Plagiarism Detection: A combined Approach

Erisa Perleka

Master of Science in Computer Science
Submission date: July 2013
Supervisor: Bjørn Gamback, IDI
Co-supervisor: Erwin Marsi, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

Write your summary here...

Abstract

The task of automated plagiarism detection is about finding out pieces of a text which have been derived from another text. This task could require to scan a large number of documents, until the one from which the text it stolen is found. However, this project is confined only to the problem of finding stolen text in a given document when the original document from which it is taken is already given to the detector which should thus only consider a comparison of these two documents. The more refined concern of this project was to enhance the performance of a given plagiarism detector by combining its approach with another complementary approach. Moreover we tried to improve this system by addressing some of its deficiencies, the most important of these being the detection of whole paragraphs

Sammendrag

Skriv sammendrag på norsk...

Preface

This report is the result of work done in the course TDT4900 - Masters degree in computer science with a specialization in intelligent systems. I would like to thank professor Björn Gambäck for all his advice and input during the last year and André Lynum for setting up the experimental environment and all his valuable advices during the last phase of the project

Table of Contents

Problem Description i

Abstract i

Sammendrag i

Preface iii

Table of Contents vi

List of Tables vii

List of Figures ix

Abbreviations x

1 Introduction 1
- 1.1 Background and Motivation . . . 1
- 1.2 Goals and Research Questions . . . 2
- 1.3 Research Method . . . 3
- 1.4 Report Structure . . . 4
- 1.5 Contributions . . . 4

2 Theory and Background 5
- 2.1 Plagiarism . . . 5
  - 2.1.1 The concept . . . 6
  - 2.1.2 Kinds of plagiarism . . . 6
  - 2.1.3 PAN and Artificial plagiarism . . . 9
- 2.2 Formal Representations of Natural Language . . . 10
  - 2.2.1 From Texts To Tokens: Basic text processing . . . 11
  - 2.2.2 Representing syntax . . . 12
  - 2.2.3 Representing meaning . . . 13
- 2.3 Plagiarism detection techniques . . . 16
  - 2.3.1 Semantic relatedness approaches . . . 17
  - 2.3.2 Calculating semantic similarity . . . 17

2.3.3 The Longest-shared-passage problem 20
2.4 Performance evaluation 21

3 Related Work 25
3.1 Detecting Similarity Between Sentences 25
3.1.1 SimPaD 25
3.1.2 Dao & Simpson 26
3.2 Semantic Similarity in WordNet 26
3.3 Evaluating the Detection of Sentence Semantic Similarity 26
3.4 PAN13 Participants 27
3.4.1 • 27

4 A Detector Based on Graph Edit Distance 29
4.1 The Plagiarism Detector Created by Røkenes 29
4.2 Results and Deficiencies 31

5 Improvements to the Detailed Analysis Algorithm 33
5.1 Weighting Dependency Relation Substitution 33
5.2 Excluding the Candidate Retrieval phase 34
5.3 Semantic Distance 35
5.4 Combining GED and Semantic Distance 37
5.5 Detecting whole plagiarised paragraphs 37
5.5.1 Merging Adjacent Passages 45

6 Results and Evaluation 47
6.1 How does AI research proceed 47
6.1.1 Refine a topic to a task 48
6.1.2 Designing the Method 49
6.1.3 Build a Program 49
6.1.4 Design Experiments 50
6.1.5 Analyze the Experiment's Results 50
6.2 Experiments: Tuning parameters 51
6.3 Results 51

7 Conclusions 55
7.1 Further Work 56

Appendix 59

List of Tables

1.1 Sources of documents with background knowledge. 4
1.2 Keywords used in search. 4

List of Figures

2.1 An example of dependencies within a sentence.(Figure taken from nlkt.googlecode.com https://nltk.googlecode.com/svn/trunk/doc/images/depgraph0.png) 13
15figure.2.2
2.3 The Longest-shared-passage Problem 21
2.4 Generic retrieval process to detect plagiarism (Figure taken from (20)) 22

4.1 Overview of the detailed analysis phase. (Taken from (28)) 30
4.2 The overall architecture of the system. (Taken from (28)) 31

5.1 Candidate retrieval phase is removed 35
5.2 The passage detection problem 38
5.3 Document representation of 00027-02493 XML result 39
5.4 A case of random obfuscation in matrix representation 40
5.5 A case of shrinking paragraphs 41
5.6 A case of expanding paragraphs 42
5.7 Matrix of the strictly diagonal case 43
5.8 44
5.9 The resulting Detailed Analysis phase 45

Abbreviations

NLP	=	Natural Language Processing
IR	=	Information Retrieval
GED	=	Graph Edit Distance
SD	=	Semantic Distance
POS	=	Part of Speech
LSA	=	Latent Semantic Analysis
WSD	=	Word Sense Disambiguation

Chapter 1

Introduction

1.1 Background and Motivation

Plagiarism is an old phenomenon. It has been present since humans started to tell stories and write them down in words. It can manifest itself in different disguises, making it sometimes difficult to infer a clear judgement about the originality of some work. The question of under which systematic examination a work can be precisely doomed as plagiarised or not, doesn't have a definite answer. It is of course easy for a human reader to identify cases when the passage is a verbatim copy of another. This passage can be a whole document or even a sentence, a phrase,or a mere metaphor or other figures of speech. The act of detecting the misdeed boils down to identifying the lexical similarity between the texts. Detecting paraphrases of original works also falls under the category of simple tasks; here the problem is slightly more advanced as the content words may be exchanged with words of different lexical form but same meaning, broadly referred to as synonyms. Here the reader has to derive the judgement by calculating with the meaning of words. Or we could face the case when the plagiariser has stolen the structure of a passage or sentence, something which is up to some point or completely covered by the syntax of the sentence, so here a syntactic analysis could reveal the similarity. The more complicated case of whether a text shows clear features of influence by another, falls outside the topic of this thesis.

The development of text processing and distributing technologies has given new advantages to both the plagiarisers and the techniques for detecting plagiarism (16). The web provides a vast amount of information, offering to a possible plagiarist what he now may need about a specific topic. The increasingly improved search engines do provide this information in critical time and the digitalized text is easily copied over in a matter of seconds. But on the other hand, these same strengths of the technology can assist a human detector in a plagiarism-hunt. It is impossible for a human to skip through eventually millions of source documents in reasonable, practical time, while information retrieval systems can complete the task in short time. It is also difficult for a human to remember details of texts he has already read, and find these same details reproduced in a plagiarised text, while a plagiarism detection system is better suited to find the needle in the haystack. The qualities still lacking in plagiarism systems are those of identifying similar meanings in acclaimed different texts, or detecting an overall similar structure among large texts.

This thesis focuses on that part of plagiarism detection that has to do with finding similar passages between two given texts, thus skipping the step of retrieving candidate source texts on the basis of a given suspicious text. So we are only concentrated on the problem of text alignment between two documents where one of them has, or has not, text snippets that are derived from the other.

1.2 Goals and Research Questions

Goal 1 Find existing solutions and methods for the plagiarism detection systems in the NLP literature.

This goal is reached by researching the literature available, about NLP and more specifically, automated plagiarism detection systems.

Goal 2 Enhance the performance of the system described by (28) by:

- Improving the weights used in the Graph-Edit-Distance algorithm.
- Improve the precision of paragraph detection by accounting for sentences that are part of plagiarised paragraphs but give low score of similarity.
- The system shall still scale well enough to run and be tested on the PAN data. PAN is going to be introduced later in this paper.
- Combine the analysis of the above mentioned system (28) with the features obtained by another algorithm that accounts for semantic similarity between text snippets.

In order to achieve this goal we have to acquire an insight into the workings of the system, by understanding the overall architecture and the low level details of the implementation in order to interfere in it. Moreover, a survey of the theoretical background upon which the system is built has to be surveyed.

The fulfilment of this goal requires also an investigation in the workings of this other algorithm, and the implementation that allows for an integration of these two similarity/distance measures.

Research question 1 How to improve the detection of whole paragraphs, when sentences adjacent to an already detected sentence, get a lower similarity score, even though part of the plagiarised section?

Research question 2 Can graph edit distance methods, in particular the approach taken by Røkenes (28), be combined with methods that measure semantic relatedness/similarity of texts?

Research question 3 Given that a combination of the two methods is possible, is this done best by adding a semantic similarity measure in the word level, or by combining the scores derived by the two algorithms when these are given the same text passages as input? The two approaches should be tried.

Research question 4 What are the respective advantages and disadvantages of measuring syntactic similarity versus measuring semantic similarity?

1.3 Research Method

In order to attempt an experimentation in the field, firstly the state of art of automatic plagiarism detection systems had to be reviewed. This did also ask for a further review, that of background material, most of which was unknown before. Most of the basic knowledge about NLP was derived from the Jurafsky and Martin's book on the topic (2), which was curriculum for the "Natural Language Interfaces" course, TDT4275 [1]. The slides of the "Computational Semantics" course TDT13[2] were also reviewed, when knowledge about Computational Semantics was needed. Since this work relies primarily on a previous master thesis[3] and the prototype system that was developed in its context, much of the initial research was constrained by the themes covered in it. These themes included: *graph-based representation of natural language sentences*, *graph-edit distance algorithms* and *the assignment problem*. Since this system is built on the assumption that it will be trained and tested by the datasets offered by plagiarism detection competition held in the PAN workshop at CLEF[4], we also had to review the techniques and concepts relevant to this workshop.

The papers produced by the previous PAN participants, have also been an important part of the curriculum, as PAN claims to promote a state of the art situation for the plagiarism detection techniques.

When using the web, the main keywords that were adopted in searching were :*plagiarism*, *plagiarism detection* and *text alignment*. The subcategories that fall under these fields are listed in the Table 1.2.

Of course, when doing literature research one has to pick only a portion of the articles retrieved on a given topic. When deciding which articles would be considered relevant, the criteria that we used for the selection were:

1. Does the article describe a solution to the problem topic we have? A time-constraint of choosing not-too-old articles, was also applied.

2. Does the article provide commanding background material that underlies the understanding of the field? In other words, is the material relevant enough to be used in the theoretical part of the thesis?

Some of the articles that were cited in a given relevant article were also exploited for gaining further insight. Sometimes this was necessary, as the understanding of a work that relies on another craves knowledge of its precursor. The relevance of an article was mostly judged by reading the abstract or introduction and/or the conclusion. When this was not enough, we had to flip through the whole thing. Most of the articles retrieved were provided from the sources given in Table 1.1.

[1]http://www.idi.ntnu.no/emner/tdt4275/
[2]http://www.idi.ntnu.no/emner/tdt13/
[3]Røkenes,Graph-based Natural Language Processing,2012
[4]http://www.clef2013.org/

Source	Url
IEEE Xplore	http://ieeexplore.ieee.org/Xplore/guesthome.jsp
Springer Link	http://springerlink.com
CiteSeerX	http://citeseerx.ist.psu.edu

Table 1.1: Sources of documents with background knowledge.

Plagiarism	**Natural Language Processing**	**Plagiarism Checker**
Plagiarism	Dependency Grammar	CaPD
Automated Plagiarism Detection	Parsing	
	Dependency Parsing	
	Semantic Similarity	
	Language Models	

Table 1.2: Keywords used in search.

1.4 Report Structure

In this section we presented the method used to explore the field. The following chapter shall go through the concepts and definitions that are considered necessary to comprehend the rest of the thesis. After we have showed an understanding of the basic concepts and principles that are important to automatic plagiarism detection systems and natural language processing, some related work of the area is introduced in the next chapter. After that we present the prototype that is implemented to test the ideas cited in the background material. In the end an evaluation of the system is launched. The thesis is closed with conclusions derived from the experiments and proposals for further work.

1.5 Contributions

The work that we contributed with during this project, was that of defining the weights of dependency relation substitutions, integrating the graphd-edit distance algorithm with an algorithm that accounted for semantic distance between pairs of sentences and finally, we implemented a strategy that merges sparse detected passages into whole paragraphs.

Chapter 2

Theory and Background

Plagiarism detection by an application is a topic which subsumes the even more fundamental matter of how to make computers cope with natural language in general, may the task include language understanding, language generation or both (9). The issue of natural language processing has been addressed as an important aspect of artificial intelligence, since the early years of computer science, the most important milestone being Turing's 1950 paper "Computing Machinery and Intelligence" (6), where the Turing-test[1] is sketched and proposed. Using natural language (besides reasoning, having knowledge and being able to learn) is one of the criteria to be fulfilled in the test. This aim is hitherto addressed by trying to formalize natural language down to the point of making it processable by the application (14). In this chapter we show, among other things, some of these formalization techniques.
Our plagiarism detection goal can thus be transfigured into the goal of finding similarities between these formalized structures of language. How we define and measure the similarity is strictly dependent on the specific representation chosen. Various plagiarism techniques ask for different detection strategies. Plagiarism is a product of the human mind; it is though a highly non-deterministic event. This thesis is confined to the goal of detecting the plagiarism on document level, which means, finding which passages of a suspicious document correspond to and can be claimed equivalent to passages that are to be found in some source (original) documents. This task entails two subtasks: the first is finding the source document from which the suspicious document has stolen the passage. The second is pointing exactly to the passage plagiarised. Again, these two tasks can be broken down into other tasks, but in the bottom of both of them, lie three problems: the first, also mentioned above, that of representing natural language in computational terms.The second is that of determining when two passages are similar enough for the case to be an instance of plagiarism. One challenge here is that plagiarism assessment is discussable even among humans. The third, to develop techniques that measure these similarities.

2.1 Plagiarism

This section presents what is understood by plagiarism in general and discusses some instances of this phenomenon that are of interest to this thesis.

[1]http://plato.stanford.edu/entries/turing-test/

2.1.1 The concept

The term *plagiarism* appears for the first time as early as in the 1st century, when it was coined by the latin poet Martial, in the 1.52 passage of his *"Epistemmes"* (30). It has since then been used to define the theft of not only language artifacts, but of any work that beares authorship and originality. We shall, however, confine our study to the detection of plagiarism in texts.
Plagiarism is the act of using someone else's work without giving a proper recognition of its original author. Although the concept and its connotation are familiar, it has given rise to a variety of definitions, making thus vague the parameters that would decide if a work can be considered as plagiarised or not. This lack of clarity in the definition also contributes to the difficulty of automating plagiarism detection. Sometimes it can be unambiguously clear that a certain piece of text is a mere copy of another. But plagiarists often use techniques that alienate the new, plagiarised text from the source. Here the problem becomes more intriguing, but it still remains decidable to a human reader who is faced with both the source and the suspicious text. Here is a definition that lies most nearly to our understanding of the concept:

> "The wrong in plagiarism lies in misrepresenting that a text originated from the person claiming to be its author when that person knows very well that it was derived from another source, knows that the reader is unlikely to know this, and hopes to benefit from the reader's ignorance." (Samuelson, 1994)

The core phrase that serves our purposes is stating that a plagiarised text is "derived from another source". The task of an automatic plagiarism detector, is primarily to find out from which source the text is taken, but also argue about the nature of the specific derivation used. Showing what kind of plagiarism the text has been a victim of, is more persuasive than giving a plagiarised/not-plagiarised binary classification. Upon which rules a plagiarism checker should make its examination, is one of the oldest and still relevant research problems. So one of the goals for the automatic detection is to find suitable, quantifiable features that can be used as parameters in the detection (10). In the following, we show some instances of plagiarism, to make concrete the tasks a detector has to accomplish.

