklären, würde man eine Vortragsreihe und nicht ein einstündigen Vortrag brauchen. Eine andere Alternative wäre es gewesen, Ihnen zu geben, was man einen populärwissenschaftlichen Vortrag nennt, das ist ein Vortrag, der dazu bestimmt ist, Sie glauben zu lassen, dass Sie eine Sache verstehen, die Sie eigentlich nicht verstehen, und um, was ich glaube, einen der niedrigsten Wünsche der modernen Menschen, nämlich die oberflächliche Neugier auf die neuesten Entdeckungen der Wissenschaft, zu befriedigen. Ich habe diese Alternativen abgelehnt und beschlossen, mit Ihnen über ein Thema zu sprechen, das mir von allgemeiner Bedeutung erscheint, in der Hoffnung, dass es Ihnen helfen kann, Ihre Gedanken über dieses Thema zu entwirren (auch wenn Sie dem was ich sage vielleicht auch vollständig widersprechen). Meine dritte und letzte Schwierigkeit ist eine, die in der Tat an den meisten philosophischen Vorträgen haftet, und das ist es, was der Hörer nicht in der Lage ist, sowohl den Weg zu sehen, der ihn führt, als auch das Ziel, zu dem er führt. Das heißt: er denkt entweder: "Ich verstehe alles, was er sagt, aber worauf in aller Welt will er hinaus" oder er denkt "ich sehe, worauf er hinaus will, aber wie in aller Welt wird er dorthin kommen". Alles was ich tun kann ist wieder, Sie zu bitten, geduldig zu sein und zu hoffen, dass Sie am Ende vielleicht beides sehen, den Weg und wohin er führt.

Table A.3: Wittgenstein's TS 207 - translated version

Ladies and Gentlemen, Before I begin to talk concerning my subject proper let me build a couple of introductory remarks. I feel I shall have great difficulties in communicating my thoughts to you and that i assume a number of them could also be diminished by mentioning them to you beforehand. the primary one, which just about i would like not mention, is, that English isn't my native tongue and my expression so typically lacks that exactness and subtlety which might be desirable if one talks about a tough subject. All I will do is to raise you to make my task easier by making an attempt to induce at my meaning in spite of the faults that i'll perpetually be committing against English grammar. The second problem i'll mention is this, that in all probability several of you come back up to the current lecture of mine with slightly wrong expectations. And to line you right during this purpose i'll say many words concerning the rationale for selecting the topic I actually have chosen: once your former secretary honored me by asking me to read a paper to your society, my initial thought was that i'd definitely do it and my second thought was that if i was to have the chance to talk to you I ought to speak concerning one thing that i'm keen on communicating to you which I mustn't misuse this chance to allow you a lecture about, say, logic. I call this a misuse for to clarify a scientific refer you it'd need a course of lectures and not an hour's paper. an other alternative would have been to allow you what is known as a popularscientific lecture, that's a lecture supposed to make you think that you just perceive a issue that truly you do not perceive, and to gratify what i think to be one amongst the lowest wishes of contemporary individuals, particularly the superficial curiosity concerning the newest discoveries of science. I rejected these alternatives and determined to speak to you a few subject that seems to me to be of general importance, hoping that it's going to facilitate to clear up your thoughts concerning this subject (even if you must entirely trouble what i'll say concerning it). My third and last problem is one that, in fact, adheres to most prolonged philosophical lectures and it's this, that the perceiver is incapable of seeing each the road he's led and therefore the goal that it results in. that's to say: he either thinks: "I perceive all he says, however what on earth is he driving at" as an alternative he thinks "I see what he is driving at, however how on earth is he reaching to get there". All I will do is once more to raise you to be patient and to hope that within the end you will see each the approach and where it results in.

Table A.4: Wittgenstein's TS 207 - Small SEO rewritten version

B Detailed Results

Exact Copy	TP	TN	FP	FN	Recall	Precision	F ₁
Academic Plagiarism	13	0	0	3	0.8125	1	0.8965517241
CopyLeaks	15	0	0	1	0.9375	1	0.9677419355
Plagiarisma	16	0	0	0	1	1	1
Plagium	16	0	0	0	1	1	1
PlagTracker	16	0	0	0	1	1	1
Small Seo	11	0	0	5	0.6875	1	0.8148148148

Table B.1: Exact copy results

Paraphrased	TP	TN	FP	FN	Recall	Precision	F ₁
Academic Plagiarism	1	0	0	15	0.0625	1	0.1176470588
CopyLeaks	10	0	0	6	0.625	1	0.7692307692
Plagiarisma	8	0	0	8	0.5	1	0.6666666667
Plagium	16	0	0	0	1	1	1
PlagTracker	0	0	0	16	0	1	0
Small Seo	5	0	0	11	0.3125	1	0.4761904762

