NTNU – Trondheim
Norwegian University of
Science and Technology

Graph-based Natural Language Processing

Graph edit distance applied to the task of detecting plagiarism

Håkon Drolsum Røkenes

Master of Science in Informatics
Submission date: December 2012
Supervisor: Bjørn Gamback, IDI
Co-supervisor: erwin marsi, IDI

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

Abstract

The focus of this thesis is the exploration of graph-based similarity, in the context of natural language processing. The work is motivated by a need for richer representations of text. A graph edit distance algorithm was implemented, that calculates the difference between graphs. Sentences were represented by means of dependency graphs, which consist of words connected by dependencies. A dependency graph captures the syntactic structure of a sentence.

The graph-based similarity approach was applied to the problem of detecting plagiarism, and was compared against state of the art systems. The key advantages of graph-based textual representations are mainly word order indifference and the ability to capture similarity between words, based on the sentence structure.

The approach was compared against contributions made to the PAN plagiarism detection challenge at the CLEF 2011 conference, and would have achieved a 5th place out of 10 contestants. The evaluation results suggest that the approach can be applicable to the task of detecting plagiarism, but require some fine tuning on input parameters.

The evaluation results demonstrated that dependency graphs are best represented by directed edges. The graph edit distance algorithm scored best with a combination of node and edge label matching. Different edit weights were applied, which increased performance.

Keywords: Graph Edit Distance, Natural Language Processing, Dependency Graphs, Plagiarism Detection

Sammendrag

Fokuset i denne oppgaven er utforskingen av graf-basert liket, i forbindelse med naturlig språk prosessering. Arbeidet er motivert av et behov for sterkere representasjoner av tekst. En graf redigeringsdistanse-algoritme ble implementert, some sammenlikner forskjellen mellom to grafer. Setninger ble representert som avhengighetsgrafer, som består av ord koblet sammen av avhengigheter. En avhengighetsgraf fanger den syntaktiske strukturen til en setning.

Den graf-baserte tilnærmingen ble anvendt til problemet å avdekke plagiat, og var sammenliknet mot toppmoderne systemer. Hovedfordelen med graf-basert representasjon av tekst er muligheten til å fange liket mellom ord basert på setningsstruktur. Avhengighetsgrafer er i tillegg likegyldige til rekkefølgen av ord.

Tilnærmingen ble sammenliknet med bidrag til PAN plagiarism detection challenge ved konferansen CLEF 2011, og ville ha oppnådd 5. plass av 10 deltakere. Resultatene tyder på at tilnærmingen kan brukes til å avdekke plagiat, men krever finjustering av inputparametre.

Resultatene demonstrerte at avhengighetsgrafer er best representert med rettede kanter. Redigeringsdistanse-algoritmen skåret best med en kombinasjon av node- og kant-matching. Forskjellige redigering-vekter ble brukt, som økte ytelsen.

Preface

This thesis describes the work done during the final year of my Master of Science MSc degree in Informatics. The work was conducted from January to December 2012, at the Department of Computer and Information Science at Norwegian University of Science and Technology.

I wish to thank my supervisors Björn Gambäck and Erwin Marsi for their guidance, support and encouragement throughout the course of this thesis. I also take the opportunity to thank Lars Bungum for technical support on the equipment used for this thesis.

In addition, I would like to thank the organisers of the PAN evaluation lab, for providing the environment I needed to evaluate my results.

Håkon Drolsum Røkenes
Trondheim, December 10, 2012

Contents

List of Figures

List of Tables

Terminology & abbreviations

CLEF Conference and Labs of the Evaluation Forum

Edge Edges connect nodes in a graph structure. An edge can be directed or undirected, and may be labelled or weighted.

Granularity A measurement of how something is partitioned into smaller pieces.

Graph A data structure which consists of nodes connected by edges.

Hypernym A word which represents a generalisation of another word. Example *blue* is a *colour*, where *colour* is the hypernym to *blue*.

Natural language processing A field within computer science which covers processing of human language by computers.

Node A graph consists of a set of nodes. Each node usually has a label and is connected by edges. Vertices is another name for nodes.

PAN Plagiarism analysis, Authorship identification and Near-duplicate detection

Plagiarism passage Plagiarism passages refers to pairs of sentences, paragraphs or whole texts which are considered plagiarised.

Precision In an information retrieval system, precision is the fraction of retrieved items which are relevant.

Recall In an information retrieval system, recall is the fraction of items retrieved from all relevant items.

Semantics The study of meaning.

Synonym Words with nearly identical meaning are considered to be synonyms.

Syntax The study of how sentences are constructed.

TF-IDF Term Frequency - Inverse Document Frequency

Chapter 1

Introduction

The main objective of this thesis is to explore the usage of graph-based representations for the field of natural language processing. The following sections present a motivation for said approach and define a problem description. In addition, a problem solution is proposed, which guides the practical work done in this thesis. Finally, an overview of the rest of the thesis is presented.

1.1 Context and motivation

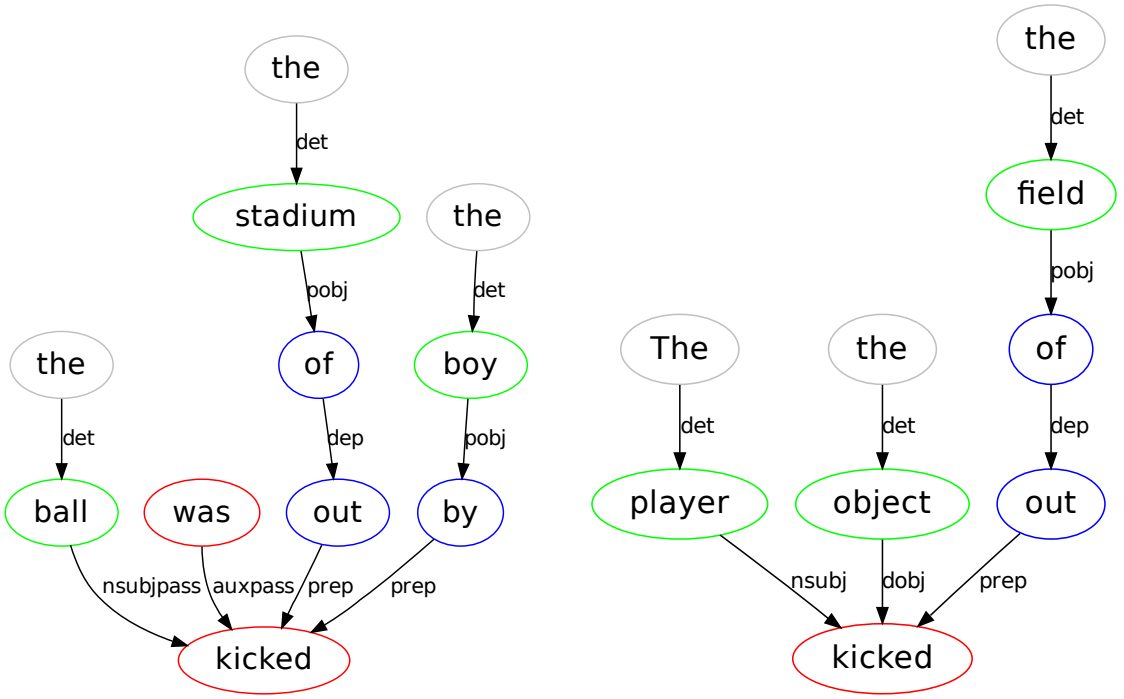
The problem of finding similarities between two texts is a general problem which has many applications within the field of natural language processing. In order to measure the similarity between two texts, each text needs a representation. One way is plain text, a list of words which form a sentence. Plain text is often used for simplicity, but lacks explicit information about syntactical structure.

Some aspects of language are better represented using structured representations such as dependency graphs, which consist of words connected by dependencies.

Dependency graphs capture the syntactical structure of a sentence, and is limited to a sentence scope. One of the main advantages of a graph based representation, is that dependency graphs are for the most part indifferent to word order. This allows the representation to capture similarity between sentences where the word order is shuffled around.