2.1.2 Kinds of plagiarism

To illustrate the concept and the kind of problems we are faced with, we present here some examples of plagiarising from a text, taken from the website of Georgetown University. [2] The recycled text is typed in red, the reformulated phrases or sentences are in blue:

THE ORIGINAL PASSAGE

> This book has been written against a background of both reckless optimism and reckless despair. It holds that Progress and Doom are two sides of the same medal; that both are articles of superstition, not of faith. It was written out of the conviction that it should be possible to discover the hidden mechanics by which all traditional elements of our political and spiritual world were dissolved into a conglomeration where everything seems to have lost specific value, and has become unrecognizable

[2] http://gervaseprograms.georgetown.edu/honor/system/53501.html

for human comprehension, unusable for human purpose. Hannah Arendt, **The Origins of Totalitarianism** (New York: Harcourt Brace Jovanovich, Inc., 1973 ed.), p.vii, Preface to the First Edition.

EXAMPLE I: Word-for-word(verbatim) plagiarism

This book has been written against a background of both reckless optimism and reckless despair.It holds that Progress and Doom are two sides of the same medal; that both are articles of superstition,not of faith.Interestingly enough, Arendt avoids much of the debates found in some of the less philosophical literature about totalitarianism.

EXAMPLE II: The Paraphrase

Hannah Arendt's book, The Origins of Totalitarianism, was written in the light of both excessive hope and excessive pessimism. Her thesis is that both Advancement and Ruin are merely different sides of the same coin. Her book was produced out of a belief that one can understand the method in which the more conventional aspects of politics and philosophy were mixed together so that they lose their distinctiveness and become worthless for human uses

EXAMPLE III: The Mosaic

The first edition of The Origins of Totalitarianism was written in 1950. Soon after the Second World War,this was a time of both reckless optimism and reckless despair. During this time, Dr. Arendt argues, the traditional elements of the political and spiritual world were dissolved into a conglomeration where everything seems to have lost specific value. In particular, the separation between the State and Society seems to have been destroyed. In this book, she seeks to disclose the hidden mechanics by which this transformation occurred.

EXAMPLE IV: The "Apt Phrase"

Following the Second World War, scholars from a variety of disciplines began to explore the nature of "totalitarianism." One of the most pressing issues for these writers was understanding the "essence" of totalitarianism. How, for example, is a totalitarian regime different from an authoritarian regime? Although authors disagree on the precise answer to this question, a common thread running throughout most of the classic works on totalitarianism deals with the relationship between State and Society. In a totalitarian state, the traditional boundaries between State and society are dissolved into a conglomeration so that the two become indistinguishable.

To define the plagiarism methods, we follow (10) who distinguishes five ways of doing plagiarism. The list below is taken from (8):

1. **Verbatim (word-for-word) plagiarism**: direct copying of phrases or passages from a published text without quotation or acknowledgement. This is what we see in Example I and is also the easiest to detect. The algorithm could simply identify the perfect one-to-one match between the lexemes of the text.

2. **Paraphrasing plagiarism**: is the case when words or syntax are rewritten, but the source text can still be recognised. This is illustrated in Example II. The task is more complicated for an automated detector as the phrases are lexically different. Here is the list of transformations:

- ”*reckless despair and reckless optimism*” $\longrightarrow$ ”*excessive hope and excessive pessimism*”. Here we have both a change of lexicon and a switch of the word-order inside the phrase. In this case, the algorithm would have to know how to find out that ”despair” $\rightarrow$ ”pessimism” , ”optimism” $\rightarrow$ hope and ”reckless” $\rightarrow$ ”excessive” are pairs of synonyms. In order to give a high score of phrase similarity the algorithm must also ignore the order of content words.

- ”*progress and doom are two sides of the same medal* ” $\longrightarrow$ ”*both advancement and ruin are the merely different sides of the same coin*”
  In addition to reformulation of the phrases, here we have the insertion of ”merely” and ”both”. Besides equating between ”progress” and ”advancement”, ”doom” and ”ruin” and, ”medal” and ”coin”, the algorithm should assign very little importance to the insertion of ”both” and ”merely”. One could for example argue that ”both” is in the second sentence a semantic equivalent of ”two” in the first sentence, which is deleted. While ”merely”, as an adverb, has little to add to the meaning of the sentence. So the algorithm should be able to judge ”merely” as redundant.
  While the last sentence of the Example II is completely reformulated, asking for the detector to disclose the similarities by detecting the similarities between the two sentence meanings. Here the problem of defining the meaning of a sentence computationally is encountered. This shall be further discussed later on.

3. **Plagiarism of secondary sources**: When original sources are referenced or quoted, but obtained from a secondary source text without looking up the original. We will not pay much attention to this case, as it involves a chain comparison between texts. E.g, comparing text1 with text2 which is in turn compared to text3.

4. **Plagiarism of the form of a source**: Appears when the structure of an argument in a source is copied. This can be the same case as when paraphrasing by keeping the sentence structure and exchanging content words with synonyms as in Example II. Or merely copying the syntactical content of a sentence, without regard to word meaning.

5. **Plagiarism of ideas**: Is the reuse of an original thought in a source text without dependence on the words or form of the source. Assuming that ideas have a meaning, this case could count as paraphrasing with different words. This can be illustrated by the last sentence of Example II.

Another method, that is not listed by (10) or (8), is that of **Back Translation**. This constitutes in translating a text, verbatim, from the original to another language and then retranslating it back to the original with the structure, words and meaning somehow changed. In this case we mean machine translation, so the semantic and the syntax of the translated/retranslated text could miss some of its accuracy after this process. (16) argues that among other factors, this form of plagiarism is encouraged by the easiness of pursuing it, given the continuously ameliorating state of the online automatic translation applications. For example, by using Google Translate[3]. In the case below, the application obfuscates an English sentence by translating it to Italian and then back to English:

”*They have lived here all their lives, they know the land*” $\longrightarrow$ ”Hanno vissuto qui tutta la vita, conoscono il territorio ”$\longrightarrow$ ”*They have lived here all my life, know the territory*” . So Google suggests substituting ”land” with ”territory”.

In this thesis, we try to address the *Verbatim*, *Paraphrasing* and the *Mosaic* cases of plagiarism since, as we shall see below, they are considered by the PAN challenge (see next section) which is used as an evaluation reference.

2.1.3 PAN and Artificial plagiarism

In order to develop an intelligent plagiarism checker, in most cases one has to train it on a large amount of data. Test data is also needed when one has to measure the performance of the system. It is, however, quite difficult to get hold of big sets of real plagiarism cases, as these cases are often held secret and publishing them would ask for a permission from both the plagiariser and the author of the original (25).

PAN (Plagiarism analysis, Author identification and Near-duplicate detection) evaluation lab is a workshop held in the context of the annual CLEF (Conference and Labs of the Evaluation Forum) [4] event.

The PAN evaluation lab proposes, following (25), a corpus of suspicious and source documents, where the suspicious documents are created utilising the source documents by using three obfuscation strategies. Here s_{plg} denotes the plagiarised passage and s_{src} denotes the source passage. The below definitions are taken directly from (25):

- Random Text Operations : s_{plg} is created from s_{src} by shuffling, removing, inserting, or replacing words or short phrases at random. Insertions and replacements are taken from the document d_{plg} where s_{plg} is to be inserted.
- Semantic Word Variation: s_{plg} is created from s_{src} by replacing words with their synonyms, antonyms, hyponyms, or hypernyms, chosen at random. A word is kept if none are available.
- POS-preserving Word Shuffling: The sequence of parts of speech in s_{src} is determined and s_{plg} is created by shuffling words at random while retaining the original POS sequence.

The drawback of the artificial plagiarism, is that is does not allow us to talk about the meaning of the sentence as a whole. It is also not reasonable to try capturing the meaning of a word by analysing its neighbours, as there is no intentional connection between them. We show here a fragment taken from a plagiarised document of the PAN13 data set, which has been prone of random obfuscation. The text below is reproduced keeping the original indentation:

[3]http://translate.google.com
[4]clef2014.clef-initiative.eu

”table alcohol” in the sake existence, and make up about 80 of all welfare that is make. Sake like this
is added with distilled amounts of copious concentrated intoxicant make to addition give.

2.2 Formal Representations of Natural Language

Machines do not speak. In order for a computer application to use the knowledge that relies in a natural language utterance, it must have it represented in a way that is suitable for processing. This section discusses some of the techniques used to formalize natural language. Representing linguistic knowledge is a subtopic of the the problem of knowledge representation in computer science. As (11) show, one of the properties of knowledge representation is that of being a surrogate, which means, being an incomplete representation of a real world concept. This quality, or lack of quality, is also valid for the problem of representing knowledge about natural language. What makes this problem more complicated, is the fact that natural language is prone to containing ambiguities that occur at all levels of representation, as well as across representation levels. These ambiguities are a source of confusion not only for a computer application, but even for humans. The list below, taken from (5), states some instances of these ambiguities.

- **Lexical ambiguity**: Appears when a given morpheme can take on different part-of-speech roles, depending on the context. For example, ”light” can be a noun, as in ”the light is on” or an adjective, as in ”the package is light”.
- **Structural or syntactic ambiguity**: Has to do with finding out in which phrase group a word belongs to. For example, ” I saw a man with a telescope”, here the telescope could ”be held” by the subject or by the object, and it is be debatable what the sentence actually denotes.
- **Semantic ambiguity**: Many words have more than one meaning, as for example the verb ”go”,for which we could list ten different meanings.
- **Pragmatic ambiguity**: Sometimes it is confusing what a phrase or sentence is hinting to. For example, ”Can you lift that rock?” could be a yes/no question or a request to lift the rock.

Humans resolve the lexical ambiguity by using the context defined by the sentence in which a phrase is used. The syntactic ambiguity craves a larger context, like in the above example it can be solved in the context of a paragraph. The semantic ambiguity above could be solved within the sentence, while the pragmatic ambiguity asks again for a larger context. It is obvious that the larger the context the more difficult it is to take a decision by formal means.
Another feature of language is that of being redundant, which means that phrases or whole passages have elements that they can exist without, both semantically and syntactically. As in the Example II of section 2.1.2 above, where deleting ”merely” from the sentence would not harm it in any way.
These characteristics of natural language are kept in mind when we try to build formal representations of it that are computationally consistent.

2.2.1 From Texts To Tokens: Basic text processing

Expressing linguistic knowledge in a way that it can be processable by formal means is a continuously developing trend, as (14) argue. The majority of NLP systems do employ a

Below are listed the most widespread techniques of expressing text in computational terms that are shared by the majority of systems that do natural language processing.

- Sentence partitioning: In some cases it is desirable to work within or with relations among sentences, so whole texts are partitioned into sentences.
- POS-tagging : Part-of-speech tagging is also known as word-category disambiguation, as it assigns a corresponding part-of- speech tag to every word in a sentence. This is done based on the words' definition and on the context where it is used. In computational linguistics, POS-tag algorithms are divided into rule-based and stochastic. The first perform the tagging based on a set or rules, similar to the rule-based expert systems, while the second ones rely on probabilistic models.There are also two types of taggers: the one the assigns syntactic roles (like subject, object, etc) and the ones that attach functional roles (like noun, verb, etc).
- Lemmatisation: Is the process of determining the lemma of a word. Lemma denotes the cannonical or dictionary form of a word. For example the past tense ”fed” has as lemma ”feed”.
- Stopword removal: This step removes function words like articles, pronouns, prepositions and determiners.
- Punctuation removal: This technique removes punctuation symbols.
- Parsing: This is a more complicated task as it requires both the POS-tag preprocessing step and the lemmatisation of the text as a prerequisite for its own processing of the sentence. Parsing means to do a grammatical analysis of the sentence and this requires a definition of the language grammar. The three main categories of approaches to parsing are those of :formal grammars , head-driven phrase structure grammar and dependency parsing. A detailed representation of dependency parsing is going to be carried later on.

A central question posed by the need for formal representation is how we model natural language in computational components. Or, in the case of a plagiarism detector: How do we represent documents and sentences in computational terms? These concerns are addressed by several techniques. In the following sections, we list some of these:

N-grams

N-grams are sequences of units that follow after each other consecutively. These units can be single characters, words or even phrases.The *n* in n-grams denotes how many elements we have in this structure. When talking about the similarity between two texts, n-grams can be used to denote the number of phrasal overlaps between these texts. A question that arises in this context is that of the size of *n*. A large *n* would help to detect larger pieces of similar text. The problem that a large *n* is faced with is that of sparsity. For example in the face of paraphrasing or that of syntactical obfuscation, the plagiarised text does not show the same consequence of words as the original one.

Bag of Words

When the *n* in n-grams is equal to 1, then we get the bag-of-words model. This model is widely used in information retrieval when a simple lexical representation of a document is needed (**?**). Here the ordering of the words is completely lost, but we do retain the number or times a words appears in a document. This representation is useful when we do not want to consider word-reordering during plagiarism detection.

Vector space model

As the name suggests, the Vector Space Model (VSM), represents texts as vectors of terms. The elements of these vectors denote the occurrence of a term in a document. These occurrence quantifiers are defined by several methods. The simplest one is to set a 0 when the given term does not show up in the document, and a 1, if it is present in the document. Another quantifying scheme is the one known as *tf*idf* weighting, where *tf* denotes the term frequency in the document, which is, how many times the given term appears in this document, while *idf* stands for inverse document frequency, which denotes how many documents in the whole given corpus do contain this term.

In the following sections we discuss techniques used to represent two central features of language: its syntax and its semantics.

2.2.2 Representing syntax

In linguistics, language is analysed in terms of its syntax and its semantics. The syntax sets the rules by which sentences must be constructed,i.e its grammar, while semantic is concerned with meaning. We can use syntax to analyse the structure of a sentence and find out properties about the syntactical relations between words in a sentence. While semantics can be used in both a word level and text level. We can argue about the semantic relation between two words that do or do not belong to the same sentence, but we can also talk about the meaning of a whole sentence.

When looking for plagiarism cases, we could use syntax to find out syntactically similar sentences. If these sentences in addition share a number of common lexemes, then they can be easily detected as plagiarised. Semantics can be used to claim sentences that mean the same thing, even though words can be substituted or syntactical structure may have been changed. We shall come back to these two linguistic fields when discussing various plagiarism detection techniques.

Dependency-based representation

Dependency grammar is a class of grammar formalisms where the syntactical relations between the words in a sentence are expressed as dependencies. (23) states that the ”[...] dependency syntax postulates that syntactic structure consists of lexical items linked by binary asymmetric relations (”arrows”) called dependencies. These arrows are commonly typed with the name of grammatical relations(subject, prepositional object, apposition, etc.) The arrow connects a head (governor, superior, regent) with a dependent (modifier, inferior, subordinate).” Consider the figure below:

1. I shot an elephant in my pajamas

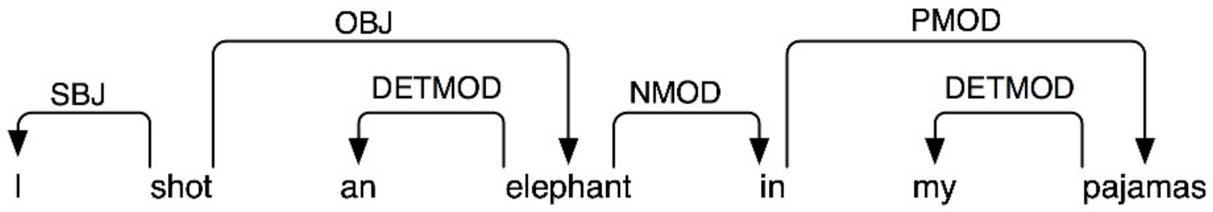

Figure 2.1: An example of dependencies within a sentence.(Figure taken from nlkt.googlecode.com https://nltk.googlecode.com/svn/trunk/doc/images/depgraph0.png)

The arrows denote the direction of the dependency from the head to the dependent. The descriptions over the arrows sign the type of the relation that exist between the two nodes. A list of these relations and their meaning can be found in (19). For example, the SBJ arrow means that *I* is the subject of the verb *shot*, the arrow NMOD means that the preposition *in* is the noun modifier of the noun *elephant* and so on. It results that dependency parsing turns the sentences into directed graphs, called dependency graphs. It is worth mentioning that this representation operates with the sentence as a unit.