Table B.2: Paraphrased results

Translated	TP	TN	FP	FN	Recall	Precision	F ₁
Academic Plagiarism	1	0	0	15	0.0625	1	0.1176470588
CopyLeaks	0	4	0	16	0	1	0
Plagiarisma	6	0	0	10	0.375	1	0.5454545455
Plagium	1	0	0	15	0.0625	1	0.1176470588
PlagTracker	0	0	0	16	0	1	0
Small Seo	0	0	0	16	0	1	0

Table B.3: Translated results

Small SEO	TP	TN	FP	FN	Recall	Precision	F ₁
Academic Plagiarism	1	0	0	15	0.0625	1	0.1176470588
CopyLeaks	0	0	0	16	0	1	0
Plagiarisma	2	0	0	14	0.125	1	0.2222222222
Plagium	0	0	0	16	0	1	0
PlagTracker	0	0	0	16	0	1	0
Small Seo	0	0	0	16	0	1	0

Table B.4: Small SEO rewritten results

Overall Results	Direct Copy	Paraphrased	Translated	Small SEO	Overall
Academic Plagiarism	0.90	0.12	0.12	0.12	0.31
CopyLeaks	0.97	0.77	0.00	0.00	0.43
Plagiarisma	1.00	0.67	0.55	0.22	0.61
Plagium	1.00	1.00	0.12	0.00	0.53
PlagTracker	1.00	0.00	0.00	0.00	0.33
Small Seo	0.81	0.48	0.00	0.00	0.33
Hybrid Approach	1.00	0.97	0.11	0.95	0.76

Table B.5: Overall results

C Lists of Stop Words

a, about, above, after, again, against, all, am, an, and, any, are, aren't, as, at, be, because, been, before, being, below, between, both, but, by, can't, cannot, could, couldn't, did, didn't, do, does, doesn't, doing, don't, down, during, each, few, for, from, further, had, hadn't, has, hasn't, have, haven't, having, he, he'd, he'll, he's, her, here, here's, hers, herself, him, himself, his, how, how's, i, i'd, i'll, i'm, i've, if, in, into, is, isn't, it, it's, its, itself, let's, me, more, most, mustn't, my, myself, no, nor, not, of, off, on, once, only, or, other, ought, our, ours, ourselves, out, over, own, same, shan't, she, she'd, she'll, she's, should, shouldn't, so, some, such, than, that, that's, the, their, theirs, them, themselves, then, there, there's, these, they, they'd, they'll, they're, they've, this, those, through, to, too, under, until, up, very, was, wasn't, we, we'd, we'll, we're, we've, were, weren't, what, what's, when, when's, where, where's, which, while, who, who's, whom, why, why's, with, won't, would, wouldn't, you, you'd, you'll, you're, you've, your, yours, yourself, yourselves

Table C.1: List of all English stop words

aber, alle, allem, allen, aller, alles, als, also, am, an, ander, andere, anderem, anderen, anderer, anderes, anderm, andern, anders, auch, auf, aus, bei, bin, bis, bist, da, damit, dann, das, dass, dasselbe, dazu, daß, dein, deine, deinem, deinen, deiner, deines, dem, demselben, den, denn, denselben, der, derer, derselbe, derselben, des, desselben, dessen, dich, die, dies, diese, dieselbe, dieselben, diesem, diesen, dieser, dieses, dir, doch, dort, du, durch, ein, eine, einem, einen, einer, eines, einig, einige, einigem, einigen, einiger, einiges, einmal, er, es, etwas, euch, euer, eure, eurem, euren, eurer, eures, für, gegen, gewesen, hab, habe, haben, hat, hatte, hatten, hier, hin, hinter, ich, ihm, ihn, ihnen, ihr, ihre, ihrem, ihren, ihrer, ihres, im, in, indem, ins, ist, jede, jedem, jeden, jeder, jedes, jene, jenem, jenen, jener, jenes, jetzt, kann, kein, keine, keinem, keinen, keiner, keines, können, könnte, machen, man, manche, manchem, manchen, mancher, manches, mein, meine, meinem, meinen, meiner, meines, mich, mir, mit, muss, musste, nach, nicht, nichts, noch, nun, nur, ob, oder, ohne, sehr, sein, seine, seinem, seinen, seiner, seines, selbst, sich, sie, sind, so, solche, solchem, solchen, solcher, solches, soll, sollte, sondern, sonst, um, und, uns, unse, unsem, unsen, unser, unses, unter, viel, vom, von, vor, war, waren, warst, was, weg, weil, weiter, welche, welchem, welchen, welcher, welches, wenn, werde, werden, wie, wieder, will, wir, wird, wirst, wo, wollen, wollte, während, würde, würden, zu, zum, zur, zwar, zwischen, über

Table C.2: List of all German stop words

D Sample Program Output

Text 1:

Augustinus beschreibt wirklich einen Kalkül; nur ist nicht alles, was wir Sprache nennen, dieser Kalkül. (Und das muß man in einer großen Anzahl von Fällen sagen, wo es sich fragt: ist diese Darstellung brauchbar oder unbrauchbar. Die Antwort ist dann: "ja, brauchbar; aber nur dafür, nicht für das ganze Gebiet, das Du darzustellen vorgabst".)