A richer representation provides a better basis for determining similarity in difficult cases. Consider the two sentences represented as dependency graphs in Figure 1.1. A human should be able to determine that the sentences have similar meanings. However, automatically detecting such cases can be troublesome, due to word replacement. Given a plain text representation, the only common words are *kicked*, *the*, *out* and *of*. The word *kicked* is the only word which gives a clear indication of the meaning of each sentence. By looking at the dependency graphs, it becomes clear that there are some structural similarities between the sentences. The edge labels, colours and construction of dependency graphs will be described in detail in Section 2.2.

Due to increasing availability of documents available on the web, performing plagiarism is getting easier and easier. Large amounts of potentially plagiarised text is submitted to educational institutions every year. As a result, there is an increasing need for automatic plagiarism detection.



The ball was kicked out of the stadium by the boy

The player kicked the object out of the field

Figure 1.1: Example of dependency graphs

1.2 Problem description

The concrete task in this thesis is to implement a graph edit distance algorithm, which calculates similarity between two graphs. The algorithm is based on calculating the number of edit operations needed to transform one graph into another (Riesen and Bunke, 2009). Each edit operation has an edit cost, which determines how costly the given operation is.

Automatic plagiarism detection is a research field which relies primarily on text similarity. This makes it an interesting test bed for the topic of this thesis. The problem of applying graph-based text similarity to plagiarism detection is defined in research question 1.

Research question 1. *Is graph-based similarity, in particular graph edit distance, applicable to plagiarism detection and computationally feasible?*

State-of-the-art plagiarism detection systems mostly rely on simpler representations of text, such as n-gram matching and the vector space model (Suchomel et al., 2012; Kong et al., 2012; Grman and Ravas, 2011; Oberreuter et al., 2011; Grozea and Popescu, 2011). As a result, the approach can be considered relatively unique. Due to the uniqueness of the approach, some implementation details remain undefined. Research question 2 addresses the problem of defining the details of the algorithm.

Research question 2. *What is the best way to calculate graph edit distance between sentences, in particular with respect to edit costs and graph representation, in the context of plagiarism detection?*

In order to evaluate the performance of the graph edit distance algorithm, its performance should be compared to existing state-of-the-art systems. Research question 3 presents the problem of comparing the algorithm to existing approaches.

Research question 3. *How does graph-based similarity compare to other approaches to plagiarism detection, such as index-based retrieval and n-gram matching?*

1.3 Problem solution

The following goals will serve as solutions to the corresponding research questions formulated in Section 1.2.

Research goal 1. *A prototype plagiarism detection system based on graph edit distance will be implemented. The system should be able to handle large text corpora, and detect plagiarism on a sentence level.*

Research goal 2. *Various edit cost functions and graph representations will be implemented. The different approaches will be compared to each other in the same manner as in Research goal 1.*

Research goal 3. *The output of the prototype system will be compared to existing state-of-the-art plagiarism detection systems. This comparison will serve as an empirical evaluation of the system.*

In addition, the system will feature other phases than just the graph edit distance algorithm. A standalone evaluation will be made on just the graph edit distance algorithm.

1.4 Thesis structure

The thesis is structured as follows:

Chapter 2 Presents the theory needed to implement the solution described in Chapter 4.

Chapter 3 Covers relevant research done in the field of plagiarism detection and graph edit distance computation.

Chapter 4 Describes the prototype graph-based plagiarism detection system implemented for this thesis.

Chapter 5 Lists the results generated from the prototype system, compared to state-of-the-art plagiarism detection systems.

Chapter 6 Discusses the results presented in the previous chapter.

Chapter 7 Summarises the work and discusses possible future work.

Chapter 2

Theory and background

This chapter covers relevant research done surrounding the main topics of the thesis. The main goal of this chapter is to introduce relevant terminology used in the field of plagiarism detection and graph edit distance computation.

2.1 Plagiarism detection

In order to detect plagiarism, various information retrieval techniques are applied. This section describes techniques which are useful during text matching.

2.1.1 N-gram matching

An n-gram is a sub-sequence of n items from a text. Each item is typically represented by the word in its original form, or any other representation available. n refers to the size of each sub-sequence. An n-gram of size 1 is called a *unigram*, size 2 *bigram* and size 3 *trigram*. *n-grams* of larger sizes are just referred to as *fourgrams*, *fivegrams*, etc. n-gram items may represent other items, such as the characters in a word, which has other applications. The focus in this thesis is sentence level matching, so only word based n-grams will be considered.

n-gram matching is quite common approach in the field of text categorisation. The core concept behind n-gram matching is to count the number of matching n-grams in two texts (Papineni et al., 2002). A common approach is to combine matching of different n-gram sizes. Different applications utilise different sizes of n-grams, and the best size depends on the specific problem and data set.

2.1.2 Term weighting

During information retrieval, the importance of each term can be represented by a weight. Each term is weighted according to how well it contribute to solving a particular task. The following subsections presents some common term weighting schemes, which are relevant for plagiarism detection.

Stop-word removal

Stop-words are commonly used terms, such as *a*, *the*, *but*, etc. Such terms are content-independent, and carry relatively little semantic information (Stamatatos, 2011). Stop-word removal can be applied to better capture the semantics of a text, as stop-words are considered less important for determining the semantics of a text (Koppel et al., 2006).

Term Frequency - Inverse Document Frequency weighting

A more advanced term weighting technique is *Term Frequency - Inverse Document Frequency weighting* (TF-IDF). The technique looks at the number of occurrences a term has in a given document, and the amount of occurrences in all the documents together. This way less frequent, more specific terms are of greater value than matches on more common terms (Jones, 1972). There exist some variants of the formula, but the most common is the following (t is the term, d is the given document and D is the document collection):

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D) \tag{2.1}$$

where $tf(t, d)$ is the term frequency of t in document d. Usually the term frequency is calculated using the number of term occurrences divided by the number of terms in document. $idf(t, D)$ is the inverse document frequency, usually calculated by taking the logarithmic of the number of documents divided to the number of documents containing term t.

The strength of TF-IDF weighting is the ability to weight all terms. Unique terms are weighted highest, while more common terms, yet not classified as stop-words, will receive a lower weight. TF-IDF weighting does not require a dictionary lookup, thus making it language independent. However, finding the IDF values for each term does require some calculations.

2.1.3 Term generalization

The following sub-sections describe techniques which allow us to express terms (words) in a more general form.

Lemmatisation

Lemmatisation describes the process of representing different forms of words in a more general representation, called lemma. For instance, the word *be* has seven different forms: *be, am, are, is, was, were, been*. By representing the different forms with the word *be*, word matching becomes more robust.

However, representing words as lemmas may cause erroneous behaviour, in cases where two different words have the same lemma. Consider the words *recording* and *records*, which may refer to the verb (to) *record*, or a (music) *record*. Both words have the same lemma, and would falsely be regarded as equal if only lemmas are considered.

Part-of-speech tagging

Part-of-speech (POS) tagging is the process of identifying words as nouns, adjectives, etc. By identifying the POS-tag of a word, it is easier to deal with word-sense disambiguation. For instance, the word *left* may mean the opposite of *right*, or that a person just *left* the building. By identifying the POS-tag of the two words, the difference is clear, because the first usage is a noun whereas the second is a verb. There exists automatic tools which determine the POS-tag of words in a sentence, called POS-taggers. Since a word can have multiple POS-tags, a POS-tagger requires a full sentence as input, in order to perform disambiguation.

The Penn Treebank is a corpus consisting of over 4.5 million words of American English, which are annotated with part-of-speech tags. Appendix A.1 features a list of part-of-speech tags, taken from the Penn Treebank (Marcus et al., 1993). The corpus is used by popular POS-taggers such as the Stanford Log-linear Part-Of-Speech Tagger (Toutanova et al., 2003). Other corpora are available for languages other than English.