Here we list some of the criteria that have been proposed for identifying a syntactic relation between a head *H* and and a dependent *D* in a construction *C* taken from (23) :

1. *H* determines the syntactic category of *C* and can often replace *C*.
2. *H* determines the semantic category of *C*; *D* gives semantic specification.
3. *H* is obligatory, *D* may be optional.
4. The form of *D* depends on *H*.

This list contains a mixture of different criteria, some syntactic and some semantic. In our work we have used the dependency relations specified in (19), which does exclusively use only the syntactic criteria.

This kind of sentence representation is favourable for the task of detecting sentences that are similar to each other.Since sentences are transformed into graphs, then the detection of similar sentences can be done by using graph edit distance algorithms. The smaller the distance between two graphs, the more similar are the two compared sentences. (28) concludes that the problem of measuring the distance between them can be transformed to that of finding the least cost for an assignment of changes, thus, to the assignment problem (17).

2.2.3 Representing meaning

In linguistics, semantics is the area coupled with the study of the meaning of texts. These texts can have the form of a word, phrase, sign or symbols [5]. So when talking about the semantics of

[5]http://en.wikipedia.org/wiki/Semantics

a word or a sentence we are looking for a definition of its meaning. The discussion of how we assign meaning to a single word both individually and by using what Eco (1) calls *Contracts* can soon become a philosophical one, so we shall not give much attention to the precise meaning of (the meaning?) of a word.
The meaning is a quality of the word and we can define a relation between two words by quantifying the relation between their meanings. Linguistics provides a list of these relations, we are mentioning some of them that are of most practical interest:

- Homonymy: The words share a form but have unrelated, distinct meanings
- Polysemy: A polysemous word has different related meanings.
- Metonymy: It's a form of polysemy that arises when we use a concept to denote some similar related concept. E.g, use the name of the author when talking about his work, or use the name of the tree when we actually mean the fruit.
- Synonymy: Different words that have the same meaning in some or all contexts.
- Antonymy: Two words that define a binary opposition, like cold/warm, long/short etc.
- Hyponymy and Hypernymy: A hyponim is a word whose meaning is included within that of another word, its hypernym. In other words, a hyponym shares a IS-A relationship whith its hypernym. E.g, pine is a hyponym of tree.

The relevant question is how to represent these relations formally. The following sections discuss this question.

Taxonomies

The most ubiquitous resource that defines the meaning of a word is the dictionary. Computational approaches to the formalization of the dictionary include both ontologies and taxonomies. Ontologies are used to represent knowledge as a "[...]hierarchy of concepts within a domain" [6]. They are hierarchically structured formalizations where two nodes are connected with each other by a link that represents the relationship between them. Taxonomies are also hierarchical structures in which concepts are organized as nodes linked by edges. The links in the taxonomies do also qualify the relationships between nodes. Ontologies do however define a larger number of relationships than taxonomies, since an ontology should ideally capture the whole knowledge that relies in a certain domain. While the type of relationship that is defined between two directly connected nodes in a taxonomy is of IS-A type, which means that a node can be either a parent or a child. E.g car IS-A vehicle entails that car is a child of vehicle. Taxonomies, thus, are branched schemes which are used to classify concepts or things. The structure of a taxonomy can be used to classify words of natural language. How words are organized and what their connecting edges denote is specific to the taxonomy in consideration. Semantic networks do also adopt the hierarchical form of a taxonomy. They consist of nodes which represent concepts and edges that represent relations between concepts [7]

[6]http://en.wikipedia.org/wiki/Ontology˙(computer˙science)

[7]http://en.wikipedia.org/wiki/Semantic˙network

WordNet

WordNet [8] is an example of a semantic network.It is a lexical database which is available online and provides a large repository of English lexical items. There is a multilingual WordNet for European languages which are structured in the same way as the English language WordNet. WordNet was designed to establish the connections between four types of POS: noun, verb, adjective and adverb. The smallest unit in WordNet is a synset, which represents a specific meaning of a word. It includes the word, its explanation and its synonyms. The specific meaning of one word under one type of POS is called a sense. Each sense of a word is in a different synset.Synsets are equivalent to senses, structures containing sets of terms with synonymous meanings. Each synset has a gloss which defines the concept it represents. For example, the words night, nighttime and dark constitute a single synset that has the following gloss: the time after sunset and before sunrise while its dark outside. Synsets are connected to one another through explicit semantic relations. Synsets are nodes in this taxonomy and they represent one sense of the words contained, so if a word has more than one sense it will be a member of different synsets in various locations in the taxonomy. WordNet defines relations between synsets and relations between word senses. A relation between synsets is a semantic relation, while a relation between word senses is a lexical relation. The difference is that lexical relations are relations between members of two different synsets, but semantic relations are relations between two whole synsets. In the example below, the relation between the synset conveyance, transport and vehicle is a semantic one, while the relation between the word automotive and car is a lexical one. The most used relation is the hyponymy, hypernymy or the IS-A relation. It connects the more general synsets to the more specific ones. The IS-A relation is transitive, so if car is a kind of automotive and automotive is a kind of wheeled vehicle than car is also a wheeled vehicle. WordNet distinguishes between common nouns and specific instances. Instances are always leaf nodes in the hierarchies they belong to.

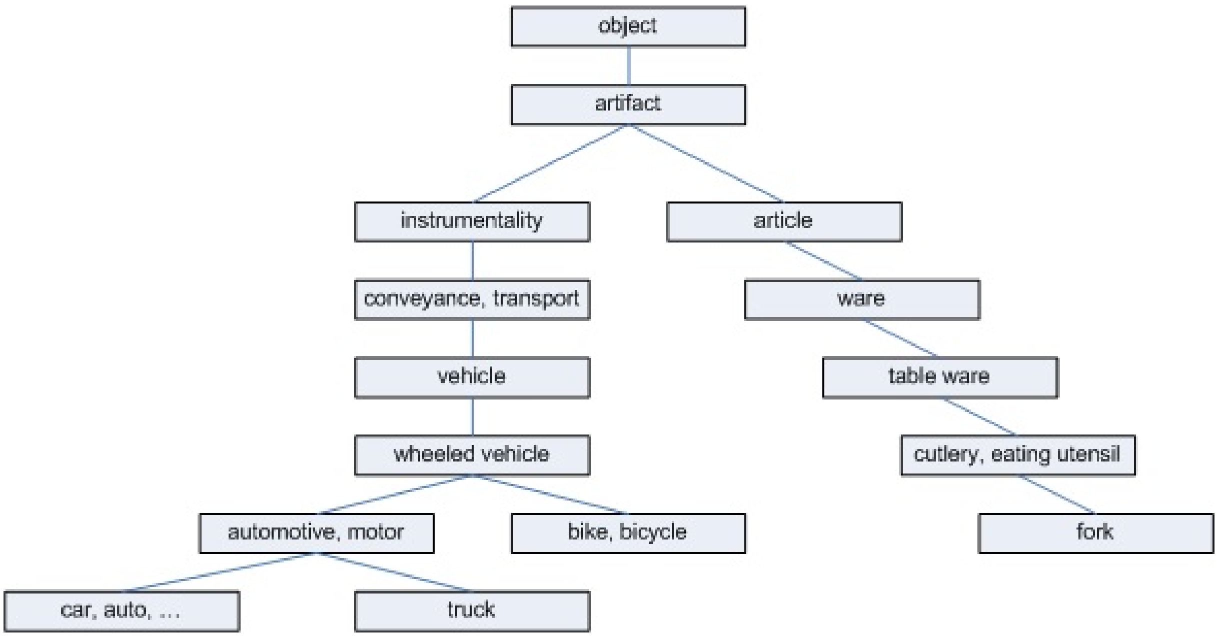

Figure 2.2: An extract from WordNet. (Taken from [9])

[8] http://wordnet.princeton.edu/

Also verbs are structured in hierarchies where the verbs that lie in the bottom express the more specific characterization of an event, like in communicate-talk-whisper. While adjectives are arranged by using the antonymy relation. Pairs of antonyms denote their strong semantic relation between their members but they are also associated with semantically similar ones. In the dry-wet dichotomy, the word dry is linked to parched , arid, dessicated etc.
As mentioned, the most common practice in WordNet is to connect words that belong to the same POS category, but a few cross-POS points do also exist. These do formalize the morphosemantic links that exist between words that share a stem with the same meaning like in: observe, observant, observatory.
Since WorNet groups the words together according to their meanings, it corresponds to a kind of thesaurus. But, unlike a thesaurus, it can be used to disambiguate the semantics of words that lie near each other.

2.3 Plagiarism detection techniques

The advantage of automatic detection is that of being able to determine the authenticity of a given text by comparing it with a large amount of other texts in a reasonable time span, something which is not feasible for a human detector. A plagiarism detector is a system that when faced with a certain text is able to determine if it contains plagiarised passages and to disclose these passages. This process is conducted differently by different detectors. A detector may choose to detect plagiarism internally, by doing an analysis of the style and vocabulary changes inside a single document, something that can be an indicator of the fact that not all parts of the document come from a single, proclaimed author. But it can also choose to disclose plagiarism by using external sources. In this case the detector compares the document either with a large number of other documents, and in this task it must first retrieve the most relevant ones, or it can compare two given documents where the one is suspected to contain passages that are derived from the other.
As (**?**) states, different systems of automatic plagiarism detection employ different techniques, but usually all of these approaches go through three main steps: 1) text-preprocessing, 2) filtering (or candidate retrieval) and 3)detection. The most severe challenges posed to automatic detectors come from the changes made to the original text. (21) lists some techniques of rewriting an original text:

- **Lexical changes**
  With lexical changes is understood the substitution of content words with respective synonyms or related concepts.

- **Structural changes**
  These changes may include a transformation from active to passive voice. Changing the word order without affecting the meaning of the sentence, like changing the order of the tokens that constitute an enumeration. Another technique is that of splitting long sentences or merging short ones.

- **Textual Entailment**
  Is the case of deriving a new, but at least semantically equivalent text, from another one. In this scenario the source text is merely refered to as *the text*, while the derived one is

called *the hypothesis*. The authors employ this term to mean paraphrasing as discussed above in 2.1.2.

(3), makes an account of the main plagiarism detection techniques employed in detection systems. These techniques differ from the unit of comparision they use, the particular feature of natural language they use as a measure of similarity and from the computational approach they utilize. The list below, taken from (3) is not complete, as we chose to mention only those techniques that are more relevant to this project:

- Syntax-based: These are techniques that use the POS tags in the representation, in order to derive a distance/similarity measure between the texts according to how similar they are in their syntactical structure.
- Cross language-based: In this method, the similarity between a plagiarised and an original document is calculated by using statistical models to establish the probability that the plagiarised document was derived by the original one, regardless of the order of the terms in these documents.
- Semantic-based: These methods use the semantic dimension of texts to derive a score of their similarity. They rely primarily on the meaning of the word (or lemmas) and employ further techniques for getting the meaning of larger texts built from these lemmas. These methods can be knowledge-free, like in the case when the meaning of a word is derived from its distribution in a corpus, or knowledge-based, when we look up the definition of a word in a dictionary.
- Citation-based: Are methods that try to find the academic documents that are obviously used in another document, but which have not been referred to. According to , these techniques are related to the semantic-based ones, since they look at the semantic content of the document in order to find out which part of it has been stolen from another document.

cite here

2.3.1 Semantic relatedness approaches

When talking about the semantic relatedness between words, we mean one of the semantic relations described in section 2.2.3. Two words can be semantically related by one of these relationships.Semantic similarity is a special case of semantic relatedness. It is the case when the words are not only semantically related but also semantically similar, which is, when they express the same meaning [10]. In computational approaches we do also talk about semantic distance, which is the reverse of similarity and quantifies how dissimilar two words are.

2.3.2 Calculating semantic similarity

write a shor
introduction
here

[10]http://en.wikipedia.org/wiki/Semantic–*similarity*

Edge-based approach

The structure of a taxonomy makes it natural to think about the distances between the concepts as the sum of the edges that one must traverse in order to get from one concept to the other. These concepts do have meanings so the distance between them would also denote how closely their semantics are related. In semantic networks, concepts could be related by a various number of properties, not only by the IS-A relation.
(29) represent the Distance metric which can be used to quantify the degree of semantic relatedness between the concepts in a hierarchical knowledge base. They argue that the more properties the concepts have in common, the more links are there between them, which means that they are closely related. While semantic distance would be path between the concepts which has the minimal length. (29) do also argue that when only IS-A relations are considered, then semantic relatedness is similar to semantic distance, since IS-A relations are based on similarity between the properties of the concepts. This approach of measuring the semantic distance between concepts by counting the edges that lie in between, is called the edge-based approach. (29) points that Distance should have the properties of a metric: zero property, symmetric property, positive property and triangular inequality.
This approach does however have some weaknesses.

Resnik Similarity

In (27), Resnik presents a method for assessing semantic similarity between words based on their representation in a semantic space like WordNet. This method embraces the suggestion of (29) that in semantic networks, similarity can be evaluated by considering only the (IS-A) links. As stated in 2.2.3, one way of using taxonomies in assessing the semantic similarity between words, is that of measuring the distance between the concepts that these words are assigned to. (27) acknowledges the problem that the edge-counting methods does not take into account the spanning of these links. In a taxonomy, there can be dense sections with a large number of links between closely related concepts as well as there can be sparse sections where a single link spans between two less related concepts. (27) proposes to evaluate the relatedness by using the view of information content by deflecting thus the dependency on link distances. It uses the intuition that two similar concepts have much information in common. This common information is represented in the taxonomy by the most specific concept that subsumes both these two concepts. The edge counting schema is also aware of this property, as the long distances would mean that we have to go up high in the taxonomy in order to find a concept that subsumes the two respective concepts we are comparing. In order to avoid edge distances, (27) supplements the taxonomy with the function $p : C \rightarrow [0, 1]$ where C is the set of concepts, such that for any $c \in C, p(c)$ is the probability of encountering an instance of the concept c. So, if $c_1 IS - Ac_2$ then $p(c_1) \leq p(c_2)$. From this follows that the p is monotonic and that the most abstract concept in the taxonomy (if this is unique) has a probability of 1.
The *information content* of a concept c is defined as the negative log likelihood $-\log(p(c))$. This form of quantifying information content gives the desirable result that while the probability increases the information content decreases,something which corresponds with the intuition that abstract concepts bear little specific information. The similarity between concepts based on this notion is than measured by the amount of information content they have in common. This is measured by the information content of the concept that subsumes these two concepts, as stated below:

$$sim(c_1, c_2) = \max_{c \in S(c1,c2)} [-\log p(c)] \tag{2.1}$$

Where $S(c_1, c_2)$ is the set of concepts that subsume both $c1$ and $c2$. Notice that although similarity is computed by considering all upper bounds for the two concepts, the information measure has the effect of identifying minimal upper bounds, since no class is less informative than its superordinates. In practice one often needs to measure word similarity, rather than concept similarity. Using $s(w)$ to represent the set of concepts in the taxonomy that are senses of word w, define:

$$sim(w_1, w_2) = \max_{c \in S(c_1,c_2)} [-\log p(c)] \tag{2.2}$$

where c_1 and c_2 range respectively over $s(w_1)$ and $s(w_2)$.