Text 2:

Augustinus beschreibt einen Kalkül unserer Sprache, nur ist nicht alles, was wir Sprache nennen, dieser Kalkül. (Und das muß man in vielen Fällen sagen, wo die Frage vor uns steht: "ist diese Darstellung brauchbar, oder unbrauchbar". Die Antwort ist: "ja, brauchbar, { aber nur dafür; nicht für das ganze Gebiet, das Du darzustellen vorgabst".) S. A

Tokenised Text 1:

('Augustinus beschreibt wirklich Kalkül Sprache nennen Kalkül muß großen Anzahl Fällen sagen fragt Darstellung brauchbar unbrauchbar Antwort ja brauchbar dafür ganze Gebiet darzustellen vorgabst',)

Tokenised Text 2:

('Augustinus beschreibt Kalkül unserer Sprache Sprache nennen Kalkül muß vielen Fällen sagen Frage steht Darstellung brauchbar unbrauchbar Antwort ja brauchbar dafür ganze Gebiet darzustellen vorgabst S A',)

Feature Vectors:

```
[ 'word': 'Antwort', 'lemma': 'Antwort', 'pos': 'NP', 'position': 'Antwort.16',
  'syns': 'Antwort', 'word': 'Anzahl', 'lemma': 'Anzahl', 'pos': 'NP', 'position':
  'Anzahl.9', 'syns': 'Anzahl', 'word': 'Augustinus', 'lemma': 'Augustinus', 'pos':
  'NP', 'position': 'Augustinus.0', 'syns': 'Augustinus', 'word': 'Darstellung',
  'lemma': 'Darstellung', 'pos': 'NP', 'position': 'Darstellung.13', 'syns':
  'Darstellung', 'word': 'Fällen', 'lemma': 'Fällen', 'pos': 'NP', 'position':
  'Fällen.10', 'syns': 'Fällen', 'word': 'Gebiet', 'lemma': 'Gebiet', 'pos': 'NP',
  'position': 'Gebiet.21', 'syns': 'Gebiet', 'word': 'Kalkül', 'lemma': 'Kalkül',
  'pos': 'NP', 'position': 'Kalkül.3', 'syns': 'Kalkül', 'word': 'Kalkül', 'lemma':
  'Kalkül', 'pos': 'NP', 'position': 'Kalkül.6', 'syns': 'Kalkül', 'word':
  'Sprache', 'lemma': 'Sprache', 'pos': 'NP', 'position': 'Sprache.4', 'syns':
  'Sprache', 'word': 'beschreibt', 'lemma': 'beschreibt', 'pos': 'NN', 'position':
  'beschreibt.1', 'syns': 'beschreibt', 'word': 'brauchbar', 'lemma': 'brauchbar',
  'pos': 'NP', 'position': 'brauchbar.14', 'syns': 'brauchbar', 'word':
  'brauchbar', 'lemma': 'brauchbar', 'pos': 'NN', 'position': 'brauchbar.18',
  'syns': 'brauchbar', 'word': 'dafür', 'lemma': 'dafür', 'pos': 'NN', 'position':
  'dafür.19', 'syns': 'dafür', 'word': 'darzustellen', 'lemma': 'darzustellen',
  'pos': 'NP', 'position': 'darzustellen.22', 'syns': 'darzustellen', 'word':
  'fragt', 'lemma': 'fragt', 'pos': 'NP', 'position': 'fragt.12', 'syns': 'fragt',
```

```

'word': 'ganze', 'lemma': 'ganze', 'pos': 'NN', 'position': 'ganze.20', 'syns':
'ganze', 'word': 'großen', 'lemma': 'großen', 'pos': 'NP', 'position': 'großen.8',
'syns': 'großen', 'word': 'ja', 'lemma': 'Ja', 'pos': 'NP', 'position':
'ja.17', 'syns': 'ja', 'word': 'muß', 'lemma': 'muß', 'pos': 'NP', 'position':
'muß.7', 'syns': 'muß', 'word': 'nennen', 'lemma': 'nennen', 'pos': 'NP',
'position': 'nennen.5', 'syns': 'nennen', 'word': 'sagen', 'lemma': 'Sagen',
'pos': 'NP', 'position': 'sagen.11', 'syns': 'sagen', 'word': 'unbrauchbar',
'lemma': 'unbrauchbar', 'pos': 'NP', 'position': 'unbrauchbar.15', 'syns':