2.2 Dependency graphs

A dependency graph is a structured representation of a sentence. In order to create dependency graphs, a sentence is processed by a dependency parser, which is based on the theoretical foundations of dependency grammar. Dependency grammar can be described as a family of theories that share some basic assumptions about grammatical structure, mainly the assumption that syntactic structure is represented by words connected with relations called

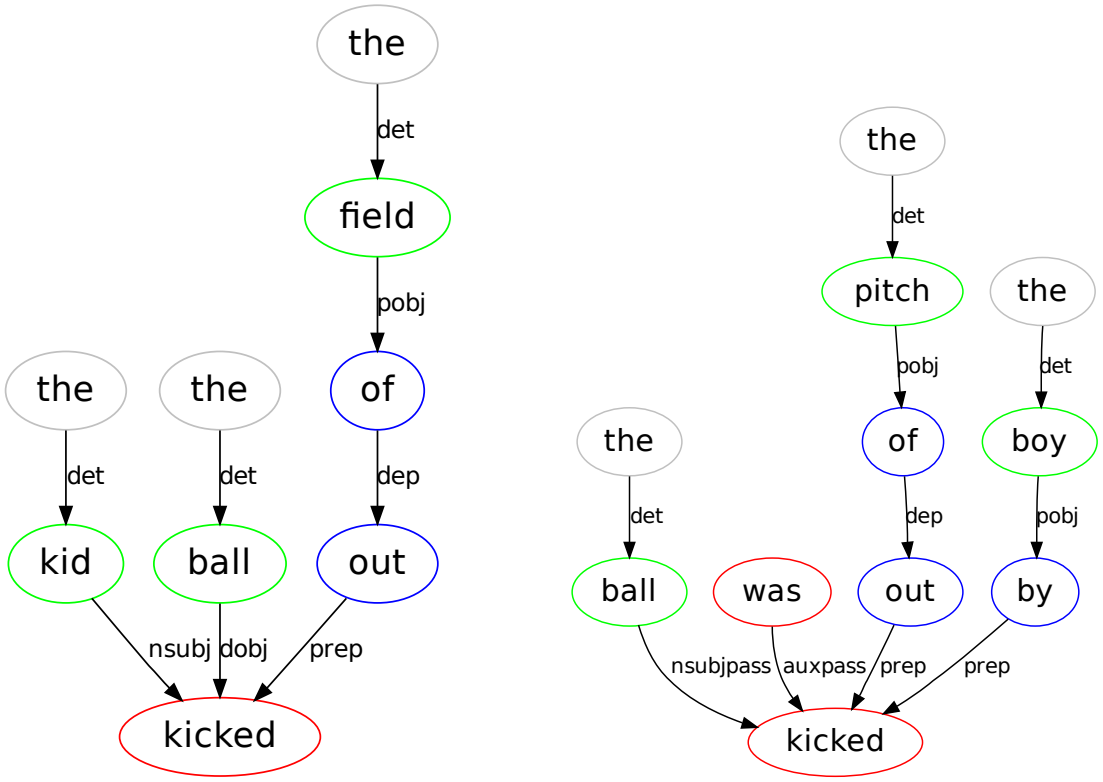
Syntactical structure can be very complex and most dependency graph versions are simplifications of theoretical dependency structures. Complex dependency graphs require complex algorithms. Simple graphs express less, but are easier to work with, especially if the goal is automatic dependency parsing.

Consider the two sentences in Figure 1.1, which is repeated for convenience. The dependency graphs are created using MaltParser, which is one of several dependency parser alternatives. The node colours represent the POS-tag of the node's word. Table 2.1 lists the different colours and their respective POS-tag.

Colour	**POS-tag**
Green	Noun
Red	Verb
Blue	Preposition
Orange	Adverb
Grey	Other POS-tags

Table 2.1: POS-tag colours used in figures

The edges represent the *dependencies* between nodes. The edge label represents the type of dependency relation. These attributes will be described in detail in the following sections.



The kid kicked the ball out of the field

The ball was kicked out of the pitch by the boy

Example of dependency graphs

2.2.1 Dependencies

A dependency between two terms, using the terminology from Nivre (2005), consists of a *head* and *dependent*.[1] In other words, dependencies are directional, making dependency graphs directed graphs.

Often there is a need to distinguish between different dependency relations, annotated by the dependency relations in Figure 1.1. The edge labels, called deprel for short, describe the relationship between the *head* and *dependent*. Consider the relationship between *ball* and *kicked* in Figure 1.1. The deprel *dobj* denotes that *ball* is a direct object of a verb phrase, in this case the word *kicked*. In general, this dependency involves a relation between an action (*kicked*) and an entity directly affected by this action (*ball*). In other words, the deprel states that the *ball* was kicked. De Marneffe and Manning (2008) presents a full description of the different dependency relations used in this project.

[1]The literature is not consistent when it comes to drawing dependencies (arrows from *head* to *dependent* or vice versa). The figures in this thesis will have dependency graphs with arrows pointing from *dependent* to *head*.

2.3 Conference and Labs of the Evaluation Forum

Conference and Labs of the Evaluation Forum, formerly known as Cross-Language Evaluation Forum is a European conference held in various countries each year since 2010 (Braschler et al., 2010; Petras et al., 2011; Forner et al., 2012). The conference is abbreviated CLEF for short. CLEF promotes research within the field of information retrieval, with focus on multilingual and multi-modal information. For each CLEF conference, workshops are held, which focus on solving a specific problem. Amongst these is the PAN workshop,[2] which is divided into three sub-tasks:

- plagiarism detection
- authorship detection
- social software misuse

2.3.1 Plagiarism detection challenge

Plagiarism detection can be defined as detecting text passages between one or several documents and a document corpus, which are above a chosen degree (threshold) of similarity. This makes it an interesting application for this project. The PAN workshop allows researchers within the same field to compare results. The Plagiarism Detection challenge consists of the following sub-tasks:

- Intrinsic detection

  The task is to detect plagiarism in a document by detecting writing style changes

- External detection

  The task is to detect plagiarised passages between a source and suspicious document collection.

Only the external detection task is of interest to this thesis, since it is based on text similarity.

Data sets

Each PAN plagiarism challenge comes with a new data set, which is reviewed and improved for each year. The different data sets will be referred to as PAN{year}. Each competition uses data sets from the preceding year as training corpus, for parameter tuning.

The reason for working with PAN11 instead of PAN12 is due to changes in the PAN12 workshop. Unfortunately, the PAN12 data set is not freely available, as it contains real plagiarism cases. To protect the identity of the authors, the data must be accessed through a search API. In order to use the search API, one has to be a contestant in the competition. The deadline for PAN12 ran out before the completion of the system, thus making a submission impossible (Potthast et al., 2012). As a result, the PAN10 and PAN11[3] are used the training and test sets used to obtain the results

[2]PAN website: http://pan.webis.de/

[3]The PAN10 and PAN11 data sets are available at http://www.webis.de/research/events/pan-11

in Chapter 5. The data sets are summarised in Table 2.2.

Data set	Suspicious documents	Source documents	Translated	Artificial	Manual
PAN10	15925, 1.7GB	11148, 3.5GB	14%	40%	6%
PAN11	11093, 1.6GB	11093, 2.5GB	11%	63%	8%

Table 2.2: Summary of PAN10 and PAN11 data sets

Each corpus is separated into source and suspicious documents. Each suspicious document is analysed against every source documents, with the goal of detecting plagiarism. All suspicious documents come with a corresponding annotation file. The annotation files offer pointers to source file, starting offset and length.

Translated, artificial and manual refer to different plagiarism approaches. Translated plagiarism refers to plagiarism cases between two documents of different languages. Artificial plagiarism is created by artificially obfuscating text. Manual plagiarism refers to manually obfuscated text, generated by crowd sourcing.

2.3.2 Evaluation

Each contribution is evaluated using a measure called *plagdet*, which is the evaluation measure used for the PAN workshop. The following formulas are taken from Potthast et al. (2010), which describe the evaluation process in detail.

The *plagdet* measure is a combination of *precision*, *recall* and *granularity*. *granularity* is a measure which recognises whether or not adjacent passages are detected. In other words, if a system detects many adjacent passages, they should be merged to one. Formula (2.2) is used to calculate *granularity*, where S and R denotes the set of plagiarism cases and detections.

$$gran(S,R) = \frac{1}{|S_R|} \sum_{s \epsilon S_R} |R_s| \tag{2.2}$$

Precision p is defined as the fraction of correctly classified plagiarism passages. Recall r is the fraction of retrieved plagiarism passages, from S. Macro-averaged precision and recall is unaffected by the length of each passage, while micro-averaged considers each passage in detail. The harmonic mean of precision and recall is calculated with formula (2.3).

$$F_1 = \frac{2}{\frac{1}{r} + \frac{1}{p}} \tag{2.3}$$

The *plagdet* score is calculated with Formula (2.4), where F_1 is the harmonic mean of precision and recall.

$$plagdet(S,R) = \frac{F_1}{log_2(1 + gran(S,R))} \tag{2.4}$$

A *plagdet* scoring script has been published through the PAN workshop website,[4] The script uses macro-averaged precision and recall as default.