The probabilities are usually derived by large corpora, by using the following formula:
The frequency of a concepts is estimated by the counting the occurrences of the words that are subsumed by the concept.

$$freq(c) = \sum_{n \in words(c)} count(n) \tag{2.3}$$

Then the probability of the concept is computed as the relative frequency against the total number of nouns N observed.

$$\hat{p}(c) = \frac{freq(c)}{N} \tag{2.4}$$

Jiang & Conrath Similarity

(15) present a method for computing word semantic similarity where the taxonomy structure is combined with statistical evidence derived from a corpus. This approach takes off from the edge counting method and enhances this with the node-based approach. Firstly, it accounts for the fact that the link between nodes should be weighted. In particular, it determines the strength of the edge that links a parent node to a child node. It starts with the argument that the strength of a child link is proportional to the conditional probability of encountering an instance of the child concept c_i given an instance of its parent concept $p : P(c_i|p)$. Further it defines the link strength (LS) by taking the negative logarithm of the probability, hence obtaining the formula:

$$LS(c_i, p) = -\log(P(c_i|p) = IC(c_i) - IC(p). \tag{2.5}$$

So utilizing the notion of information content, the link strength is defined as the difference between the information content values of the child concept and its parent concept. The paper provides the derivation of the overall edge weight for a child node and its parent, as well as the distance between two nodes when this derivation is set in the distance formula. But the implementation of the algorithm for our project does not employ these fine grained calculations. The distance used in our work is that described by the following formula:

$$Dist(w1, w2) = IC(c_1) + IC(c_2) - 2 \times IC(LSuper(c_1, c_2) \tag{2.6}$$

where $LSuper(c_1, c_2)$ denotes the lowest superordinate of c_1 and c_2.

Semantic Similarity Between Sentences

The approaches described above compute the semantic distance between two single words. But in the task of plagiarism detection we are concerned with, it is desirable to consider the semantic similarity between sentences. Here we describe the algorithm from (26) which uses one of the metrics for word to word similarity and a measure for the specificity of words to compute the semantic similarity between two text segments, which we call sentences, $T1$ and $T2$. The algorithm, which we shall later refer to as the *aggregator* follows this procedure:

1. For each word in sentence T1 determine which word of sentence T2 gives the highest semantic similarity when compared with the given word. The similarity score is derived by using an algorithm that computes semantic similarity between words. The function words are discarded and only open-class words and cardinals are considered in the process.

2. Repeat the same procedure for each word in sentence T2.

3. Weight the results with the corresponding word specificity, sum up and normalize with the length of each segment: The word specificity is computed by using the inverse document frequency (idf), which is defined as the total number of documents in the corpus divided by the total number of documents including the word in question (31).

$$sim(T1,T2) = \frac{1}{2}\times(\frac{\sum_{w\in T_1}(maxSim(w,T_2)\times idf(w))}{\sum_{w\in T_1} idf(w)}+\frac{\sum_{w\in T_2}(maxSim(w,T_1)\times idf(w))}{\sum_{w\in T_2} idf(w)}) \tag{2.7}$$

The result is a number between 0 and 1, where 0 denotes perfect dissimilarity and 1 denotes perfect semantic match. If the algorithm used is a knowledge-based one it can happen that it doesn't compute the similarity between words with some given POS tags,like adjectives or adverbs. In this case the *aggregator* uses a lexical similarity measure which assigns a *maxSim* of 1 for exact matches of the words in the two sentences.

2.3.3 The Longest-shared-passage problem

This problem is formulated like this: given a document that contains plagiarised text and another one from which the plagiarised passage is derived, how to detect which passage is plagiarised from which passage when the detection algorithm returns a score in a word/sentence level?. In this case, with passage we mean a text that is longer than a sentence. The figure below is used to illustrate the case:

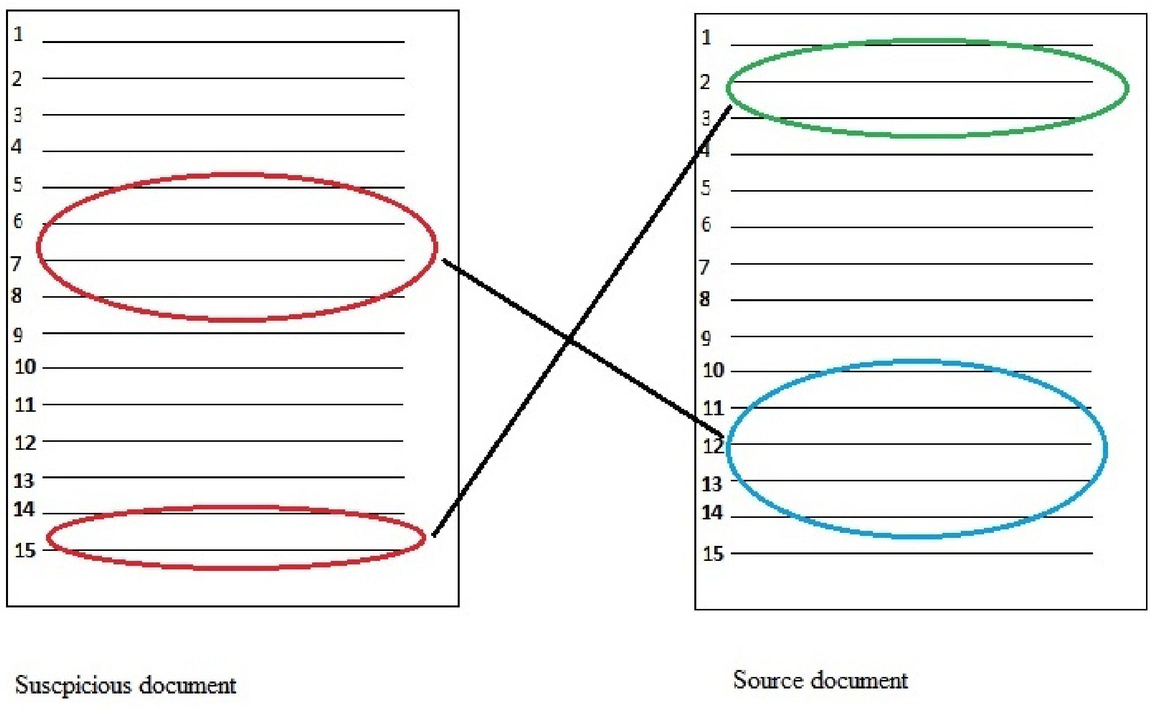

Figure 2.3: The Longest-shared-passage Problem

If the detector rightly reports every occurrence of plagiarised word/sentence, then this problem is problem could be reduced to the question of how to merge the detected elements in a way that the result constitutes in a whole paragraph.

2.4 Performance evaluation

As mentioned in section 2.1.3, the PAN challenge offers a corpus of documents for both training and testing a plagiarism detector. The corpus consists of a set of suspicious documents, hereafter referred as D_{susp}, a set of source documents, D_{src}, and sets of XML document-pairs listing the suspicious and source documents that are related to each other, with pointers to the plagiarised passages. As stated in (20) the majority of the systems that were delivered to be tested on the task, implement three basic steps which are shown in the **??** taken from (7). These steps are seen as standard for all the detectors.

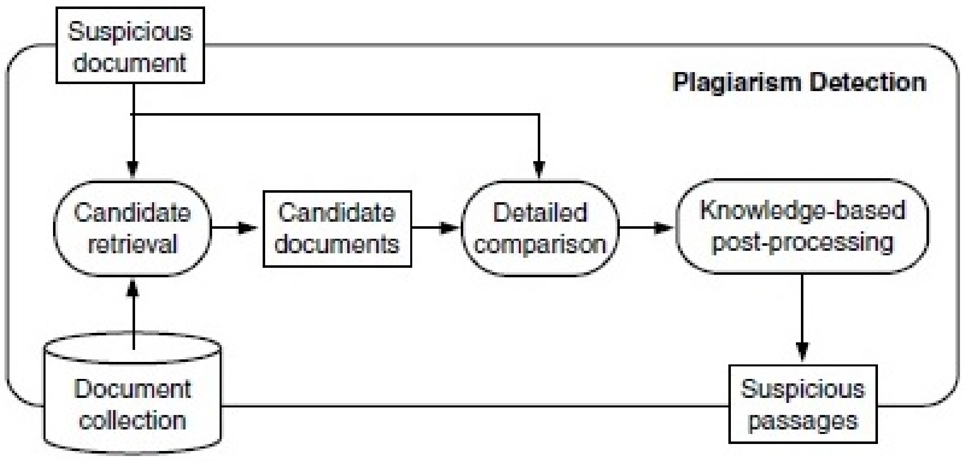

Figure 2.4: Generic retrieval process to detect plagiarism (Figure taken from (20))

These steps consist in:

1. Candidate retrieval, which is the process of selecting a set of documents from the source documents corpus that most likely correspond to the source documents from which the suspicious documents have been derived.
2. The detailed comparison phase where each retrieved source document is compared to the plagiarised document, and passages of high similarity are extracted.
3. In the last phase the results go through a kind of post-processing where they are cleaned and filtered.

Before, the detectors were evaluated as a whole in their performance, but since many contestants were dropping the candidate retrieval step, given the relatively small amount of source documents available, the evaluation was changed to a stepwise one, where the candidate retrieval and the detailed analysis are judged separately. Here the focus shall be only on the measures used for the detailed analysis stage.
First, it is important to give a summary description of the text corpus which is used as input in training and in a possible cross-validation. This corpus contains documents grouped into two categories: suspicious-documents and source documents, where the suspicious documents are derived by obfuscating the source ones by the methods described in Section 2.1.3. It does also contain sets of result XML documents, similar to those that a detection system conforming to the PAN criteria should produce as a result, grouped by the kind of plagiarism that they point to. These result XML documents are used as a blueprint when evaluating the performance of a detector.
The plagiarism disclosed can be one of these cases:

- No-plagiarism. These documents point exactly to the passages that have nothing to do with each other.
- No-obfuscation. These are the passages that are copied verbatim.

- Random obfuscation. These passages are obfuscated by using artificial plagiarism techniques and are of the kind shown in 2.1.3.
- Translation obfuscation. These are passages that are obfuscated by the method of back translation.
- Summary obfuscation. These are the ones that belong to the case of paraphrasing. The sentences of a source document may be split into a number of shorter ones, or a small group of shorter sentences may be merged to form a longer one.

In 2012 (20), the organizers of PAN changed their submission requirements from that of delivering submission results, to delivering the detection software itself. This was justified by the , as by getting hold of the system one could check the runtime, could give as input real plagiarism cases and it could be later tried on other datasets ,as for example that of PAN13. This last thing allows for a more fair comparison between the systems that were originally tested on different data sets. Since the participants were allowed to deliver systems based on whatever programming language, the The TIRA (35) platform was used for dealing with the complexity of the organizational issues. TIRA was also used in the training phase, as it returned the performance value for the submitted results after the system was run on the corpus described above.

In order to measure the performance of the detector, PAN provides a quantifier called *plagdet*, which combines precision, recall, the F-measure and a measure that designates the granularity of the results. Since plagiarism is usually carried out in whole paragraphs, the granularity score tells if the detector fragments the whole paragraphs into smaller passages and whether these reported passages overlap or not. R is the set of plagiarised passages reported from the detector while S is the set of plagiarism cases in the corpus.

Granularity measure:

$$gran(S,R) = \frac{1}{S_{\mathrm{r}}} \sum_{s \in S_{\mathrm{r}}} \mid R_{\mathrm{s}} \mid \tag{2.8}$$

The *plagdet* measure combines the scores mentioned above, together with the F-measure, denoted below, in order to provide a unique classification of the participant's results.

$$plagdet(S,R) = \frac{F_1}{log_2(1 + gran(S,R))} \tag{2.9}$$

F-measure:

$$F_1 = \frac{2}{\frac{1}{r} + \frac{1}{p}} \tag{2.10}$$

The pros and cons about the *plagdet* score and a more thorough evaluation of this measure are to be found in (20).

Chapter 3

Related Work

This chapter presents works that are somehow related to this project. Since the main novelty we brought to the existing detector was that of integrating semantic similarity between sentences, the first section does discuss two other approaches to detecting similarity between sentences. The second section is concerned with the creation of datasets used for training and evaluating semantic similarity detectors. The third section provides a short description of the team that scored best in the ”Text Alignment” task, in PAN13.

3.1 Detecting Similarity Between Sentences

3.1.1 SimPaD

(Nathaniel Gustafson) present a plagiarism detection method called *SimPaD*. This method is designed for finding similar passages between document pairs and is based on finding similarity between sentences. Here we will present only the derivation of similarity between two sentences, not between documents.
The *SimPaD* method accounts for synonym substitution, word addition/deletion and the fact that the plagiarised sentence could be a result of merging or splitting. But not all the sentences are considered. Based on the assumption that identical short sentences are more probable to appear in texts that are not plagiarised from one another, these short sentences are removed from the analysis input. In order to define the similarity between sentences, the approach first applies a similarity-match between the single words. The stopwords are removed and do not participate in the process. In the case when the sentences share a large number of lexically identical words, they are easily defined as plagiarised. When the content words are substituted by synonymous ones, the algorithm derives the similarity score by comparing their meanings. The larger the number of words that ware similar meanings the higher is the similarity score for the two sentences. In the case of word deletion/addition, the algorithm searches in the ”opposite” sentence for words that share the same meaning with those added to the sentence under suspicion. The semantic similarity between lexically different words is derived by using word-correlation factors. These scores are produced by using the Wikipedia collection.
Another thing to be mentioned is that the algorithm does not take into account the changes in the sentence structure, so the word-reordering does not have any weight in the calculation process. To make up for this, the algorithm uses n-gram correlation factors, for $n = 2$ and $n = 3$. After

having calculated a similarity value for each word pair, the algorithm computes the similarity for each sentence pair by a function that adds these particular word-word scores and delivers a result between 0 -no similarity and 1 - identical sentences.
To summarize, this method accounts for both lexical matching and semantic relatedness between two sentences.

3.1.2 Dao & Simpson

This subsection treats the work of (Thanh Ngoc Dao) on measuring similarity between two sentences. Their approach is based on comparing the semantic content of two input sentences. The input must first undergo a preprocessing phase which consists of 1)tokenization 2)stemming 3) POS-tagging. Afterwards, there is taken a word sense disambiguation procedure for each word in the two sentences. (**?**) first consider the Michael Lesk algorithm for WSD (18). This algorithm is based in the dictionary definition of senses and decides the most appropriate sense of a word in a given sentence, by comparing the gloss of each of the senses of the given word with that of each of the senses of every other word in the sentence. Then the algorithm chooses as the proper sense the one whose gloss has most overlapping words with another sense gloss.
(Thanh Ngoc Dao) do however use what they call, ”an adapted version” of the algorithm which does no longer perform WSD for a single word by using all the other words in the sentence, but by choosing only a limited number of words (say k words) both to the left and to the right of the word in question. It does moreover use WordNet synsets in the disambiguation process. Besides considering the word’s gloss, it also considers the glosses of the synsets that are connected to this word’s synset by these relations: 1) hypernymy, 2)hyponymy 3) meronymy and 4)troponymy. After having gathered all these gloss-pairs, the algorithm performs a similarity match between these glosses in order to find the pair that reports the highest overlap between the two glosses involved. After having assigned a sense to each word, the next step in the process is that of defining the pairwise similarity of these word-senses. WordNet is again used in this step as the similarity is defined by measuring the path length from one synset to the other. A word-word matrix is built where the cells denote the similarity between two words.
The two sentences are then transformed to two sets of disjoint nodes which constitute a bipartite graph. Finding the semantic similarity between these sentences is than transformed to calculating the maximum total matching weight of this graph.