'unbrauchbar', 'word': 'vorgabst', 'lemma': 'vorgabst', 'pos': 'NP', 'position':
'vorgabst.23', 'syns': 'vorgabst', 'word': 'wirklich', 'lemma': 'wirklich',
'pos': 'NP', 'position': 'wirklich.2', 'syns': 'wirklich', 'word': 'A', 'lemma':
'A', 'pos': 'NP', 'position': 'A.26', 'syns': 'A', 'word': 'Antwort', 'lemma':
'Antwort', 'pos': 'NP', 'position': 'Antwort.17', 'syns': 'Antwort', 'word':
'Augustinus', 'lemma': 'Augustinus', 'pos': 'NP', 'position': 'Augustinus.0',
'syns': 'Augustinus', 'word': 'Darstellung', 'lemma': 'Darstellung', 'pos': 'NP',
'position': 'Darstellung.14', 'syns': 'Darstellung', 'word': 'Frage', 'lemma':
'Frage', 'pos': 'NP', 'position': 'Frage.12', 'syns': 'Frage', 'word': 'Fällen',
'lemma': 'Fällen', 'pos': 'NP', 'position': 'Fällen.10', 'syns': 'Fällen',
'word': 'Gebiet', 'lemma': 'Gebiet', 'pos': 'NP', 'position': 'Gebiet.22',
'syns': 'Gebiet', 'word': 'Kalkül', 'lemma': 'Kalkül', 'pos': 'NP', 'position':
'Kalkül.2', 'syns': 'Kalkül', 'word': 'Kalkül', 'lemma': 'Kalkül', 'pos': 'NP',
'position': 'Kalkül.7', 'syns': 'Kalkül', 'word': 'S', 'lemma': 'S', 'pos': 'NP',
'position': 'S.25', 'syns': 'S', 'word': 'Sprache', 'lemma': 'Sprache', 'pos':
'NP', 'position': 'Sprache.4', 'syns': 'Sprache', 'word': 'Sprache', 'lemma':
'Sprache', 'pos': 'NP', 'position': 'Sprache.5', 'syns': 'Sprache', 'word':
'beschreibt', 'lemma': 'beschreibt', 'pos': 'NN', 'position': 'beschreibt.1',
'syns': 'beschreibt', 'word': 'brauchbar', 'lemma': 'brauchbar', 'pos': 'NP',
'position': 'brauchbar.15', 'syns': 'brauchbar', 'word': 'brauchbar', 'lemma':
'brauchbar', 'pos': 'NN', 'position': 'brauchbar.19', 'syns': 'brauchbar',
'word': 'dafür', 'lemma': 'dafür', 'pos': 'NN', 'position': 'dafür.20',
'syns': 'dafür', 'word': 'darzustellen', 'lemma': 'darzustellen', 'pos': 'NP',
'position': 'darzustellen.23', 'syns': 'darzustellen', 'word': 'ganze', 'lemma':
'ganze', 'pos': 'NN', 'position': 'ganze.21', 'syns': 'ganze', 'word': 'ja',
'lemma': 'Ja', 'pos': 'NP', 'position': 'ja.18', 'syns': 'ja', 'word': 'muß',
'lemma': 'muß', 'pos': 'NP', 'position': 'muß.8', 'syns': 'muß', 'word':
'nennen', 'lemma': 'nennen', 'pos': 'NP', 'position': 'nennen.6', 'syns':
'nennen', 'word': 'sagen', 'lemma': 'Sagen', 'pos': 'NP', 'position': 'sagen.11',
'syns': 'sagen', 'word': 'steht', 'lemma': 'steht', 'pos': 'NN', 'position':
'steht.13', 'syns': 'steht', 'word': 'unbrauchbar', 'lemma': 'unbrauchbar',
'pos': 'NP', 'position': 'unbrauchbar.16', 'syns': 'unbrauchbar', 'word':
'unserer', 'lemma': 'unserer', 'pos': 'NP', 'position': 'unserer.3', 'syns':
'unserer', 'word': 'vielen', 'lemma': 'vielen', 'pos': 'NP', 'position':
'vielen.9', 'syns': 'vielen', 'word': 'vorgabst', 'lemma': 'vorgabst', 'pos':
'NP', 'position': 'vorgabst.24', 'syns': 'vorgabst']

```

Feature Vector Document 1 (sentences):

```
[[ 0., 1., 0., ..., 0., 0., 0.], [ 0., 0., 1., ..., 0., 0., 0.], [ 0., 0., 0.,  
..., 0., 0., 0.], ..., [ 0., 0., 0., ..., 0., 0., 0.], [ 0., 0., 0., ..., 0., 1.,  
0.], [ 0., 0., 0., ..., 0., 0., 1.]]
```