[4] http://www.webis.de/research/corpora/corpus-pan-pc-09/perfmeasures.py

2.4 Graph edit distance

Graph edit distance is a measure of how different two graphs are, defined as the minimum amount of edit operations needed to transform one graph into another. An edit operation can be done on either a node or edge, following definition 1 (Riesen and Bunke, 2009; Hu et al., 2009).

Definition 1. *An edit operation between two graphs g_1 and g_2 is either a substitute $(u \rightarrow v)$, insert $(\epsilon \rightarrow v)$ or delete $(u \rightarrow \epsilon)$ operation, where u is a node in g_1 and v is a node in g_2.*

A substitute operation is defined in definition 2, following Riesen and Bunke (2009).

Definition 2. *A substitute operation $(u \rightarrow v)$ consists of an insert $(\epsilon \rightarrow v)$ and a delete $((u \rightarrow \epsilon)$ operation.*

2.4.1 Edit operations

The edit operations needed to transform the source graph g_1 into the target graph g_2, is form an *edit path*. Figure 2.1 illustrates the edit path required to turn the sentence *Mary was kissed by Bob* into the sentence *Bob kissed Mary*.

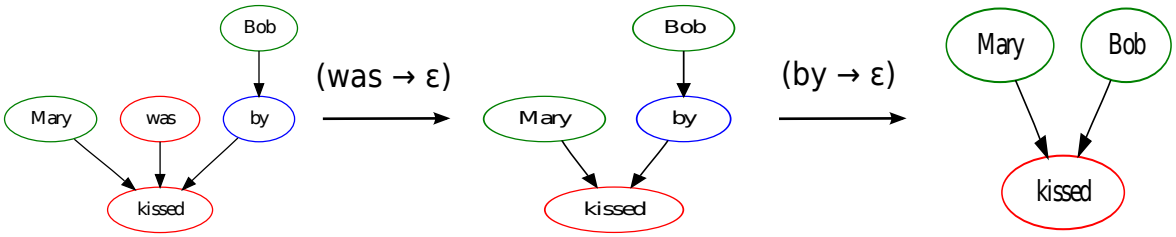


Figure 2.1: An example of edit operations for two graphs

In this case, if only nodes are considered, the edit path of the two graphs in Figure 2.1 can be expressed as in (1).

$$[(was \rightarrow \epsilon),\ (by \rightarrow \epsilon)] \tag{1}$$

Each edit operation has an associated cost defined as a cost function. A typical cost function matches node labels, as well as edge difference. The cost function may differ from application to application, and is a matter of implementation details. This shows how general graph edit distance can be. The algorithm can be modified to solve a specific problem by simply replacing a single function.

2.4.2 Assignment problem

The time complexity for matching n nodes with m nodes, where n and m are the nodes of two graphs, is given by $\frac{n!}{m!}$, making the problem exponential in the number of involved nodes (Fankhauser et al., 2011; Zeng et al., 2009). As a result, there has been a need for alternative approaches.

A fast, but sub-optimal solution has been proposed by Riesen and Bunke (2009). Their solution is based on the assignment problem, which is the problem of assigning n agents to m jobs, where each agent has an individual cost for performing each job. The task is to find the cheapest assignment of the agents. In this case, nodes are considered agents, and the edit operation costs are the different task costs.

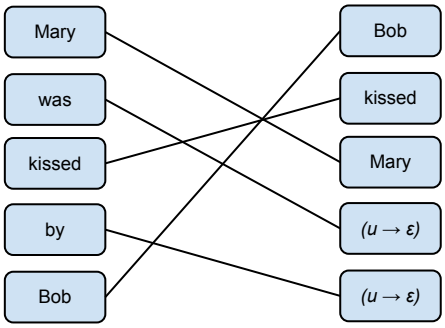


Figure 2.2: Assignment problem

Figure 2.2 illustrates the assignment problem for the two sentences from Figure 2.1. Since the two sentences have unequal number of words, two delete operations ($u \rightarrow \epsilon$) have been included, so every node receive an assignment. The edit cost is the total cost of assignments. In this case, if only node operations are considered, the assignment of the three words *Mary*, *Bob* and *kissed* should cause zero cost, as the words are present in both graphs. The cost of deleting *by* and *was* should be lower than the total cost of substituting them with any of the other words.

2.4.3 Cost matrix

When calculating the graph edit distance between two graphs, a two-dimensional cost matrix can be created, which represents the cost of each possible node edit operation. By creating this cost matrix, the graph edit distance problem is reduced to an assignment problem, which is a simpler problem to solve. The cost matrix is then used as input to an assignment problem algorithm, which returns the minimal cost assignment of the edit operations.

The minimum cost assignment is the approximated graph edit distance of the two graphs. The cost assignment is approximated since each node edit operation is considered individually, only looking at local structure surrounding each node (Riesen and Bunke, 2009). The cost matrix is constructed in the following format:

$$C = \left[\begin{array}{ccc|ccc} c_{1,1} & \cdots & c_{1,m} & c_{1,\epsilon} & \infty & \infty \\ c_{2,1} & \cdots & c_{2,m} & \infty & c_{2,\epsilon} & \infty \\ c_{3,1} & \cdots & c_{3,m} & \infty & \infty & c_{3,\epsilon} \\ \hline c_{\epsilon,1} & \infty & \infty & 0 & 0 & 0 \\ \infty & c_{\epsilon,2} & \infty & 0 & 0 & 0 \\ \infty & \infty & c_{\epsilon,3} & 0 & 0 & 0 \end{array}\right]$$

The matrix is of size $(|N| + |M|) \times (|M| + |N|)$, where $|N|$ and $|M|$ refers to the number of nodes in each graph. Each cell represents the edit operation cost between node n_i and m_j. Each node is permitted only one edit operation. The upper left region of the matrix represents node substitution costs, the upper right region represents node deletion costs, while the lower left region represents node insertion costs. The bottom right consists of substitutions of the form $(\epsilon \rightarrow \epsilon)$, and should not cause any cost.

In Figure 2.2, two extra delete operations had to be added, since the graphs were of unequal size. The algorithm does not necessarily know how to do this.

By adding the additional insert and delete regions mentioned above, the algorithm is free to assign nodes to either of the three operations. Since the cost matrix now is twice the size of the two graphs, twice as many assignments are done. To compensate for this, the bottom right region has been introduced. This region consists of only free edit operations.

As can be seen in the matrix, the insert and delete sections have one cell for each node with the cost $c_{\epsilon,j}$ or $c_{i,\epsilon}$, which is considered the *insert* or *delete cost*. The rest of the cells in these sections have the cost ∞, which is an effective way to restrict each node to only one insert or delete operation.

2.4.4 Assignment problem algorithms

There exists several algorithms which solve the assignment problem, which differ mainly in terms of execution speed. Fankhauser et al. (2011) mention three algorithms which solve the assignment problem: Hungarian algorithm, Munkres algorithm and Volgenant-Jonker. The paper claims that the VolgenantJonker algorithm is the fastest of the three, with a time complexity of $O(n^3)$.

Execution speed is not the primary focus of this thesis, so no in-depth analysis of the different algorithms will be made. However, a simple comparison between two Java implementations of Munkres and VolgenantJonker algorithms will be made in Section 5.5.

Chapter 3

Related work

This chapter considers the related work done on graph edit distance algorithms and the different contributions made to the PAN plagiarism detection task.

3.1 Graph edit distance

There are numerous approaches to the problem of calculating graph edit distance. The different approaches can be divided into suboptimal and optimal approaches. The optimal approaches generally have a very high time and space complexity (Riesen and Bunke, 2009).

As a result, numerous suboptimal approaches have been proposed, that attempt to deal with the complexity of graph edit distance calculation. The different approaches all deal with the problem of defining edit costs, which makes it hard to come up with a general purpose graph edit distance algorithm (Gao et al., 2010).