3.2 Semantic Similarity in WordNet

3.3 Evaluating the Detection of Sentence Semantic Similarity

There exist many methods that approach the problem of disclosing similar texts by applying semantic similarity measures. SemEval http://www.cs.york.ac.uk/semeval-2012/ is a workshop which evaluates systems that detect various semantic traits in texts. STS, short for Semantic Textual Similarity, is a task presented in SemEval 2012, which raises the challenge of finding how semantically similar two given sentences are. The expected output from a detector system is a graded similarity value ranging from 0 - no similarity, to 5 - identical sentences. STS provides a unified framework for testing of various semantic modules that otherwise have been tested in isolation from each other. The set of these different modules is described in (13). One

of the main points of STS was the construction of training and testing datasets for this kind of task. As mentioned also in chapter 2, it is not easy to obtain a large dataset of plagiarism cases. The smallest dataset STS bestows is the one containing 65 sentence pairs whose scale of similarity is annotated by humans. This set is however to small for the purposes of the task. A larger one was that of 50 documents whose pairwise similarity was also assessed by human users, but though a bit larger, this dataset does not address the problem of sentence- sentence similarity.
The main datasets used from STS are those provided by the Microsoft Research. The first one, MSR Paraphrase which contained cases of paraphrasing annotated by binary tags was enriched by STS with tags that reflected a similarity score from 0 to 5. The second MSR dataset was the MSR Video, which is constructed by letting human annotators from Amazon Mechanical Turk (AMT) watch the same video and write down a sentence or a few sentences that describe the semantics of the video. There were used 2000 videos which produced 120 thousand sentences. The dataset created was divided into two parts: in the first one, they put all the sentence-pair combinations for a given video, while in the second is put sentence pairs that come from different videos. It is however not given that different annotators do not denote the same video with semantically contradictory descriptions or, likewise, that different videos do not get descriptions with similar semantic content.
The STS task in SemEval 2012 had 35 teams which delivered 88 runs of their systems. Besides the similarity scores, the team was allowed to assign a confidence score in addition. This second score did however not play any role in the final evaluation. STS provides an evaluation baseline, which derives the results by using a simple word overlap metric between the sentence pairs that were represented as vectors in a multidimensional token space.
STS does provide three different evaluation metrics:

3.4 PAN13 Participants

The plagiarism detector we present in this project was originally built for the ”Plagiarism Detection” task posed by PAN10 and PAN11. We have further adapted the system, so that its functionality can be tested by the dataset of PAN13. PAN13 does however present a different challenge by splitting the ”Plagiarism Detection” into two subtasks: ”Source Retrieval” and ”Text Alignment”. We are only concerned with the second one. (**?**) presents in his ”Related Work” chapter, a description of the most successful participants of PAN11. We do here present the method from

3.4.1 •

Chapter 4

A Detector Based on Graph Edit Distance

In chapter 2 we mentioned the graph-based representation of sentences. Here we discuss the plagiarism detection system described in (28) whose detection algorithm is based on this representation. A brief description of the system: it reads files from a corpus and processes them with NLP techniques presented in section 2.2. It then reduces the plagiarism search space by doing document retrieval on the basis of the post-processed format and undertakes a detailed plagiarism analysis on the relevant documents.Finally the system produces XML formatted files that contain information about the plagiarised passages.

4.1 The Plagiarism Detector Created by Røkenes

The system is intended for use on the datasets published as a part of PAN10 and PAN11 [1]. The input corpus is divided into original source documents and suspicious documents which are a production of the artificial plagiarism techniques performed on these source documents. These techniques were described in chapter 2. The preprocessing step takes as input the plain documents of the dataset. These documents are then partitioned into sentences, discarding thus the document structure. They are sent afterwards to the preprocessing pipeline that consists of these steps:

1. POS-tagging
2. Lemmatising
3. Dependency Parsing

The POS-tagging used is the *english-left3words-distim.tagger*, one of the models of the Stanford Part-Of-Speech Tagger[2]. After the lemmatising, the sentences are turned into dependency graphs. The dependency graphs are created by the MaltParser[3], which is data-driven [4].

[1]http://pan.webis.de/
[2]http://nlp.standford.edu/software/tagger.html
[3]http://maltparser.org
[4]http://www.delph-in.net/courses/07/nlp/Niv:Hal:Nil:07.pdf

The graph-sentences are put into a MongoDB graph database [5]. After these steps, each sentence in the database exists as a list of lemmatised tokens.Each token is annotated with a POS-tag and an integer *id* that denotes which token in the sentence it is dependent on, and the name of this dependency relation, which is represented by the *deprel* field. In the case when there is no dependency, for example when the token is the root of the sentence, this *id* is set to 0.
So when discussing ”a sentence” from this point in the text,we refer to a structure such as the one in the example below taken from the PAN13 dataset:

```
"id": ObjectId( "51ec1e1ee4b029dcab101afc" ),
"id": "suspicious-document00006.txt-135",
"filename" : "suspicious-document00006.txt",
"sentenceNumber": 135, "offset": 14156,
"length": 112,
"tokens": [  "id": "1", "word": "In", "lemma" : "in", "pos": "IN",
"rel": "8", "deprel" : "prep" ,  "id": "2", "word": "addition",
"lemma": "addition",
"pos": "NN", "rel": "1", "deprel" : "pobj"]
```

As it is argued in (20) doing an exhaustive all-to-all comparison does increase the recall of the result but the strategy is far from optimal or even feasible, given a set of large source documents such as the web. In order to reduce the amount of sentences that each suspicious sentence is compared to, the system performs a ”Candidate Retrieval” phase, during which a number of source sentences are discarded as not relevant to getting analysed against the suspicious ones. After the retrieval step, the database elements have this form:

```
"source_file: source_documentxxxxx.txt, suspicious_sentence: xxx,
 candret_score: x.x, source_sentence: xxx,
 suscpicious_file :"suscpicious_documentxxxxx"
```

This is the format of the input handed over to the ”Detailed Analysis” phase whose main processing steps are pictured below:

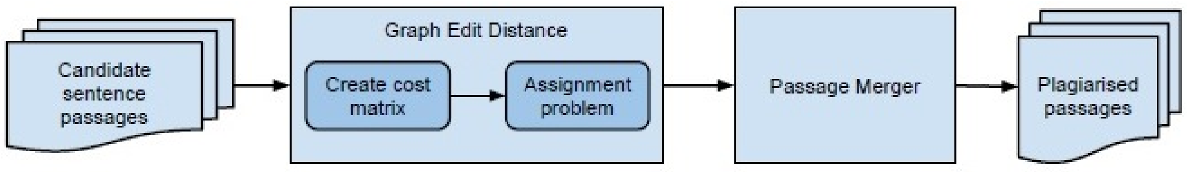

Figure 4.1: Overview of the detailed analysis phase. (Taken from (28))

As the system is based on graph representations and dependency parsing whose scope is limited to the sentence, the comparison algorithm returns a distance value that denotes the lack

[5]http://www.mongodb.org/

of similarity between two sentences. The main idea of the detection strategy is: The Graph-Edit-distance algorithm decides the distance between each pair of source sentence and suspicious sentence and uses this information to report for a suspicious document, which passages are plagiarised from which passages of one or more source documents.
As the internal workings of the program are described in (28), we shall not delve into many details here. It is important to mention that the distance calculation between these graphs is reduced to an assignment problem for a cost matrix. The cost matrix itself is a word-word matrix for a sentence pair, where the values in the cells represent the distance between two words that come from a suspicious sentence and a source sentence. If the two lexemes compared are identical and their *deprel* and *POS* values the same, they are acclaimed to be perfectly similar. When this is not the case, the following formula is used to calculate the distance value between two nodes,

$$SC_{n,m} = \frac{NLD_{n,m} + ED_{n,m}}{2} \times S \tag{4.1}$$

where *n* and *m* are two nodes, *SC* stands for the substitution cost, *NLD* stands for Node Label Difference, which is the difference between the *POS* values, *ED* stands for Edge Difference, the difference between the *deprel* values, while *S* is the highest possible substitute cost given as input, which in this case was set to 2 (two). The sum of $NLD_{n,m} + ED_{n,m}$ is a number between 0 and 1. The system accounts for the weighting of the differences between *POS* labels by taking as input a table with these weights. But it does not take into consideration the difference in *deprel* relations, a possible deficiency that was pointed out by the author (28).

The picture below shows the main building modules of the system created by (28)

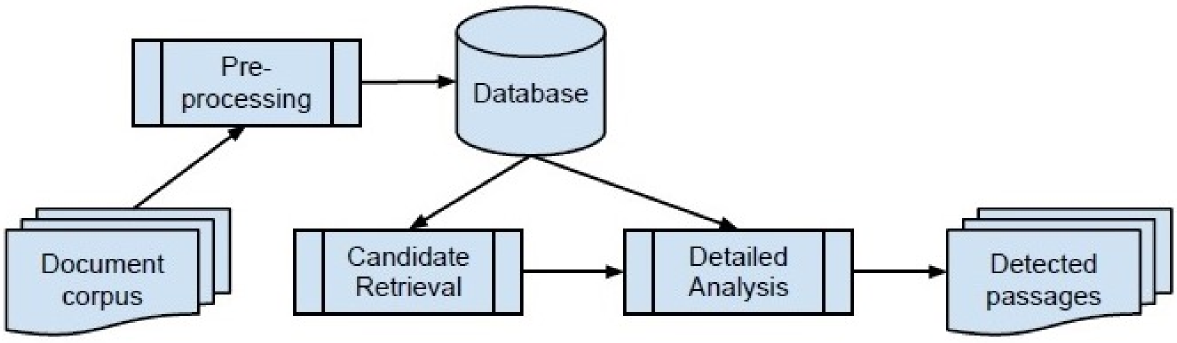

Figure 4.2: The overall architecture of the system. (Taken from (28))

Fig. 4.2

4.2 Results and Deficiencies

When tested on the PAN11 dataset, the ”Detailed Analysis” phase of the system gives this result:

Plagdet	Recall	Precision	Granularity
0.224	0.223	0.479	1.56

This result would place the system in the fourth place among the participants of PAN11 if we consider only precision and recall. The system does however exhibit some deficiencies which are discussed below:

1. It does often not detect the sentences that lie near an already judged-as-plagiarised sentence as part of the plagiarised passage, even though they in reality are. (28) argues that the approach taken by the PassageMerger class, by gluing together all sentences that are within a given distance, called *mergedist*, in order to reduce the *granularity* which is held as an important trait by the PAN challenge, does not solve this problem. These sentences could in the first place have been left out of the ”Detailed Analysis” phase by the ”Candidate Retrieval” phase. (28) thus proposes, to lower the threshold of similarity required, for the sentences in the original suspicious documents (not the truncated ones) that lie after a detected sentence to classify them as plagiarised. The problem could also be solved by increasing the recall of the ”Candidate Retrieval” case. We also discuss another solution in chapter 5.

2. It does not account for the semantics of the texts. The example below shows how the algorithm gives a low distance for a sentence pair with completely different semantics and a higher distance for a pair with different syntax but identical semantics. The examples above show the results of running the GED.java class which gets two sentences as input and returns their distance which is calculated by the Graph-Edit distance algorithm:

   ```
   John kissed Mary.
   John killed Mary.
   ```

   GED for the two graphs: 0.083
   Edit-path:(kill $\rightarrow$ kiss) = 0.25

   ```
   John kissed Mary.
   Mary was kissed by John.
   ```

   GED for the two graphs: 0.625
   Edit path:
   (john $\rightarrow$john) = 0.5
   (mary $\rightarrow$ mary) = 0.5
   (? $\rightarrow$ be) = 1.0
   (? $\rightarrow$ by) = 0.5

   As it can be seen, the algorithm gives more importance to syntactic changes in the sentence. A hypothetical help to this problem could be to differentiate between the possible substitutions between dependency relations. This is discussed further in chapter 5.

Chapter 5

Improvements to the Detailed Analysis Algorithm

This chapter describes the details of the implementation done in order to validate our theoretical assumptions. The main practical goal of the implementation was to enhance the performance of the system described in chapter 4 by addressing its deficiencies which were listed in section 4.2. In order to achieve this we needed to thoroughly understand the existing codebase and adapt it to the new environments and tasks, such as those required for the PAN13 shared task.
The first approach towards improving the detailed analysis phase was to adjust the weights to the dependency relation substitutions. The second approach was to combine the largely syntactic sentence distance measure with one incorporating semantic information. Finally, we implemented a new strategy for merging of detecting in order to reduce the granularity of the produced results.

5.1 Weighting Dependency Relation Substitution

As stated in section 4.2, the original system does not take into account the impact that the change of dependency relations could have on the derivation of different types of the distance between sentences. In order to measure the impact of these weights we tried to decide the degree of difference between these relations taken pairwise. A distance measure is used to denote the difference between two relations, where 0 denotes identical roles and 1 denotes a maximal difference.
This process was guided by the definition and description of these relations found in (19). The weights were estimated by answering for each pair of relations the following questions: ”How much does the structure of the sentence change if the actual dependency relation is transformed into the one we are comparing it to?” and ”How much does its meaning change in this occasion?”. It should be noted that the degree of how much a sentence has changed is arbitrarily defined. Take as an example the case when the *advmod* relation, short for adverbial modifier, that modifies the meaning of a word, when it is substituted by the *amod* relation,which is the adjectival modifier of a noun phrase. This

is illustrated by the example below:

```
"Genetically modified food"    advmod(modified, genetically)

"Sam eats red meat"            amod(meat,red)
```

Trying to transform the first sentence into a semantically similar one where the word *genetically* is associated with *amod* relation, it would result in a sentence like:

```
The food is modified by genetic techniques.            .
```

This leads to the conclusion that a word connected by the *advmod* relation in a sentence could possibly get assigned an *amod* relation, when the sentence is changed. This substitution is weighted by 0.25.
It is, however, rarely the case that words get assigned new dependency relations without requiring significant changes to the entire sentence. A quick look at the table of weights (this is the deprel-sub.csv file in the *resources* folder in the attachment delivery), shows that it is not usual for a word to get another grammatical role in the sentence when this is transformed to an equivalent one by the help of syntactic transformations.

5.2 Excluding the Candidate Retrieval phase

In 2013, PAN changed the ”Text Alignment” task such that an implementation of the Candidate Retrieval phase could be omitted, given that one would integrate in the system the *pairs.txt* file included with the rest of the training corpus. This text file holds a list of ”suspicious-document” – ”source-document” pairs, which tells for each suspicious document, which source documents it must be compared to. This reduces the search space for the detailed analysis that would otherwise have to make an all-to-all comparison or, as in the approach of (28), to implement a candidate retrieval phase. This phase could however leave out relevant documents as well as include the wrong ones. In order to adopt this file we have added the *getSentencesFromPairs()* method in the PlagiarismSearch.java class. This method reads the file and uses its information to generate sentence pairs and puts them into the queue of the jobs to be processed in the next step. A job is simply a pair of documents, the suspicious one and the source one. Moreover, the method matches all sentences from the suspicious document with all sentences from the source document. The sentences are then transformed into graphs. The format in which the detailed analysis phase expects the input to be is thus preserved. In this way, the second phase, that of retrieving candidates, is eliminated. The figure below shows the system after this package is removed.