Feature Vector Document 2 (sentences):

```
[[ 1., 0., 0., ..., 0., 0., 0.], [ 0., 1., 0., ..., 0., 0., 0.], [ 0., 0., 0.,  
..., 0., 0., 0.], ..., [ 0., 0., 0., ..., 0., 0., 0.], [ 0., 0., 0., ..., 1., 0.,  
0.], [ 0., 0., 0., ..., 0., 1., 0.]]
```

Feature Vector 1 (document):

```
[ 0. , 0.04166667, 0.04166667, 0.04166667, 0.04166667, 0. , 0.04166667,  
0.04166667, 0.04166667, 0.08333333, 0. , 0.04166667, 0.04166667, 0.04166667,  
0.08333333, 0.04166667, 0.04166667, 0.04166667, 0.04166667, 0.04166667,  
0.04166667, 0.04166667, 0. , 0.04166667, 0. , 0. , 0.04166667, 0.04166667,  
0.16666667, 0.83333333, 0. , 0.04166667, 0. , 0.04166667, 0.04166667, 0.04166667,  
0. , 0. , 0.04166667, 0.04166667, 0. , 0. , 0.04166667, 0.04166667, 0. , 0. ,  
0.04166667, 0. , 0.04166667, 0.04166667, 0. , 0.04166667, 0. , 0.04166667, 0.  
 , 0.04166667, 0. , 0.04166667, 0.04166667, 0. , 0.04166667, 0.04166667, 0. ,  
0.04166667, 0. , 0.04166667, 0. , 0.04166667, 0. , 0.04166667, 0. , 0. , 0. ,  
0.04166667, 0. , 0.04166667, 0. , 0.04166667, 0.04166667, 0.04166667, 0.04166667,  
0. , 0.04166667, 0.04166667, 0.08333333, 0. , 0.04166667, 0.04166667, 0.08333333,  
0.04166667, 0.04166667, 0.04166667, 0.04166667, 0.04166667, 0.04166667,  
0.04166667, 0.04166667, 0.04166667, 0. , 0. , 0.04166667, 0. , 0.04166667,  
0.04166667, 0. , 0.04166667, 0.04166667, 0.04166667, 0.04166667, 0. , 0.04166667,  
0.04166667, 0.08333333, 0. , 0.04166667, 0.04166667, 0.08333333, 0.04166667,  
0.04166667, 0.04166667, 0.04166667, 0.04166667, 0.04166667, 0.04166667,  
0.04166667, 0.04166667, 0. , 0.04166667, 0. , 0. , 0.04166667, 0.04166667]
```

Feature Vector 2 (document):

```
[ 0.03703704, 0.03703704, 0. , 0.03703704, 0.03703704, 0.03703704, 0.03703704,  
0.03703704, 0.03703704, 0.07407407, 0.03703704, 0.03703704, 0.07407407,  
0.03703704, 0.07407407, 0.03703704, 0.03703704, 0. , 0.03703704, 0. , 0.03703704,  
0.03703704, 0.03703704, 0.03703704, 0.03703704, 0.03703704, 0.03703704, 0. ,  
0.18518519, 0.81481481, 0.03703704, 0. , 0.03703704, 0. , 0.03703704, 0. ,  
0.03703704, 0.03703704, 0.03703704, 0. , 0.03703704, 0.03703704, 0. , 0. ,  
0.03703704, 0.03703704, 0.03703704, 0.03703704, 0.03703704, 0. , 0.03703704,  
0. , 0.03703704, 0. , 0.03703704, 0. , 0.03703704, 0. , 0. , 0.03703704, 0. ,  
0. , 0.03703704, 0. , 0.03703704, 0. , 0.03703704, 0.03703704, 0.03703704, 0. ,  
0.03703704, 0.03703704, 0.03703704, 0. , 0.03703704, 0. , 0.03703704, 0.03703704,  
0. , 0.03703704, 0.03703704, 0.03703704, 0.03703704, 0.03703704, 0.07407407,  
0.03703704, 0.07407407, 0.03703704, 0.07407407, 0.03703704, 0.03703704, 0. ,  
0.03703704, 0. , 0.03703704, 0.03703704, 0.03703704, 0.03703704, 0.03703704,  
0.03703704, 0.03703704, 0.03703704, 0.03703704, 0. , 0.03703704, 0.03703704,  
0. , 0.03703704, 0.03703704, 0.03703704, 0.03703704, 0.03703704, 0.07407407,  
0.03703704, 0.07407407, 0.03703704, 0.07407407, 0.03703704, 0.03703704, 0. ,  
0.03703704, 0. , 0.03703704, 0.03703704, 0.03703704, 0.03703704, 0.03703704,  
0.03703704, 0.03703704, 0.03703704, 0.03703704, 0. ]
```

Weight Vector:

[2. , 2. , 2. , 2. , 2. , 2. , 2. , 2. , 2. , 2. , 2. , 2. , 2. , 2. , 2. , 2. , 2. ,
2. , 2. , 2. , 2. , 2. , 2. , 2. , 2. , 2. , 2. , 2. , 1. , 1. , 0.5, 0.5,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 1. , 1. , 1. , 1. , 1. , 1. , 1. , 1. ,
1. , 1. , 1. , 1. , 1. , 1. , 1. , 1. , 1. , 1. , 1. , 1. , 1. , 1. , 1. , 1. ,
1. , 1. , 1. , 1.]