3.2 Plagiarism detection

Plagiarism detection systems differ in many ways, but all systems face the same problem - large amounts of text data which need to be processed. To cope with this problem, most systems include a candidate retrieval phase which reduces the problem space in an efficient manner. Candidate plagiarised passages are retrieved, and sent further to a detailed analysis phase. The following sections will give a brief summary of the approaches from the top contestants in the PAN11 and PAN12 challenge.

3.2.1 PAN results

Tables 3.3 and 3.1 show the contributions to the PAN11 external plagiarism task. The contributions are ranked by their *plagdet* score. Only the overall score is shown to save space. Potthast et al. (2011, 2010) has a complete list of *plagdet* scores for the different contributions listed in Table 2.2.

Team	plagdet	precision	recall	granularity
Kasprzak and Brandejs (2010)	0.80	0.98	0.89	1.00
Zou et al. (2010)	0.71	0.93	0.80	1.09
Muhr et al. (2010)	0.69	0.93	0.87	1.19
Grozea and Popescu (2010)	0.62	0.93	0.58	1.02
Oberreuter et al. (2010)	0.61	0.92	0.57	1.01
Torrejón and Ramos (2010)	0.59	0.91	0.55	1.00
Pereira et al. (2010)	0.52	0.82	0.60	1.00
Palkovskii et al. (2010)	0.51	0.85	0.49	1.01
Devi et al. (2010)	0.44	0.97	0.38	1.01
Gottron (2010)	0.26	0.53	0.47	1.87
Micol et al. (2010)	0.22	0.97	0.31	2.39
Costa-Jussà et al. (2010)	0.21	0.39	0.41	1.02
Nawab et al. (2010)	0.21	0.58	0.14	1.33
Gupta et al. (2010)	0.20	0.59	0.16	1.15
Vania and Adriani (2010)	0.14	0.93	0.35	7.74
Suárez et al. (2010)	0.06	0.43	0.07	2.74
Alzahrani and Salim (2010)	0.02	0.39	0.06	19.24

Table 3.1: The overall results from the PAN10 challenge

Team	plagdet	precision	recall	granularity
Grman and Ravas (2011)	0.56	0.94	0.40	1.00
Grozea and Popescu (2011)	0.42	.0.81	0.34	1.22
Oberreuter et al. (2011)	0.35	0.91	0.23	1.06
Cooke et al. (2011)	0.25	0.71	0.15	1.01
Torrejón and Ramos (2011)	0.23	0.85	0.16	1.23
Rao et al. (2011)	0.20	0.45	0.16	1.29
Palkovskii et al. (2011)	0.19	0.44	0.14	1.17
Nawab et al. (2011)	0.08	0.28	0.09	2.18
Ghosh et al. (2011)	0.00	0.01	0.00	2.00

Table 3.2: The overall results from the PAN11 challenge

3.2.2 Kong et al.

This team scored first in the detailed comparison sub-task, and third in the candidate retrieval subtask of PAN12. Their candidate retrieval phase consists of queries to the search API introduced in the PAN 2012 challenge (Potthast et al., 2012). The candidates are retrieved with a method based on TF-IDF weighting, stemming and stop-word removal.

The detailed analysis consists of a combination of semantic and structural similarity. Semantic similarity is calculated using a Vector Space Model and structural similarity using an Overlapping Measure Model. The overlapping measure model calculates a similarity score between two sentences by looking at overlapping terms.

As a third step, they introduce a Bilateral Alternating Sorting algorithm, which is used to merge scattered passages. Unfortunately, the algorithm is not described in detail in the notebook paper due to patent pending (Kong et al., 2012).

Team	plagdet	precision	recall	granularity
Kong et al. (2012)	0.738	0.825	0.678	1.01
Suchomel et al. (2012)	0.68	0.893	0.552	1.00
Grozea (2012)	0.678	0.775	0.635	1.04
Oberreuter et al. (2012)[1]	0.674	0.867	0.555	1.01
Torrejón and Ramos (2012)	0.625	0.834	0.501	1.00
Palkovskii and Belov (2012)	0.538	0.575	0.523	1.02
Küppers and Conrad (2012)	0.350	0.776	0.282	1.27
Sánchez-Vega et al. (2012)	0.310	0.538	0.349	1.577
Gillam et al. (2012)	0.309	0.898	0.0190	1.02
Jayapal (2012)	0.045	0.623	0.076	6.93

Table 3.3: The overall results from the PAN12 challenge

3.2.3 Suchomel et al.

This team scored overall second in the PAN12 challenge (Potthast et al., 2012). For the detailed analysis, they used a combination of lexicographically sorted n-grams and stop-word n-grams.

A sentence can be expressed in many ways by shuffling around the words, and the lexicographically sorted n-grams approach detects shuffling of word order. They used sorted five-grams with stop-word removal. Sorted n-grams do give a good representation of the semantics of the sentence, as the method focuses on word presence, and disregards word order.

While the sorted n-gram approach is a good representation of semantics, it disregards syntactical structure. The stop-word n-gram approach does the opposite. They remove all non stop-words and detect eight-gram matches of stop-words only. This way syntactical similarities between two texts are captured. Consider the two plagiarised sentences (2) and (3), taken from Stamatatos (2011). The two sentences have the exact same stop-word seven-gram $[this, from, the, in, the, of, the]$.

This came into existence likely from the deviance in the time-period of the particular billet. (2)

This probably arose from the difference in the duration of the respective offices. (3)

As many words as possible have been replaced, without disrupting the syntactic structure of the sentences. The reasoning behind this approach can be derived from the work of Koppel et al. (2006), where a new linguistic feature called *meaning-preserving stability* is introduced. It measures how easily a term is replaced by semantically equivalent terms. In order to replace a term, without changing the meaning of the sentence, the term can be replaced with synonyms. Stop-words usually do not have any synonyms, which makes them hard to replace. When a plagiariser tries to modify a text passage, the stop-words are likely to remain intact, unless the entire sentence is

restructured. In other words, the syntactical structure can be represented with stop-word n-grams. The approach detects cases where words are replaced with synonyms, without doing synonym lookups, which can be expensive in terms of execution time.

In order to speed up retrieval, they represent each n-gram with the 32 highest-order bits of its MD5 hash function (Suchomel et al., 2012). This is primarily a performance optimization, but it allows the system to deal with large text corpora, which is quite common in the field of plagiarism detection.

3.2.4 Grman & Ravas team

This team scored overall first in the PAN11 challenge (Potthast et al., 2011). Their approach differs from the popular n-gram matching approach. Their goal was to create a similarity detection which is invariant against changes of word order, omissions/additions of words. Word order is crucial for n-gram matching, thus making this a bad approach. The team's similarity detection is based on the number of matching words in an intersection of source and suspicious document passages.

Instead of just matching words in their original form, each word goes through a preprocessing phase, which includes text translation, stemming, stop-word removal and synonym normalization with WordNet. If potential plagiarised sentences are adjacent, they are merged into a single passage. This means that plagiarism is detected at paragraph level, instead of sentence level (Grman and Ravas, 2011).

3.2.5 N-gram based approaches

There are numerous n-gram based approaches among the PAN contestants. The following teams use n-gram matching as their core approach. Grozea and Popescu (2011) match n-grams of a fixed size of 256 characters (not words), which defines their candidate document retrieval phase. They retrieve all documents pairs which have 64 or more matching n-grams of 256 characters.

After retrieving the most similar document pairs, each document pair is analysed in detail. The detailed analysis phase sorts all the n-grams in the two documents with a radix sort specialised in sorting n-grams. The final similarity is then measured as the intersection of the two sets of n-grams.

Oberreuter et al. (2011) utilise four-grams with stop-word removal for retrieving candidate documents. If two documents have two four-gram matches within the same paragraph, they are potential candidates. For the detailed analysis, they utilise word tri-grams without stop-word removal.

Chapter 4

Implementation

This chapter describes the implementation of the prototype system made for this thesis. The goal of the system was to find plagiarised passages within a text corpus and to use the results to evaluate the graph edit distance approach. The text corpus used was the dataset from PAN11, described in Section 2.3.1. In addition, the preceding dataset from PAN10 was used to tune parameters. The system is written in Java, and is multi-threaded. In other words, the system utilises all available processors in a system, which is beneficial in order to process large amounts of data. The experimental results generated from the system are presented in Chapter 5.