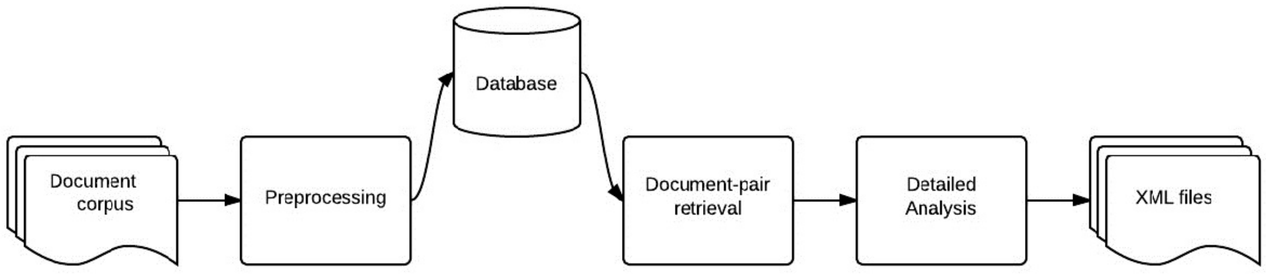

Figure 5.1: Candidate retrieval phase is removed

5.3 Semantic Distance

DKPro Similarity is a framework for text semantic similarity, which in its standalone mode offers a repository of text similarity algorithms that can be integrated even in projects that have a different preprocessing platform from that of the DKPro.
DKPro was integrated with the system of (28), by using the LSR (Lexical Semantic Resources)[1] package. The reason for choosing this package is that of wanting to compare the semantic dimension of texts. The module offering the LSA (Latent Semantic Analysis) would maybe work as well but there are two reasons for not having chosen that: 1)It would require much more recoding of the base system since its unit of comparison is the sentence and 2) it is better suited to analyse the semantic content of whole documents, while PAN, as discussed in Chapter 2, is using artificially obfuscated documents that in fact have no meaning when considered as a whole piece.
There are several subpackages inside the LSR package: the *aggregate*, the *gloss* and the *path* package. We have been using two of these three packages. The *path* package contains the implementation of among others the Resnik and Jiang & Conrath algorithms discussed in section 2.3, while the *aggregate* package contains an implementation of the *aggregator*, described in section 2.3.2. The implementations of these three algorithms bear, however, some deviations from the theoretical work of their authors. ResnikComparator.java [2] uses intrinsic informacion content (IC) that relies on the hierarchical structure of the taxonomy alone, instead of corpus-based IC as it was proposed in the (27) paper. The implementation of Jiang & Conrath does also use the intrinsic IC. Resnik and Jiang & Conrath were designed to measure semantic distance between words, while DKPro implementations measure semantic similarity. The *aggregator* [3] does also measure semantic similarity. In our implementation, we have reversed the result of the aggregator to measure the distance between texts by subtracting if from the maximum value that could

[1]https://code.google.com/p/dkpro-similarity-asl/source/browse/trunk/de.tudarmstadt.ukp.similarity.algorithms.lsr-asl/src/main/java/de/tudarmstadt/ukp/similarity/algorithms/lsr/?r=257

[2]https://code.google.com/p/dkpro-similarity-asl/source/browse/trunk/de.tudarmstadt.ukp.similarity.algorithms.lsr-asl/src/main/java/de/tudarmstadt/ukp/similarity/algorithms/lsr/path/ResnikComparator.java?r=257

[3]https://code.google.com/p/dkpro-similarity-asl/source/browse/trunk/de.tudarmstadt.ukp.similarity.algorithms.lsr-asl/src/main/java/de/tudarmstadt/ukp/similarity/algorithms/lsr/aggregate/MCS06AggregateComparator.java?r=257

be produced by the calculation , which is 1.
First we tested Resnik and Jiang & Conrath separately by giving pairs of words as input. Then there was built a test class around the *aggregator*. Since the *aggregator* uses one the algorithms in the *path* package to derive similarity between words, the Resnik and Jiang & Conrath were used interchangeably during testing. The idf values for the *aggreagator* by using the index built from the *Wesbury Lab Wikipedia Corpus* [4] which is a corpus of preprocessed Wikipedia documents from 2010.
The other similarity measures in the LSR package that used WordNet as a resource were used to derive a combined score for the distance between two words. This combined score is simply the arithmetic mean of the comparators: ResnikComparator, JiangConrathComparator, LinComparator, WuPalmerComparator. Now we show the performance of these algorithms on the examples taken from section 4.2:

```
Sentence 1:
John kissed Mary

Sentence 2:
John killed Mary
```

GED	Resnik	Jiang Conrath	Combined Measure
0.083	0.083	0.08	0.078

```
Sentence 1:
John kissed Mary

Sentence 2:
Mary was kissed by John
```

GED	Resnik	Jiang Conrath	Combined Measure
0.25	0.0	0.0	0.0

As it can be seen, the semantic distance algorithms rightly assign a 0.0 distance between these two semantically identical sentences.

The following sentence pair is taken as an example from (26). We show it to illustrate the behaviour of GED on paraphrased sentences and to test our part of the implementation of the *aggregator*, which is the derivation of the idf values:

```
Sentence 1:
When the defendant and his lawyer walked into the court,
some of the victim supporters turned their backs on him.

Sentence 2:
When the defendant walked into the courthouse with his
attorney, the crowd turned their backs on him.
```

[4]http://www.psych.ualberta.ca/ westburylab/downloads/westburylab.wikicorp.download.html

GED	Resnik	Jiang Conrath	Combined Measure
0.5	0.4	0.4	0.4

Surprisingly the difference between GED and the semantic distance algorithms is not too large. Moreover, (26) reports a score of 0.8 in similarity (or 0.2 in distance) while our distance result is 0.4. This deviation from the expected score can be attributed to the derivation of idf values and/or the specific similarity measure used by (26) which is using a combined measure where other measures are used.

5.4 Combining GED and Semantic Distance

When deciding the integration of the semantic distance measurement into the system as a whole, we decided to calculate for every sentence pair that enters the detailed analysis phase, a simple arithmetic mean of GED and the *aggregator*. Since the sentences in the database were represented as graphs and the *aggregator* expects two lists of lemmas, the input had to be transformed to the latter form. The first idea was to drop the *aggregator* and integrate one of the algorithms from *path* directly in the original GED matrix by deriving the distance between words as an arithmetic mean of GED and Resnik/Jiang & Conrath for each word-pair. This idea was dropped based on the argument that since the detailed analysis is based on comparing whole sentences, we should also add a measure that captures the distance between sentences.

5.5 Detecting whole plagiarised paragraphs

One of the questions arisen from (28) is that of the precise detection of whole plagiarised passages. The case of a plagiarised sentence that is isolated between many original sentences is quite unusual, so the task of detecting plagiarism is not that of picking out plagiarised sentences and their respective source, but that of pointing to whole passages that are derived from whole source passages. The picture below is a simplified illustration of a real example taken from the PAN 2013 dataset:

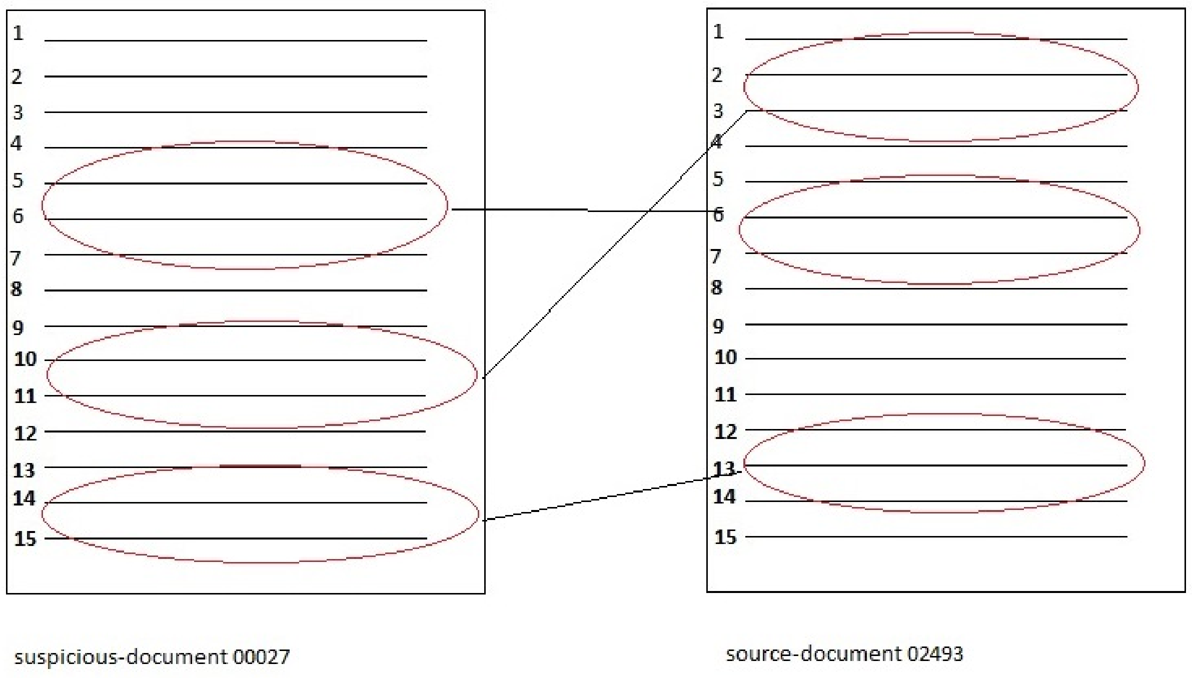

Figure 5.2: The passage detection problem

This document pair belongs to the group generated by random obfuscation. In its original form, the system detects plagiarism references by doing the core sentence to sentence comparison. A plagiarism reference is a structure that holds the information about the offset and the length of both the plagiarised text and source text. The list of references detected for a document pair is then sent further to a merging function which merges these references arbitrarily, by using the only criteria that they should not lie farther apart than a distance of 1500 characters. This approach merges together all detected passages including the passages in between which are not estimated as plagiarised by the Graph-edit-distance algorithm. These not-detected passages could or could not be plagiarised however, so including them doesn't necessarily lower the precision. Following the example above, if the algorithm hasn't detected that sentence 5 is plagiarised while both sentence 4 and 7 are detected as such, than the merging strategy pursued would correctly include sentence 5 in the block by making thus up for the deficiencies of the algorithm. The picture above shows the correct answer which is also expressed in numbers. The length and offset variables are those described above:

```
Document 00027                  Document 02493

Offset  Length                  Offset  Length

2811    262                     2027    417
3814    1093                    0       1363
```

```
4908     487                            2614     694
```

The XML file generated by the system, when using both the semantic distance and that derived by GED gives these corresponding values:

```
Document 00027                          Document 02493

Offset   Length                         Offset   Length

3906     1346                           135      2716
2967     2374                           408      2817
```

It seems like this approach doesn't account for the problem of overlapping references. This can be illustrated by the drawing below:

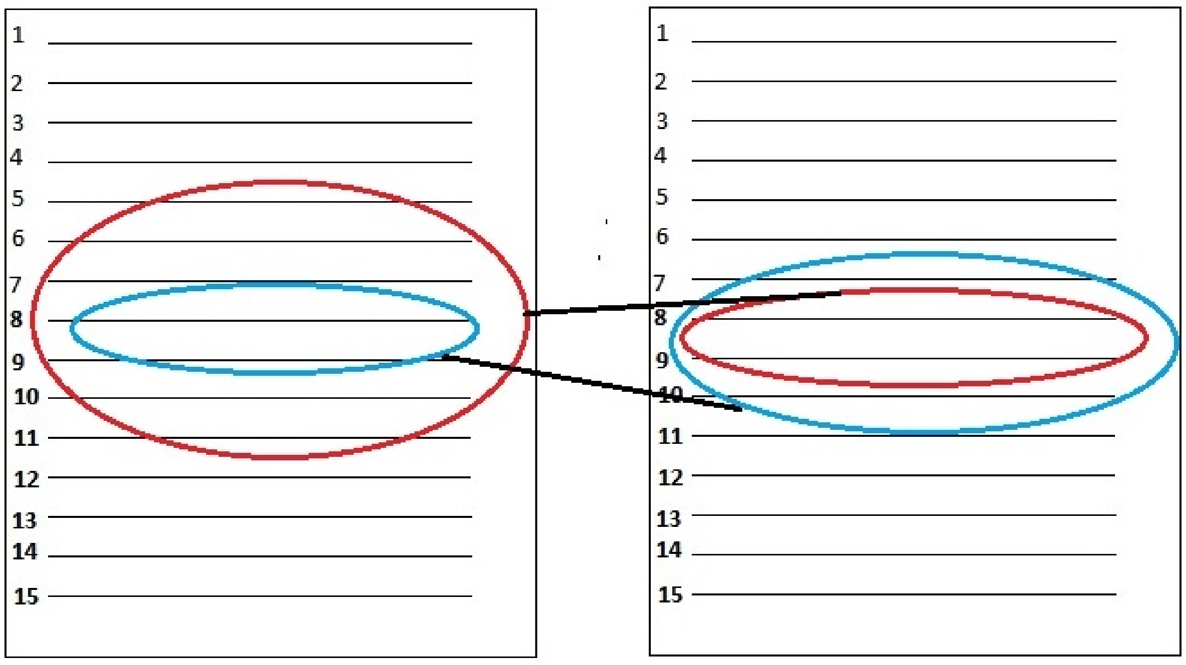

Figure 5.3: Document representation of 00027-02493 XML result

Besides the missing recall, the inner circles in both the suspicious and the source document are quite redundant and misleading. It is obvious that the kind of plagiarism that this solution presents is not a possible scenario.

Our suggestion to an improvement of the merging strategy is that of treating document pairs as matrices, where the suspicious sentences are represented by the number of the rows and the source sentences are represented by columns. The elements of the matrix define the value of the distance between two given sentences.The figure below shows the case of the example above formulated as a matrix.

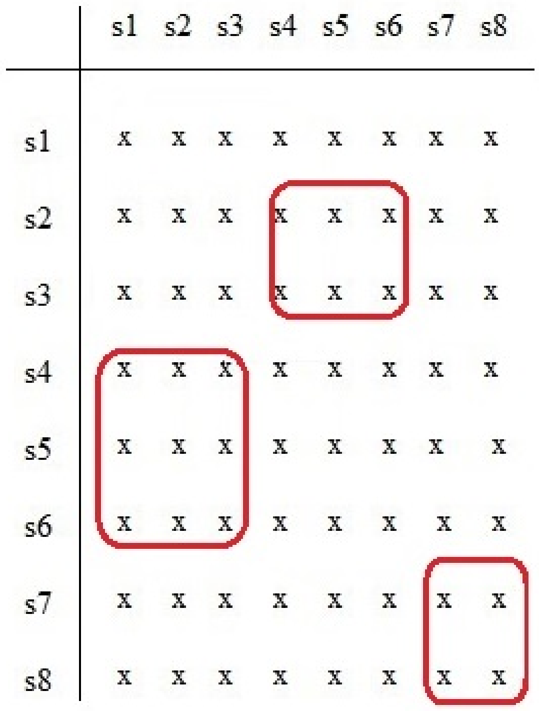

Figure 5.4: A case of random obfuscation in matrix representation

If we do not apply any merging to the detected elements, the algorithm would output a large number of plagiarism references, one for each suspicious sentence–source sentence pair. But this approach is already aware of not producing overlapping results since the sentences are now given as input in ascending order and the comparison is done corresponding to this order (i.e. suspcicious sentence 1 to source sentence 1, suspicious sentence 1 to source sentence 2, etc.). While considering the merging strategy, we went back to the question of how does one plagiarise text and how are these scenarios translated to a matrix format. We took in consideration these methods:

- Strictly diagonal case: Suspicious sentences are plagiarised from at most one source sentence. This is the 1-1 scenario which is always true in the case of verbatim plagiarism. We call this the strictly diagonal case as the plagiarised elements would lie in a diagonal line in the matrix.
- Horizontal case: The plagiarised sentence is the product of more than one original sentences.
- Vertical case: A group of plagiarised sentences have their origin in only one source sentence.
- Quadratic case (or the mosaic): Some source sentences have been transformed into some suspicious sentences. This is the case of paraphrasing.