Feature Vector Document 1 (weighted):

[0. , 0.08333333, 0.08333333, 0.08333333, 0.08333333, 0. , 0.08333333,
0.08333333, 0.08333333, 0.16666667, 0. , 0.08333333, 0.08333333, 0.08333333,
0.16666667, 0.08333333, 0.08333333, 0.08333333, 0.08333333, 0.08333333,
0.08333333, 0.08333333, 0. , 0.08333333, 0. , 0. , 0.08333333, 0.08333333,
0.16666667, 0.83333333, 0. , 0.02083333, 0. , 0.02083333, 0.02083333, 0.02083333,
0. , 0. , 0.02083333, 0.02083333, 0. , 0. , 0.02083333, 0.02083333, 0. , 0. ,
0.02083333, 0. , 0.02083333, 0.02083333, 0. , 0.02083333, 0. , 0.02083333, 0. ,
, 0.02083333, 0. , 0.02083333, 0.02083333, 0.02083333, 0. , 0.02083333, 0. , 0. ,
0.02083333, 0. , 0.02083333, 0. , 0.02083333, 0. , 0.02083333, 0. , 0. , 0. ,
0.02083333, 0. , 0.02083333, 0. , 0.02083333, 0.02083333, 0.02083333, 0.02083333,
0. , 0.02083333, 0.02083333, 0.04166667, 0. , 0.02083333, 0.02083333, 0.04166667,
0.02083333, 0.02083333, 0.02083333, 0.02083333, 0.02083333, 0.02083333,
0.02083333, 0.02083333, 0.02083333, 0. , 0.02083333, 0. , 0. , 0.02083333,
0.02083333, 0. , 0.04166667, 0.04166667, 0.04166667, 0.04166667, 0. , 0.04166667,
0.04166667, 0.08333333, 0. , 0.04166667, 0.04166667, 0.08333333, 0.04166667,
0.04166667, 0.04166667, 0.04166667, 0.04166667, 0.04166667, 0.04166667,
0.04166667, 0.04166667, 0. , 0.04166667, 0. , 0. , 0.04166667, 0.04166667]

Feature Vector Document 2 (weighted):

[0.07407407, 0.07407407, 0. , 0.07407407, 0.07407407, 0.07407407, 0.07407407,
0.07407407, 0.07407407, 0.14814815, 0.07407407, 0.07407407, 0.14814815,
0.07407407, 0.14814815, 0.07407407, 0.07407407, 0. , 0.07407407, 0. , 0.07407407,
0.07407407, 0.07407407, 0.07407407, 0.07407407, 0.07407407, 0.07407407, 0. ,
0.18518519, 0.81481481, 0.01851852, 0. , 0.01851852, 0. , 0.01851852, 0. ,
0.01851852, 0.01851852, 0.01851852, 0. , 0.01851852, 0.01851852, 0. , 0. ,
0.01851852, 0.01851852, 0.01851852, 0.01851852, 0.01851852, 0. , 0.01851852,
0. , 0.01851852, 0. , 0.01851852, 0. , 0.01851852, 0. , 0. , 0.01851852, 0. ,
0. , 0.01851852, 0. , 0.01851852, 0. , 0.01851852, 0.01851852, 0.01851852, 0. ,
0.01851852, 0.01851852, 0.01851852, 0. , 0.01851852, 0. , 0.01851852, 0.01851852,
0. , 0.01851852, 0.01851852, 0.01851852, 0.01851852, 0.01851852, 0.03703704,
0.01851852, 0.03703704, 0.01851852, 0.03703704, 0.01851852, 0.01851852, 0. ,
0.01851852, 0. , 0.01851852, 0.01851852, 0.01851852, 0.01851852, 0.01851852,
0.01851852, 0.01851852, 0.01851852, 0.01851852, 0.01851852, 0. , 0.03703704, 0.03703704,
0. , 0.03703704, 0.03703704, 0.03703704, 0.03703704, 0.03703704, 0.07407407,
0.03703704, 0.07407407, 0.03703704, 0.07407407, 0.03703704, 0.03703704, 0. ,
0.03703704, 0. , 0.03703704, 0.03703704, 0.03703704, 0.03703704, 0.03703704,
0.03703704, 0.03703704, 0.03703704, 0.03703704, 0.]