4.1 Architecture

Figure 4.1 gives an overview of the flow between the main components of the system. Each phase consists of several worker threads, which receive jobs from a queue and then send the output to the next phase.

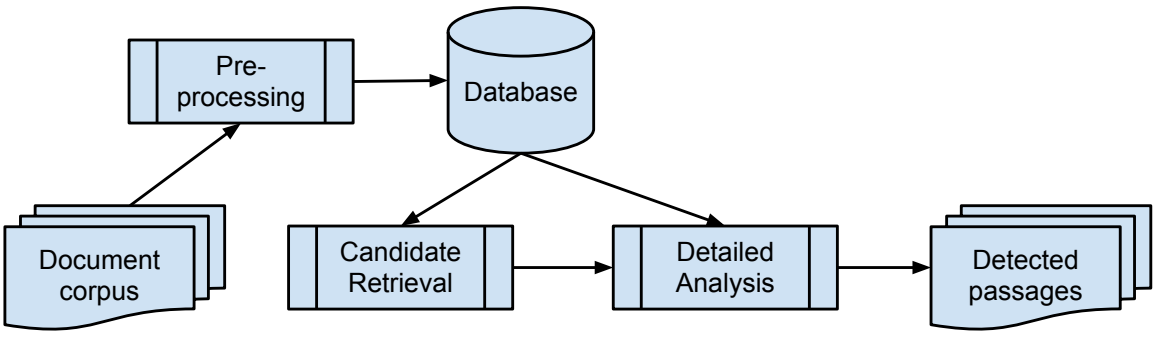


Figure 4.1: Overview of the main modules of the system

The system reads all files in a document corpus, and outputs annotated XML files with pointers to plagiarism passages. The different phases of the system will be described in detail in the following sections.

4.2 Text pre-processing

Each document undergoes a pre-processing phase, which processes the text in various ways. The output of the pre-processing phase is then saved to a database, to speed up execution. Figure 4.2 gives an overview of the pre-processing phase.

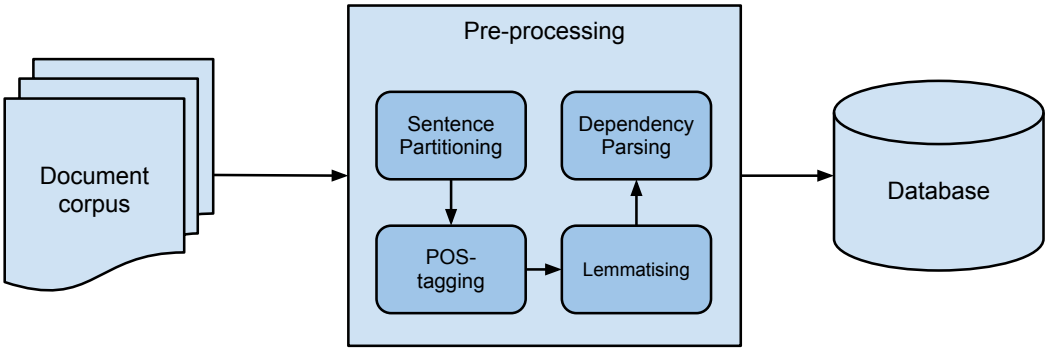


Figure 4.2: Text pre-processing phase

4.2.1 Sentence partitioning

In order to find plagiarism passages, the text needs to be partitioned into smaller pieces. Partitioning text into sentences was a natural choice, especially since dependency graphs represent dependencies within a sentence scope. Text was partitioned into sentences using the Stanford Tokenizer.[1] Some of the documents in the corpus described in Section 2.3.1 contain text not delimited by punctuation. For instance, many documents contain a table of contents without punctuation, which caused the tokeniser to output sentences of 500 words or more. In some cases the tokeniser output sentences with very few words. These falsely tokenized sentences caused problems later in the system, and had to be dealt with. A simple, yet effective solution was to filter out sentences with less than three or more than eighty.

4.2.2 POS-tagging and lemmatising

After partitioning the text into sentences, each sentence is processed by the Stanford Part-Of-Speech Tagger.[2] During this phase, each term in a sentence is given a POS-tag and lemma. The tagger requires pre-trained models for each language as input. The system used the model called *english-left3words-distsim.tagger*, which is included in the download package.

The lemma of each word is identified using the *Morphology* class from the Stanford Part-Of-Speech Tagger library. The output is then sent further to dependency parsing.

[1] http://nlp.stanford.edu/software/tokenizer.shtml
[2] http://nlp.stanford.edu/software/tagger.shtml

4.2.3 Dependency parsing

Dependency graphs are created using MaltParser,[3] which is a data-driven dependency parser. MaltParser requires POS-tag as well as word token as input, and outputs labelled dependency relations. Similarly to Stanford Part-Of-Speech tagger, the MaltParser also require a pre-trained model as input. The system used the model called *engmalt.linear-1.7.mco*, which is available on the MaltParser website. The reason for choosing this model over the alternative *engmalt.poly-1.7.mco* model was due to parsing speed. The latter is significantly slower, but uses less memory (Nivre and Hall, 2010).

Table 4.1 describes the output from MaltParser. Only the last two fields are produced by MaltParser, the rest were included in the input. Dependency refers to the id of the *head* token, explained in Section 2.2.1.

id	word	lemma	POS-tag	dependency	deprel
1	This	this	DT	4	nsubj
2	is	be	VBZ	4	cop
3	an	a	DT	4	det
4	example	example	NN	0	null
5	of	of	IN	4	prep
6	dependencies	dependency	NNS	5	pobj
7	within	within	IN	6	prep
8	a	a	DT	9	det
9	sentence	sentence	NN	7	pobj
10	.	.	.	4	punct

Table 4.1: MaltParser example output

4.2.4 Pre-processing output

After a document has gone through the pre-processing phase, it contains the information needed to construct dependency graphs for each sentence. Each pre-processed sentence is then written to a database, allowing fast look up in the later stages of the system. Each sentence contains a reference to its document name, sentence number, character offset and length. The offset points to the character where the sentence starts. Each sentence contains a list of tokens. Each token consists of the fields described in Table 4.1.

The database used is a high-performance open source NoSQL database called MongoDB.[4] NoSQL is a class of database management systems that, contrary to popular relational database systems such as MySQL, do not rely on structured relations between tables. This makes insertions and queries in a NoSQL database simpler, and faster, than MySQL databases. Kennedy (2010) shows a performance comparison between MongoDB and SQL Server 2008. While the comparison might be biased, and

[3] http://www.maltparser.org/
[4] http://www.mongodb.org/

only covers one alternative, it still shows a significant improvement for basic insertions and queries.

The primary use case for this system is to do simple insertions and queries. All 35 million sentences are stored in the database, indexed by an unique id, which is a combination of file name and sentence number. The database was set to report incidents of queries taking longer than 100 milliseconds. No incidents were reported during test runs. The reason for doing storing the pre-processing data, instead of doing parsing on the fly, was to speed up the latter phases of the system. This way tuning parameters and doing test runs during development was faster.

4.3 Candidate retrieval

Due to the size of the PAN11 data set, each document has to go through a candidate retrieval phase, which retrieves candidate sentence passages in an efficient manner. The precision is of no concern during this phase, since the candidates are sent further to the detailed analysis phase. The goal of this phase is to achieve a high recall, while reducing the problem space as much as possible.