Another way of discovering how plagiarism is carried is by inspecting the different classes of plagiarism employed by PAN. Speaking in these terms, we can refer to the different classes of plagiarism : no-obfuscation, random-obfuscation, translation-obfuscation, summary-obfuscation, in order to gain insight into the techniques used. Hopefully, these cases do not lie far from the real occasions of plagiarism.
In no-obfuscation (verbatim) plagiarism, blocks are simply copied and pasted in the suspicious document. Perfect lexical matching between the sentences in the corpus would give perfect results if we had kept the same sentence structure. The drawback here comes from the way we partition the documents into sentences. Even though we distinguish between several cases, the diagonal strategy lies at the bottom of them all. Consider the case when a suspicious sentence is a product of several source sentences, something that would coincide with the summary obfuscation data set. The matrix of this case would then have this form:

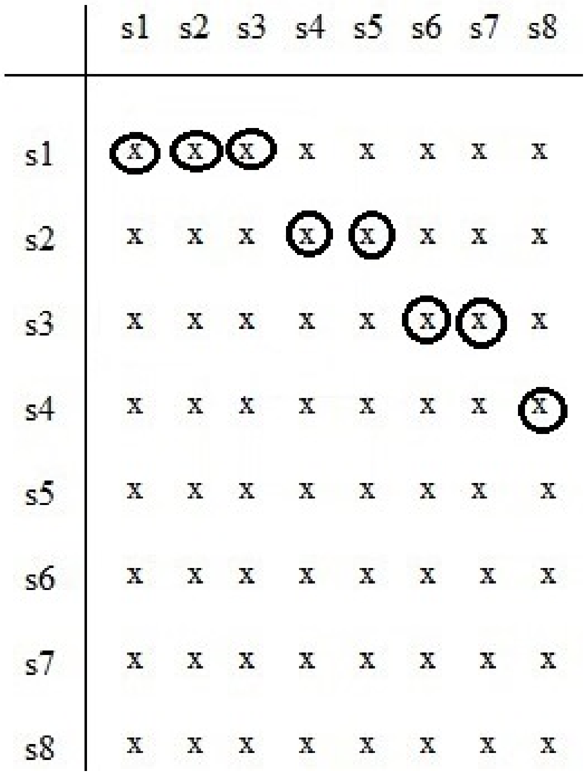

Figure 5.5: A case of shrinking paragraphs

The opposite case would be that when a bunch of suspicious sentences have sprung from one single source sentence. This is how the matrix would look in that occasion.

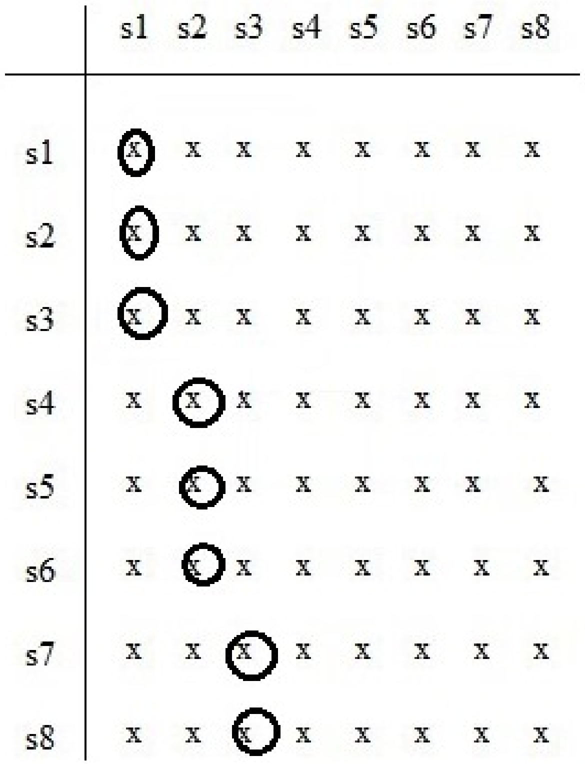

Figure 5.6: A case of expanding paragraphs

While the strictly diagonal case is that when whole paragraphs are either copy pasted or every suspicious sentence is plagiarised (by some kind of obfuscation) from the corre-spondig source sentence. The matrix would then look like:

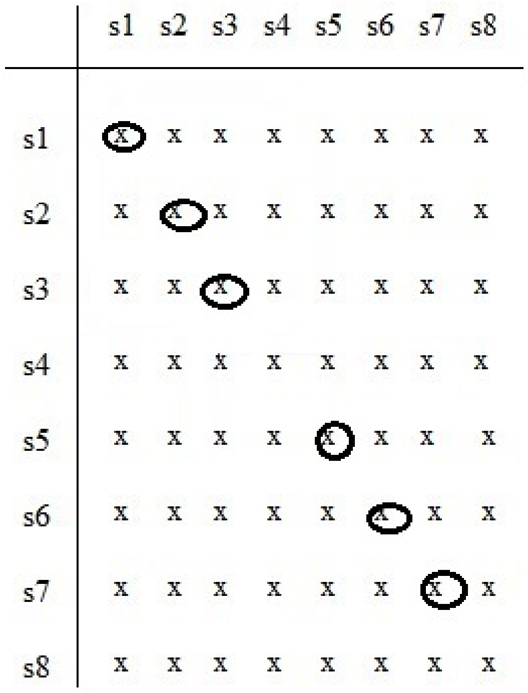

Figure 5.7: Matrix of the strictly diagonal case

As it can be seen, all of these cases present a diagonal structure.We then seek for adjacent plagiarised sentences by moving diagonally across the matrix. In the central findPlagiarism method in the PlagiarismFinder class, we first traverse the matrix picking out those elements that are below the plagiarism threshold. These elements are then put in a sorted map that reflects their position in the matrix, (row,column). This map is then fed to the method that merges the relevant passages. The merging is done by checking the distance between rows and columns for each consecutive element of the map. We allow a distance not larger than four (4) rows or columns between each two passages that should be merged. One more thing we had to consider is that same source sentences cannot have been plagiarised over and over again. Documents with several identical paragraphs are not the goal of a plagiariser anyway. This means that there must be no overlapping between the source references of different detected passages. It is obvious that there must be no overlapping between the suspicious references either.

test with 4,5

Turning back to the example of document 00027 from the group of random obfuscation, our strategy produces this result:

```
Document 00027                  Document 02493

Offset  Length                  Offset  Length

2967    105                     2027    110
3814    1092                    0       1363
5068    274                     2684    611
```

Expressed in matrix form:

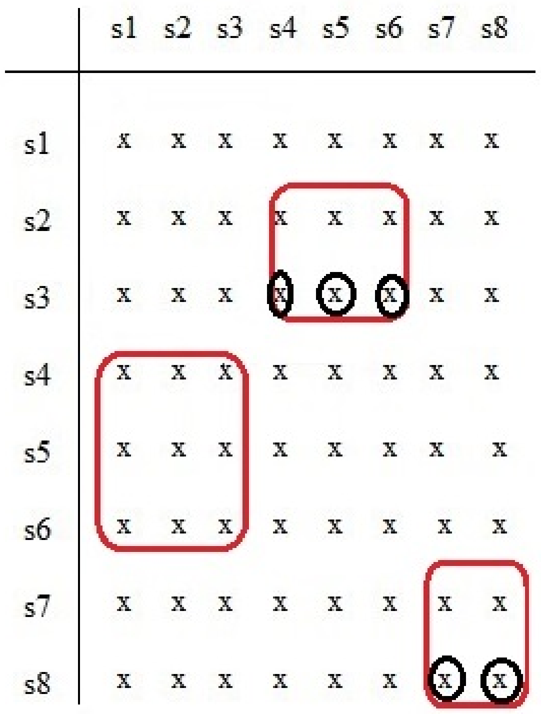

Figure 5.8

As shown, we have missed some sentences of the passage. A more detailed inspection of the matrix reveals that the detection algorithm assigns to these sentence pairs a value that is far above the threshold, so letting the merger include the nearest sentence pairs if their plagiarism score is slightly above the threshold is not always the right solution. There is a limit on how much this strategy can make up for the defeats of the algorithms. This search strategy is completely useful only for merging plagiarism passages that are part of a larger passage, when these are rightly detected by the algorithms.
We still keep the merging of 1500 characters (parameter tuning shows that in the case of semantic distance we should allow merging of paragraphs that lie as far as 2500 characters apart) in addition to the matrix strategy. After these changes, the detailed analysis phase has taken this form:

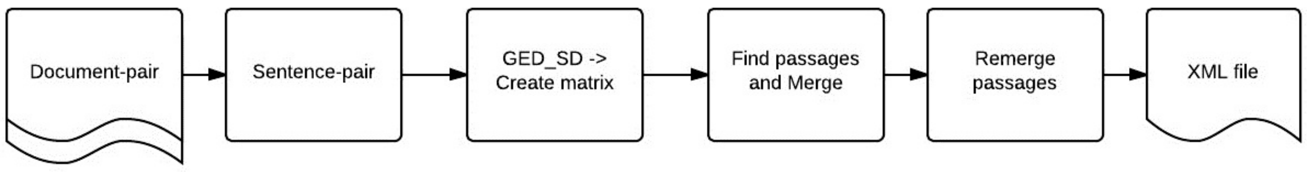

Figure 5.9: The resulting Detailed Analysis phase

5.5.1 Merging Adjacent Passages

(28) proposed a merging strategy whereby the granularity would be reduced by allowing passages adjacent to plagiarised passages get classified as plagiarised by letting their plagiarism threshold be lightly higher, for example, 0.4. We have implemented this idea by using 0.4 as threshold for adjacent passages. It should be mentioned that this strategy is not integrated with the matrix strategy but tested on its own.

Chapter 6

Results and Evaluation

Every scientific work is justified by the results of its evaluation. The nature of this evaluation depends on the project itself and different disciplines impose different evaluation methods. Computer science is mostly concerned with delivering technological products that are usable in pragmatic contexts. Therefore it is natural to think of the evaluation of an Artificial Intelligence project, in terms of the performance of the product that it has created. This performance could be judged by the professionals in a qualitative analysis, or we could measure it by how much it is widespread among the targeted users. This chapter employs the evaluation method proposed by the *"How Evaluation Guides AI Research"* (24) paper. The stages proposed for an evaluation process are first presented, then a short overview of how an evaluation plan for this work is set up, follows.

6.1 How does AI research proceed

The (24) presents a framework for doing AI research.The authors advocate from the start the importance of evaluation not only as a performance measure but also as an answer to a series of questions: why we are doing the research, why the tasks we choose are illustrative, why our views are avant-guard. Moreover they emphasize the necessity of showing how these planned tasks are implemented by the resulting system, how this system works and whether there are possibilities of improvement or the system has reached its best performance. A thorough answering of these questions would illuminate the audience of prospective researchers not only about the workings of the system, but more importantly how research of a particular topic should proceed. Besides this, evaluation is seen as a basis for accumulation of knowledge as documenting not only the answers that the project delivers but also the new questions that are produced on the way, it gives rise to new paths or other research topics. On the other hand, if a work is not evaluated, it would be difficult to replicate its results. The authors base their evaluation framework, on the five stages of a research process: refining the topic to a task; refine the view into a specific method; implement the method; design experiments to test the implementation; run the experiments. This is of course a bit idealized, they argue, but it can still work as a reference for standard AI research. In the following the evaluation of each of these steps is described.

6.1.1 Refine a topic to a task

The first stage presented is to refine the topic at hand to a task and a view. The task itself is what we want the computer to do while the view is an imprecise idea of how we expect to accomplish the task. The questions listed below are supposed to guide this process. These questions and the ones belonging to the other steps, are formulated by the authors. We have merely tried to understand their importance to the research activity.

(a) Is the task significant and why?

(b) Is your research likely to meaningfully contribute to the problem? Is the task tractable?

(c) Is the task representative of a class of tasks?

(d) Have any interesting aspects been abstracted away or simplified, if this problem has been previously defined?

(e) What are the subgoals of the research? What key research tasks will be or have been addressed and solved as part of the project?

(f) How do you know when you have successfully demonstrated a solution to the task? Is the task one in which a solution can be demonstrated?

Asking for the significance of a task is asking if the completion of the work is at all meaningful. If the specific problem addressed has been defined before, this would make the task not worthy of accomplishment. After the task has been formulated one has to make sure that it is tractable and how the solution could be classified as successful or not. It should also be clear what aspects have been left out or made simpler.

In our approach the topic of combining different distance measures to detect plagiarism was refined to the task of integrating into the given system that was based on a GED algorithm an algorithm that accounted for the semantic distance between the texts. The algorithms found were actually designed to compute the semantic similarity and we could ask if turning similarity measures into distance measures does still preserve their original confidence?
The task is representative for the class of tasks that employ a number of different features when considering the detection of plagiarism. The subgoals of the research was to find out if it was computationally plausible to use both algorithms and how they exactly should be combined. Originally the idea was to let the algorithm that decided semantic similarity between words get involved into the GED algorithm itself in the process of building the matrix in the GraphEditDistance.java. But since the system was based in comparing sentences, the approach of deciding semantic distance for the actual sentence as a whole was considered as more favorable.
The other goal was that of increasing the precision by not allowing overlapping results and increasing the recall by merging the non detected passages that lay in between detected ones. This was achieved by building a matrix of sentence pairs that came from the two documents getting compared. The ones getting scores lower than the threshold were then listed in a sorted list. This accounted for non-redundant results.

6.1.2 Designing the Method

The questions of the second phase ask for a researcher to get to know the field and the latest results concerning the task at hand. In the case of producing a system for the PAN challenge, one has for example to review the approaches taken by the participants of the previous year. It would be rare if a given task in AI would not rely on other methods, so it is important to explicitly denote which methods we are relying on. The assumptions made do also need a clarification as the falsity of one of them would be disastrous during the development. One should also count with not getting excellent results, but it is also important to distinguish poor results from results that mirror a total failure. (24) here pose these questions:

(a) How is the method an improvement over the existing technologies?

(b) Does a recognized metric exist for evaluating the performance of your method?

(c) Does it rely on other methods?

(d) What are the underlying assumptions?

(e) What is the scope of the method?

(f) When it cannot provide a good solution, does it do nothing or does it provide a bad solution?

The methods we relied on was that of the reported success of the semantic similarity measure algorithms that were used in other approaches . We cannot say that our method is an improvement over the existing technologies of plagiarism detection. The recognized metric, besides that of PAN, could be a human judgement correlation or an accuracy score for the algorithms. When talking about natural language processing, the main assumption is that of relying on the theories from linguistics.
The method does always produce a solution, though not always satisfying.

cite here

6.1.3 Build a Program

When building a program, Paul R. Cohen (24) point to four issues that need to be discussed:

(a) how demonstrative is the program?

(b) Is it specially tuned for a particular example?

(c) How well does the program implement the method?

(d) Is the program's performance predictable?

We can say that the program was somehow tuned to the dataset of PAN. The specificities that follow from this are the fact that it needs to read a file which tells it which pairs of documents should be compared. The performance of the program is not predictable.

6.1.4 Design Experiments

This phase could be totally omitted since we are going to evaluate by the PAN measures, as the experiments are already set up. In general, however, Paul R. Cohen (24) bring forth five issues that the design process needs to address:

(a) How many examples can be demonstrated?

(b) Should the program's performance be compared to a standard such as another program, or experts and novices, or its own tuned performance?

(c) What are the criteria for good performance? Who defines the criteria?

(d) Does the program purport to be general?

(e) Is a series of related programs being evaluated?

Numerous examples could be shown as the different combinations of the parameters and the features involved leave room for this. The program's performance could be compared to that of the other participants in PAN. The last year's results [1] do, however, show that the program's performance is lower than the majority's.