Weighted distance 0.946286426282
Cosine Similarity: 0.936140972642

List of Figures

2.1	Natural Language Processing tasks	3
2.2	Depth-first tree-search algorithm, taken from Holden [16]	4
2.3	Synonyms and word sense disambiguation in similarity detection, taken from Abdalgader et al. [1]	5
2.4	Example of a subject graph, as presented by Wang et al. [37]	6
2.5	Similarity detection by means of a hash function, as presented by Gipp [11]	6
3.1	Plagiarism Detection Approaches	8
3.2	Local and global similarity detection, as presented by Stein [34]. The left side illustrates all identical n-grams with a length ≥ 5 between two documents A and B, while the right side shows global similarity, where common word stems without stop words of A and B are highlighted.	9
3.3	Glyphs in an unmodified (left) and modified (right) font [14]	19
4.1	Illustration of the methodology	21
4.2	Cosine similarity between three documents vectors d_1 to d_3 and a query vector q as presented by Manning et al. [18], where q comprises the words <i>jealous</i> and <i>gossip</i>	25
4.3	Applying the weight vector to the feature vector	26
4.4	Sample tree of the TreeTagger, as illustrated by Schmid [30]	28
4.5	Storing synonyms in separate list entries	29
4.6	Time complexity of $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$	30
4.7	Incorporating words in a hash structure, where synonyms point to the same entry.	31

List of Tables

3.1	Comparison of existing plagiarism detection tools.	12
3.2	Fourfold table reflecting how values are determined	14
3.3	Evaluation of existing plagiarism detection tools.	15
3.4	Example of the paraphrased text	15
3.5	Example of the translated text	16
3.6	Example of the Small SEO rewritten text	16
3.7	Results for different degrees of disguise	17
3.8	Wittgenstein's TS 213 compared to MS 114 [26]	17
3.9	Results for the comparison of TS 213 and MS 114	18
4.1	Extract of English and German stop words	23
4.2	Sample TreeTagger output, taken from the TreeTagger website	28
4.3	Comparison of the new approach with the best performing tools	32
4.4	Vector comparison for two identical documents	33
4.5	Vector comparison with the paraphrased document	33
4.6	Comparison with tools that compare two documents	34
4.7	Similarity between Wittgenstein's TS 213 and MS 114 [26]	34
A.1	Wittgenstein's TS 207 - original version	41
A.2	Wittgenstein's TS 207 - paraphrased version	42
A.3	Wittgenstein's TS 207 - translated version	43
A.4	Wittgenstein's TS 207 - Small SEO rewritten version	44
B.1	Exact copy results	45
B.2	Paraphrased results	45
B.3	Translated results	45
B.4	Small SEO rewritten results	45
B.5	Overall results	46
C.1	List of all English stop words	47
C.2	List of all German stop words	47

CD Content

Bachelor's Thesis

Contains the electronic version of the thesis

- Latex files (all chapters, including glossary, list of abbreviations, and .bib-File)
- PDF-version

Code

Contains all software and written code

- ext: contains external software (TreeTagger)
- lib: contains two modules (calculate.py and synonyms.py)
- make: contains Makefiles
- res: contains resources (.txt files)

References

Contains all references cited in the thesis

Bibliography

- [1] K. Abdalgader and A. Skabar. Short-text similarity measurement using word sense disambiguation and synonym expansion. In *Australasian Joint Conference on Artificial Intelligence*, pages 435–444. Springer, 2010.
- [2] T. Berners-Lee. Twitter post. https://twitter.com/timberners_lee/status/511988109211627520, 2014. [Accessed 10-April-2017].
- [3] S. Boon. 21st century science overload. <http://www.cdnsiencepub.com/blog/21st-century-science-overload.aspx>, 2016. [Accessed 10-April-2017].
- [4] S.-H. Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.
- [5] Cheat for Turnitin (Blogger). How to cheat turnitin? <http://cheatturnitin.blogspot.de/2011/04/tips-for-how-to-cheat-turnitin.html>, 2011. [Accessed 14-April-2017].
- [6] M. Y. M. Chong. A study on plagiarism detection and plagiarism direction identification using natural language processing techniques. 2013.
- [7] T. A. E. Eisa, N. Salim, and S. Alzahrani. Existing plagiarism detection techniques: A systematic mapping of the scholarly literature. *Online Information Review*, 39(3):383–400, 2015.
- [8] A. Ekbal, S. Saha, and G. Choudhary. Plagiarism detection in text using vector space model. In *Hybrid Intelligent Systems (HIS), 2012 12th International Conference on*, pages 366–371. IEEE, 2012.
- [9] C. Fellbaum. Wordnet(s). In K. Brown, editor, *Encyclopedia of Language and Linguistics*, volume 13, pages 665–670. Oxford: Elsevier, 2 edition, 2006.
- [10] R. A. Finkel, A. Zaslavsky, K. Monostori, and H. Schmidt. Signature extraction for overlap detection in documents. In *Australian Computer Science Communications*, volume 24, pages 59–64. Australian Computer Society, Inc., 2002.
- [11] B. Gipp. *Citation-based plagiarism detection*. Springer, 2014.
- [12] D. Gupta et al. Study on extrinsic text plagiarism detection techniques and tools. *Journal of Engineering Science and Technology Review*, 9(5):150–164, 2016.
- [13] B. Hamp, H. Feldweg, et al. Germanet-a lexical-semantic net for german. In *Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 9–15, 1997.
- [14] J. Heather. Turnitoff: Identifying and fixing a hole in current plagiarism detection software. *Assessment & Evaluation in Higher Education*, 35(6):647–660, 2010.
- [15] V. Henrich and E. W. Hinrichs. Gernedit-the germanet editing tool. In *ACL (System Demonstrations)*, pages 19–24. Citeseer, 2010.