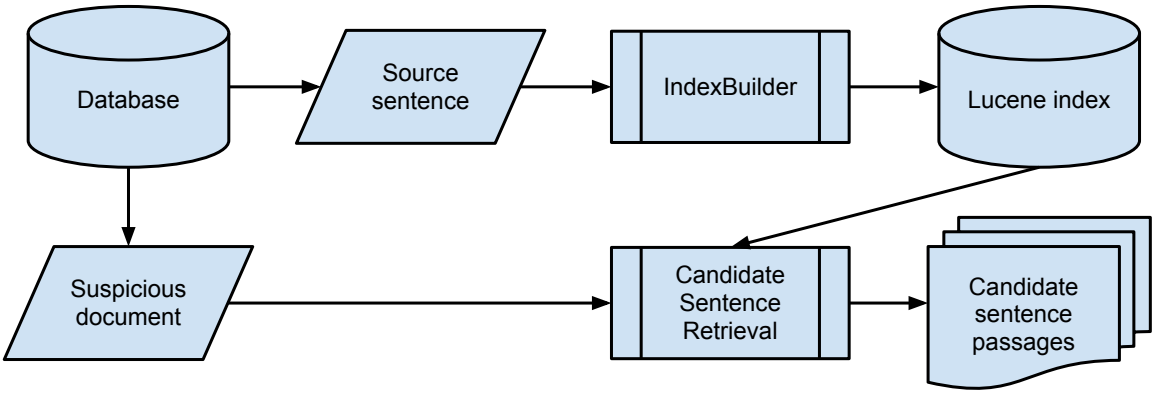


Figure 4.3: Candidate retrieval phase

4.3.1 Sentence indexing

All source sentences were sent to the IndexBuilder in Figure 4.3. The sentences were stored in a TF-IDF weighted index, using Apache Lucene,[5] which is a Java information retrieval library. The components from Apache Lucene are thread safe, which allows indexing and search to be parallelised by many concurrent threads.

When indexing the corpus, each sentence is identified by file name and sentence number. Each sentence is represented by its lemmas in the index. The index is then saved to the file system.

[5] http://lucene.apache.org/

4.3.2 Candidate sentence retrieval

When searching for similar sentences for a given sentence, a class named *MoreLikeThis* from Lucene is used. Queries to *MoreLikeThis* rank each sentence in the index by the sum of matching terms, weighted by their TF-IDF values.

Each sentence can potentially be matched with approximately 20 million source sentences. When retrieving candidate sentences, the n most similar sentences passages are retrieved for each document. As a result, the detailed analysis needs to analyse only a small number of (n) sentence pairs per document.

4.4 Detailed analysis

The detailed analysis analyses every candidate sentence passage in detail, and decides whether or not the passage is plagiarised. The graph edit distance algorithm described in this section is used as the detailed similarity measure between two graphs, and is the main focus of this thesis. Figure 4.4 gives an overview of the detailed analysis phase.

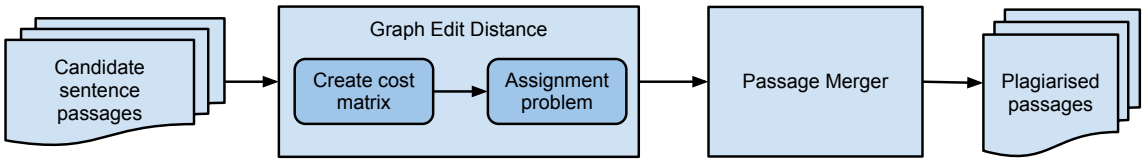


Figure 4.4: Overview of the detailed analysis phase

The output of the detailed analysis is one file for each suspicious document, with pointers to plagiarism passages. The results are then evaluated using the *plagdet* scoring script described in Section 2.3.2.

Most of the functionality is inspired by the work of Riesen and Bunke (2009). The algorithm described in this section is implemented by the author, with the exception of the assignment problem algorithm used. The implementation will be described with a concrete example, given by sentence (4) and (5).

After years of searching, the captain found the treasure on the island. (4)

The treasure was found by captain Scott after searching several years. (5)

These sentences are relatively simple, so the details of the algorithm is easier to grasp. Figure 4.5 represents the dependency graphs for the two sentences.

The output graph edit distance of these two sentences is 4.25 with the normalised distance 0.327. The distance is normalised with Equation (4.1), where $|N|$ and $|M|$ refer to the number of words in each sentence. The reason for performing normalisation is to treat sentences of different sizes equally.

$$normdist = \frac{dist}{\frac{|N|+|M|}{2}} = \frac{4.25}{\frac{14+12}{2}} = 0.327 \tag{4.1}$$

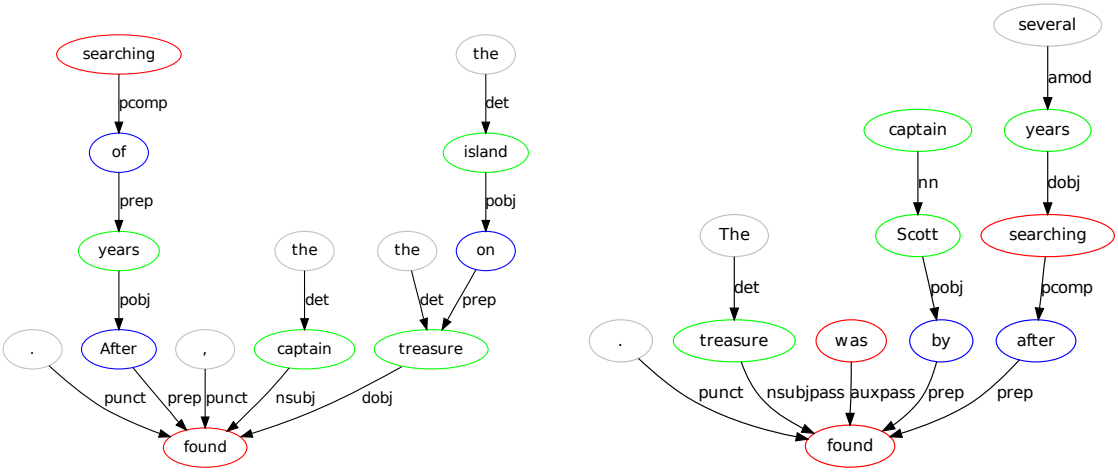


Figure 4.5: Dependency graph for sentence (4) and (5)

Table 4.2 describes the edit operations required to turn sentence (4) into sentence (5), with their respective cost. As can be seen in the table, some substitute operations, such as $(year \rightarrow year)$, have the same label yet cost more than zero. This can be explained with the addition of edge difference. Each step of the algorithm will be described in detail in the following subsections.

Edit operation	**cost**
$(year \rightarrow year)$	0.50
$(of \rightarrow by)$	0.25
$(captain \rightarrow captain)$	0.50
$(the \rightarrow \epsilon)$	0.25
$(treasure \rightarrow treasure)$	0.50
$(on \rightarrow several)$	0.75
$(the \rightarrow be)$	1.25
$(island \rightarrow Scott)$	0.25

Table 4.2: Edit operations for sentence (4) and (5)

4.4.1 Cost matrix

The graph edit distance algorithm takes three integers S, I and D as input, which determine the highest possible substitute, insert and delete cost. For the results in Section 5.3 the following input parameters were used:

- $S = 2$
- $I = 1$
- $D = 1$

As explained in Section 2.4, a substitute operation consists of one delete and one insert operation, which is the reasoning behind these input parameters. The cost matrix is constructed as described in Section 2.4.3.

Table 4.3 lists the substitution costs for the two sentences from Figure 4.5. This table represents the upper left region of the cost matrix explained in Section 2.4.3. The other three regions are omitted to save space. The edit path listed previously is highlighted with blue text.

	the	treasure	be	find	by	captain	Scott	after	search	several	year	.
after	1.38	1.50	1.38	1.50	0.25	1.50	1.50	**0.00**	1.50	0.75	1.50	1.25
year	1.50	1.50	1.50	2.00	1.50	1.50	1.00	1.50	1.50	1.50	**0.50**	1.50
of	1.38	1.50	1.38	1.50	**0.25**	1.50	1.50	0.25	1.50	0.75	1.50	1.25
search	1.50	1.50	0.75	1.25	1.50	1.50	1.50	1.50	**0.00**	1.50	1.50	1.50
,	1.13	1.50	1.13	1.00	1.25	1.38	1.50	1.25	1.50	1.38	1.50	1.00
the	**0.00**	1.50	1.25	1.25	1.38	1.50	1.50	1.38	1.50	1.50	1.50	1.13
captain	1.50	0.75	1.50	2.00	1.50	**0.50**	0.75	1.50	1.50	1.50	1.50	1.50
find	1.25	2.00	0.50	**0.00**	1.50	1.75	2.00	1.50	1.25	1.75	2.00	1.00
the	0.00	1.50	1.25	1.25	1.38	1.50	1.50	1.38	1.50	1.50	1.50	1.13
treasure	1.50	**0.50**	1.50	2.00	1.50	0.75	0.75	1.50	1.50	1.50	1.00	1.50
on	1.38	1.50	1.38	1.50	0.25	1.50	1.50	0.25	1.50	**0.75**	1.50	1.25
the	0.00	1.50	**1.25**	1.25	1.38	1.50	1.50	1.38	1.50	1.50	1.50	1.13
island	1.50	0.75	1.50	2.00	1.50	0.75	**0.25**	1.50	1.50	1.50	1.50	1.50
.	1.13	1.50	1.13	1.00	1.25	1.38	1.50	1.25	1.50	1.38	1.50	**0.00**

Table 4.3: The substitute region of the cost matrix

As can be seen in Table 4.3, each word has been assigned a cost, except for the word *the* (the first and second ones), which is deleted. The sentence on the horizontal has two more words than the vertical sentence, thus there are no available words to substitute with. A full cost matrix would show that the delete operation is the cheapest assignment for these nodes.