6.1.5 Analyze the Experiment's Results

This stage is what most people regard as evaluation, that is why the previous stages and the questions they come with are often left out or not even considered as part of the evaluation. In addition to presenting the performance by the measures of evaluation, like for example using the *plagdet* value, one could oversee other important factors like the resources used to achieve those results. Paul R. Cohen (24) list four questions which the analysis phase must answer:

(a) How did the program performance compare to its selected standard?

(b) Is the program's performance different from predictions of how the method should perform?

(c) How efficient is the program in terms of space and knowledge requirements?

(d) Is it easy for the intended users to understand?

The selected standard was that of the *plagdet* score. The version of combined GED and Jiang Conrath appears to give the best performance on the whole dataset and this regards both the *plagdet* score and the score from F-measure. This algorithm combination seems to score best on all the types of datasets. We predicted that a combination of syntax -based measures like GED and semantic based measures, like the *aggregator* and the semantic distance algorithms it incorporates, would result in an improvement of a detector. This was justified by the results achieved. The other question was if a different merging strategy from the original would elevate the result. This did as well result in an improvement of the score. Our other prediction, that by assigning weights to the substitution of dependency relations we would improve the performance of GED, did not give any positive

[1] http://pan.webis.de/

result. This could be due to the fact that our weights were not correctly defined.
The only knowledge required by the GED algorithm is that of delete, insert and substitute costs for the POS tags and for the *deprel* edges. In contrast, the Jiang & Conrath/Resnik algorithms require a lexical database, in our case WordNet. Moreover, the *aggregator* requires a corpus in order to derive the idf values.
We suppose that this approach, the method and the way it is carried out, is simple and easy for the intended users.

6.2 Experiments: Tuning parameters

As discussed in 2, we evaluate the system by using the *plagdet* score formulated by PAN. Since the system was now enhanced by both an additional detection algorithm and a new detection strategy, some tuning of the parameters was performed. The parameters to be tuned were: 1) Plagiarism Threshold, 2) Merging distance inside the matrix 3) the original Merging distance For this task, we used the method of K-Fold validation(3-Fold) by partitioning the dataset into three almost equal pieces. We chose K=3, due to time constraints. For each parameter, we ran the program on the 2 training partitions and then tested in on the remaining data. In order to generalize the tests,the training corpus was built by selecting an equal amount of data from the five categories of the PAN13 corpus . The experiments resulted in these decisions:

- Plagiarism Threshold: The value 0.3 does still give the highest *plagdet* score.
- Merging distance inside the matrix: The original intuition was to set this value to a degree lower than 10 and higher than 4, but the experiments showed that a value as large as 25 or 30 gave a better score.
- Original merging distance: This was tested only on the version where the semantic distance and the matrix strategy are employed since it was already decided as the best parameter value for the GED algorithm alone. The new merging distance is that of 2500 characters.

Another thing that had to be decided was if the original approach of merging detected passages that lay in a distance not larger than 1500 characters, should still be kept. The running experiments showed that a combination of the matrix strategy and this original merging resulted in the best performance scores.

6.3 Results

The following tables present the scores achieved by several versions of the system at hand. The short descriptions of the Algorithm column mean respectively:

- GED Original: The original system of (28) without the ”Candidate Retrieval” phase.
- GED Adjacent: The original system with the adjacent merging strategy and the original merging of 1500 characters.

- GED Matrix: The matrix merging strategy applied on the original system. The merging of 1500 characters is still kept and performed after the matrix-merging.
- GED-SD-RESNIK: The combination of the graph edit distance and the *aggregator* that uses the ResnikComparator, the matrix merging strategy and the original merging of 2500 characters.
- GED-SD-Jiang Conrath: The combination of the graph edit distance and the *aggregator* that uses the JiangConrathComparator, the matrix merging strategy and the additional merging of 2500 characters.

The parameters described during testing are those decided by training as described in chapter 4:

Plagiarism Threshold	0.3
Merging Distance inside the Matrix	30
Additional merging distance	2500

The following tables show the performance results for the respective datasets. The dataset of No-Plagiarism cases was left out, as the system evaluates to only zero values when tested both with the GED algorithm alone and when combined with the SemanticDistance. Arguably, this happens because the PAN evaluation expects the respective XML documents to point explicitly to the non-plagiarised passages/documents, while the original implementation did not account for this. The F-measure was taken in the evaluation in order to have a combined score of the precision and recall alone when granularity is not of importance.

PAN 13					
Algorithm	Plagdet	Recall	Precision	Granularity	F-measure
GED Original	0.153	0.086	0.870	1.026	0.156
GED Adjacent	0.106	0.057	0.811	1.021	0.107
GED Matrix	0.182	0.105	0.69	1.004	0.182
GED-SD-Resnik	0.239	0.141	0.808	1.006	0.240
GED-SD-Jiang & Conrath	0.243	0.147	0.717	1.003	0.244

The first thing to note is that the original system (GED Original) reports a lower performance when tested on PAN13 dataset. The *plagdet* score from PAN11 was as high as 0.24. We do therefore use the result from PAN13 as a baseline in our evaluation. We can draw these conclusions from the table above:

(a) The adjacent merging does reduce the granularity but results in a drastic fall for the recall. This sounds counterintuitive and raises the question whether this merging was correctly implemented.

(b) The matrix strategy results in a higher recall but it lowers the precision. It does however lower the granularity as well, so it can at least not be judged as unsuccessful. This strategy raises the baseline score by 19%.

(c) The semantic distance algorithms do raise the *plagdet* score. It results in an improvement of $33\&$ over the score from GED Matrix and an improvement of 58% of the GED baseline. The difference between Resnik and Jiang & Conrath is quite low (0.004) so we cannot say that the one algorithm performs considerably better than the other when integrated in this system. Therefore the GED-SD-Resnik is not presented in the following tables.

No obfuscation				
Algorithm	Plagdet	Recall	Precision	Granularity
GED Original	0.199	0.149	0.306	1.009
GED Adjacent	0.140	0.076	0.804	1.0
GED Matrix	0.172	0.156	0.214	1.070
GED-SD-Jiang & Conrath	0.255	0.157	0.674	1.0

No obfuscation is the case of verbatim plagiarism. This is also the case where the GED baseline performs best when compared to the other datasets. The Jiang & Conrath semantic distance still gives the best *plagdet* score.

Random obfuscation				
Algorithm	Plagdet	Recall	Precision	Granularity
GED Original	0.089	0.057	0.208	1.007
GED Adjacent	0.082	0.052	0.198	1.0
GED Matrix	0.090	0.072	0.236	1.337
GED-SD-Jiang & Conrath	0.215	0.124	0.816	1.0

Translation obfuscation				
Algorithm	Plagdet	Recall	Precision	Granularity
GED Original	0.087	0.569	0.184	1.0
GED Adjacent	0.069	0.043	0.169	1.0
GED Matrix	0.097	0.075	0.152	1.047
GED-SD-Jiang & Conrath	0.152	0.084	0.777	1.0

This is the case where the semantic distance algorithm gives the lowest performance when compared to other datasets. As it can be seen, the recall is very low while the precision still retains a high value.

Summary obfuscation				
Algorithm	Plagdet	Recall	Precision	Granularity
GED Original	0.062	0.066	0.067	1.102
GED Adjacent	0.061	0.065	0.066	1.10
GED Matrix	0.055	0.067	0.053	1.111
GED-SD-Jiang Conrath	0.265	0.267	0.278	1.038

This is the dataset where GED Original has its lowest performance while the combination with the semantic distance has the highest performance when compared with other datasets. This indicates that the semantic distance is better suited to detecting paraphrasing.

Another thing to be mentioned is that when the FindPlagiarism class that employs the matrix-merging strategy is used in the application, the runtime is considerably shorter than that of the other, original version.

Chapter 7

Conclusions

The main abstract goal of this thesis was to explore ways of detecting plagiarised passages by finding corresponding original passages from which they have drawn their content, form or both. The more specific focus was set on improving a given plagiarism detection system based on the graph edit distance strategy. This practical goal led us to the question of whether a graph edit distance algorithm that mainly captures the syntactic and lexical similarity between sentences could be combined with a semantic distance algorithm that values the semantic relation between sentences. The other issue was that of merging detected passages that lay near to each other.
In the course of this project, we have tried to answer the questions posed in the introductory chapter,section 1.2.

(a) **Goal 1**: Find existing solutions and methods for plagiarism detection in the NLP literature
In **??** we presented the background material that is relevant for the area of automated plagiarism detection. We presented some basic theory about NLP and did further show some plagiarism detection approaches and algorithms.

(b) Goal 2: Enhance the performance of the system described in (28)

In chapter 5 we described the implementation done in order to achieve this goal. This goal was partially achieved. A concern is still that of why the original system did perform worse in the face of the data from PAN13.

(c) Research question 1: Does the combination of a graph edit distance measure with a semantic distance measure result in an improvement of plagiarism detection when faced with a large amount of data?

This question can be answered by the results obtained in the datasets. The scores say that this combination is possible and results in an improvement of the detector.

(d) Research question 2: How to detect whole plagiarised paragraphs when not all the passages contained in are detected as plagiarised or even in the case when all the sentences are discovered as plagiarised?

Since our detailed analysis was based on comparing sentences with each other, we found out that building a sentence-sentence matrix was a plausible strategy. It resulted in an improvement of the precision and a reduction of the granularity.

7.1 Further Work

The performance of the system presented in this project is measured on a dataset of artificially constructed plagiarism cases. Since this dataset is delivered by the PAN competition we could compare our performance with that of the other participants. This comparison shows that the performance of our system is far from satisfying. Therefore there is much that still can be improved. One thing could be a better tuning of the dependency relation weights. These weights were defined by using the definition of the particular relations. A better linguistic knowledge could result in different weights.

The results showed that the integration of semantic distance did enhance the performance of the system, even though slightly. This improvement was more evident in the case of paraphrasing. A further approach would thus be to let the GED algorithm decide similarity between short sentences and use the semantic distance algorithm on the long sentences. But this requires a dataset with ”real” sentences , while our system creates sentences arbitrarily. This leads us to the idea that this system could be better tested on a dataset that is not based on artificial plagiarism.

Going back to the example where the original system assigned a low distance value to the two sentences :

```
John kissed Mary
John killed Mary
```

we should mention that the semantic distance algorithm assigned an equally low distance value to this example. This tells us that the problem presented by the original system of detecting this kind of relationships between sentences, is yet not solved. The semantic distance algorithms use WordNet where the relationships between certain kinds of POS like verbs, adjectives and adverbs are rarely defined.

The other concern would be that of using a combination of semantic distance algorithms. In our case we use only one algorithm at a time to derive the score. A combination of different algorithms could result in a more precise score. There is also the question of whether one should take a simple arithmetic mean between the algorithms or try to decide their respective weights in the combination. It should, however, be mentioned that the runtime of these semantic distance/similarity algorithms is quite high.
The last concern regarding the system as a whole is that of recall. For purposes of testing, we tried to raise the plagiarism threshold to very large values and this did not result in any significant change of the recall values. This indicates some malfunctioning of the system. Due to time constraints, we were not able to explore this still remaining problem.

Bibliography

[1] (2000). *Kant and the Platypus, Essays on Language and Cognition.*

[2] (2011). *SPEECH and LANGUAGE PROCESSING An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Pearson.

[3] Ahmed Hamza Osman, Naomie Salim, A. A. (2012). Survey of text plagiarism detection.

[4] Alexander Budanitsky, G. H. (2001). Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. *Workshop on WordNet and Other Lexical Resources.*

[5] Allen, J. F. (2003). Natural language processing.

[6] A.M.Turing (1950). Computing machinery and intelligence. *Mind , 59, 433-460 http://www.loebner.net/Prizef/TuringArticle.html.*

[7] Benno Stein, Sven Meyer zu Eißen, M. P. (2007). Strategies for retrieving plagiarized documents. *30th International ACM Conference on Research and Development in Information Retrieval (SIGIR 07).*

[8] B.Martin (1994). Plagiarism:a misplaced emphasis. *Journal of Information Ethics, Vol.3(2), 36-47.*

[9] Chowdhury, G. G. (2003). Natural language processing. *University of Strathclyde, Glasgow G1 1XH, UK.*

[10] Clough, P. (2003). Old and new challenges in automatic plagiarism detection. *University of Sheffield.*

[11] Davis, R., Shrobe, H., and Szolovits, P. (1993). What is a knowledge representation? *AI Magazine, 14(1):17-33.*

[Dere Proudian] Dere Proudian, C. P. Parsing head-driven phrase structure grammar.

[13] Eneko Agirre, Daniel Cer, M. D. A. G.-A. (2012). A pilot on semantic textual similarity. *First Joint Conference on Lexical and Computational Linguistics.*

[14] Jan Hajic, Eva Hjicova, A. R. (2009). Formal representation of language structures. *Charles University, Prague, Czech Republic*.

[15] Jay J. Jiang, D. W. C. (1997). Semantic similarity based on corpus statistics and lexical tasonomy. *Proceedings of International Conference Research on Computational Linguistics (ROCLING X) Taiwan*.

[16] Jones, M. (2009). Back-translation: The latest form of plagiarism. *University of Wollongong*.

[17] Kaspar Riesen, H. B. (2009). Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*.

[18] Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: How to tell a pine code from an ice cream cone. *Proceedings of the 5th annual international conference on Systems documentation.ACM*.

[19] Marie-Catherine der Marneffe, C. D. M. (2008). Stanford typed dependencies manual.

[20] Martin Potthast, Tim Gollub, M. H. J. G.-J. K. M. M. A. O. M. T. A. B.-C. P. G. P. R. B. S. (2012). Overview of the 4th international competition on plagiarism detection. *CLEF Evaluation Labs and Workshop*.

[21] Miranda Chong, Lucia Specia, R. M. (2010). Using natural language processing for automatic plagiarism detection. *4th International Plagiarism Conference;Northumbria University, Newcastle upon Tyne, UK.*

[Nathaniel Gustafson] Nathaniel Gustafson, Maria Soledad Pera, Y.-K. N. Nowhere to hide: Finding plagiarized documents based on sentence similarity.

[23] Nivre, J. (2003). Dependency grammar and dependency parsing.

[24] Paul R. Cohen, A. E. H. (1988). How evaluation guides ai research. *AI Magazine Volume 9 Number 4 AAAI*.

[25] Potthast, M. (2011). Technologies for reusing text from the web. *Faculty of the Media, Bauhaus-Universität, Weimar, Germany*.

[26] Rada Mihalcea, Courtney Corley, C. S. (2006). Corpus-based and knowledge-bases measures of text semantic similarity. *AAAI'06*.

[27] Resnik, P. (1999). Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language", volume 11, pages 95-130.

[28] Røkenes, H. D. (2012). Graph-based natural language processing:graph edit distance applied to the task of detecting plagiarism.

[29] Roy Rada, Hafedh Mili, E. B.-M. B. (1989). Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics*.

[30] Seo, M. (2009). Plagiarism and poetic identity in martial. *http://muse.jhu.edu/journals/ajp/summary/v130/130.4.seo.html*.

[31] Sparck-Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*.

[Tao Jiang] Tao Jiang, Ming Li, B. R.-K. W. R. Formal grammars and languages.

[Thanh Ngoc Dao] Thanh Ngoc Dao, T. S. Measuring similarity between sentences.

[34] Thomas K Landauer, Peter W. Foltz, D. L. (1998). An introduction to latent semantic analysis. *Discourse Processes*.

[35] Tim Gollub, Benno Stein, S. B. D. H. (2012). Tira: Configuring, executing and dissaminating information retrieval experiments. *9th International Workshop on Text-based Information Retrieval (TIR 12) at DEXA*.

Appendix

Write your appendix here...