- [16] S. B. Holden. Artificial intelligence I, University of Cambridge. <http://www.cl.cam.ac.uk/teaching/1314/ArtIntI/ai1-colour-2014.pdf>, 2014. [Accessed 10-June-2017].
- [17] C. K. Kent and N. Salim. Features based text similarity detection. *Journal of Computing*, 2(1), 2010.
- [18] C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.
- [19] B. Martin. Plagiarism: a misplaced emphasis. *Journal of Information Ethics*, 3(2):36–47, 1994.
- [20] N. Mazov, V. Gureev, and D. Kosyakov. On the development of a plagiarism detection model based on citation analysis using a bibliographic database. *Scientific and Technical Information Processing*, 43(4):236–240, 2016.
- [21] M. Mozgovoy. *Enhancing computer-aided plagiarism detection*. University Of Joensuu, 2007.
- [22] K. Mustofa and Y. A. Sir. Early-detection system for cross-language (translated) plagiarism. In *Information and Communication Technology-EurAsia Conference*, pages 21–30. Springer, 2013.
- [23] D. Noyes. First url active once more. <http://first-website.web.cern.ch/blog/first-url-active-once-more>, 2013. [Accessed 10-April-2017].
- [24] A. H. Osman, N. Salim, and A. Abuobieda. Survey of text plagiarism detection. *Computer Engineering and Applications Journal (ComEngApp)*, 1(1):37–45, 2012.
- [25] S. d. L. Pertile, V. P. Moreira, and P. Rosso. Comparing and combining content-and citation-based approaches for plagiarism detection. *Journal of the Association for Information Science and Technology*, 2015.
- [26] A. Pichler, H. Krüger, D. Smith, T. Bruvik, A. Lindebjerg, and V. Olstad, editors. *Wittgenstein Source Bergen Facsimile (BTE)*. Wittgenstein Source Bergen, 2009.
- [27] P. Rosso and F. M. R. Pardo. Uncovering plagiarism-author profiling at pan. In *Ercim News*, number 96, pages 49–49. ERCIM EEIG, 2014.
- [28] C. Sadowski and G. Levin. Simhash: Hash-based similarity detection, 2007.
- [29] H. Schmid. Improvements in part-of-speech tagging with an application to german. In *In proceedings of the acl sigdat-workshop*. Citeseer, 1995.
- [30] H. Schmid. Probabilistic part-of-speech tagging using decision trees. *Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart*, 43:28–37, 1995.
- [31] H. Schütze, H. Schmid, A. Fraser, P. Koehn, and N. Durrani. The operation sequence model—combining n-gram-based and phrase-based statistical machine translation. In *Computational Linguistics*. MIT Press, 2015.
- [32] H. Schütze, N. T. Vu, H. Adel, and P. Gupta. Combining recurrent and convolutional neural networks for relation classification. *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2016.

- [33] B. Stein, N. Lipka, and P. Prettenhofer. Intrinsic plagiarism analysis. *Language Resources and Evaluation*, 45(1):63–82, 2011.
- [34] B. Stein and S. M. Zu Eissen. Near similarity search and plagiarism analysis. In *From data and information analysis to knowledge engineering*, pages 430–437. Springer, 2006.
- [35] The Netherlands EU Presidency. All european scientific articles to be freely accessible by 2020. <https://english.eu2016.nl/latest/news/2016/05/27/all-european-scientific-articles-to-be-freely-accessible-by-2020>, 2016. The Netherlands EU Presidency.
- [36] J. Tomita, H. Nakawatase, and M. Ishii. Calculating similarity between texts using graph-based text representation model. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 248–249. ACM, 2004.
- [37] J. Wang, C. Vachet, A. Rumpel, S. Gouttard, C. Ouziel, E. Perrot, et al. Multi-atlas segmentation of subcortical brain structures via the autoseg software pipeline. *Front. Neuroinf.*, 8:1–11, 2014.