4.4.2 Substitute costs

The substitute cost function is a combination of node relabel cost and edge difference, as described with Equation (4.2). SC is the substitute cost between node n and m, NLD the node label difference, ED the edge difference and S is the highest possible substitute cost given as input parameter to the algorithm (2 in this case). The sum of $NLD_{n,m} + ED_{n,m}$ should be a number between 0 and 1.

$$SC_{n,m} = \frac{NLD_{n,m} + ED_{n,m}}{2} \times S \tag{4.2}$$

Node label difference

The node relabel function between two nodes n and m matches the nodes' lemmata. If the two nodes have matching lemmata, the node label difference is 0.

Given unequal lemmata, the node label difference is given by a POS substitute weight. The POS substitute weight represents the difference between two POS-tags. For instance, replacing a verb with a verb should be less costly than replacing a verb with a noun. As a result, relabel operations between words with POS-tag in the same category receive a POS substitution weight of 0.25. Other relabel operations receive a weight of 1. The different categories are as follows:

- adjectives

  JJ, JJR, JJS,

- adverbs

  RB, RBR, RBS

- nouns

  NN, NNS, NNP

- verbs

  VB, VBD, VBG, VBN, VBP, VBZ

These weights should be considered as heuristics, and leave room for improvement.

Edge difference function

The edge difference between two nodes n and m is calculated by matching edges by their *deprel* label. Similarly to the problem of assigning node costs in Section 4.3, the problem of calculating edge difference can also be reduced to an assignment problem. An edge cost matrix is constructed, where each cell represents the edit cost between each edge. This approach is similar to the approach by Riesen and Bunke (2009).

Each edge is either substituted, inserted or deleted. These operations should ideally be weighted, similarly to node edit operations. Unfortunately, due to time constraints, only edge insert and delete operations are weighted. Table 4.4 presents the different edge insert and delete weights, based on their deprel. These weights are, similarly to the POS substitute weights, considered as simple heuristics.

4.4.3 Node insert and delete costs

The node insert and delete costs is the POS insert and delete weight taken from Table 4.5. Stop-word typically receive a low POS delete and insert weight. Punctuation and other symbols receive zero weight, and should have any cost. This way, the edit distance is more robust, and is not as sensitive to noise from stop-words. The weights are created the same way as in the previous sections, and are far from ideal.

abbrev	0.25	cop	0.25	nsubjpass	1	purpcl	0.75
acomp	1	csubj	1	num	0.75	quantmod	0.25
advcl	0.75	dep	1	number	0.25	rcmod	0.75
advmod	0.75	det	0.25	parataxis	0.25	ref	0.25
agent	1	dobj	1	partmod	0.75	root	0
amod	0.75	expl	0.1	pcomp	1	tmod	0.75
apos	0.5	infmod	0.75	pobj	1	xcomp	1
attr	0.75	iobj	1	poss	0.25	xsubj	1
aux	0.25	mark	0.75	preconj	1	dep	1
auxpass	0.25	mwe	1	predet	0.25	aux	0.25
cc	0.25	neg	1	prep	0.5	arg	1
ccomp	1	nn	0.75	prepc	0.75	comp	1
complm	1	npadvmod	0.75	prt	0.25	obj	1
conj	0.25	nsubj	1	punct	0	subj	1
mod	0.75						

Table 4.4: Deprel insert and delete weights

CC	0.5	CD	0.5	DT	0.25	EX	0.1
FW	1	IN	0.5	JJ	0.75	JJR	0.75
JJS	0.75	LS	0	MD	0.25	NN	1
NNP	1	NNPS	1	NNS	1	PDT	0.25
POS	0.1	PRP	0.25	PRP$	0.25	RB	0.75
RBR	0.75	RBS	0.75	RP	0.25	SYM	0.5
TO	0.1	UH	0.1	VB	1	VBD	1
VBG	1	VBN	1	VBP	1	VBZ	1
WDT	0.5	WP	0.5	WP$	0.5	WRB	0.5
,	0	“	0	”	0	(	0
)	0	–	0	.	0	:	0
$	0						

Table 4.5: POS insert and delete weights

4.4.4 Assignment problem algorithm

The assignment problem algorithm used is an implementation of Munkres algorithm made available by Nedas (2008). The algorithm takes the cost matrix as input, and outputs the lowest cost assignment.

4.4.5 Plagiarism threshold

In order to determine whether or not a given graph edit distance is plagiarism or not, a threshold has been introduced. If the graph edit distance score is below this threshold, it is considered plagiarism.

The threshold was tuned by performing plagiarism search on the PAN10 corpus, which is offered as training corpus on the PAN11 web page. It is important to tune the threshold on a different corpus than the test corpus, as tuning on the test corpus would give an unfair advantage.

4.4.6 Passage merging

The system has partitioned the documents into sentences, and the plagiarism detection is done on sentence level. The output from the detailed analysis is a list of plagiarism passages for each file. Before writing these passages to file, all adjacent passages are merged into one passage. The main reason for doing this is to reduce granularity.

In addition to merging sentence passages which are side by side, all sentences within a reasonable distance are merged. This distance of characters is referred to as *mergedist*. By merging all the text between two passages, the intermediate text is also considered plagiarised. This is arguably a bad approach, but is done due to the high regard of the granularity measure in the PAN challenge.

4.4.7 Algorithm variations

This section will cover some different edit cost and graph representation variants. These variants are considered highly experimental.

Undirected graphs

So far the edges in dependency graphs have been defined as directed. By not considering the edge direction, the representation becomes simpler, and may be more robust to noise. By comparing directed graphs with undirected graphs, it is possible to identify the better graph representation.

Unweighted edit operations

The different edit operations weights described in Section 4.4.2 and 4.4.3 are relatively simple, and are not perfectly tuned. As a result, they might actually have a negative impact on performance. It is important to compare weighted and unweighted edit operations to make sure the weights are worthwhile.

Edge dependent matching

The previously described graph edit distance only matches edge labels. One potential improvement is to add edge dependent matching to the edge difference function described in Section 4.4.2.

The implemented solution considers edges equal if both *deprel* and dependent nodes are equal. This solution might be too specific, and vulnerable to graph noise.

Chapter 5

Results

Experimental runs were made on the PAN10 and PAN11 external detection data sets described in Section 2.3. The results of the detailed analysis are measured using the *plagdet* measure, then compared to the results listed in Section 3.2.1 and Potthast et al. (2010, 2011).

5.1 Parameter tuning

Before running the system on the PAN11 datasets, some parameters required tuning on a similar data set. Different *mergedist* and plagiarism threshold values were given as input, and multiple runs were made. Table 5.1 lists the different *plagdet* scores for different *mergedist* and threshold values. Ideally more runs should have been made, but due to time constraints there was no time to perform a thorough analysis.

Mergedist	**Threshold**	**Plagdet**	**Recall**	**Precision**	**Granularity**
1000	0.25	0.110	0.168	0.871	4.873
1000	0.3	**0.112**	0.175	0.766	4.827
1000	0.325	**0.112**	0.178	0.683	4.794
1000	0.35	0.110	0.182	0.590	4.772
1000	0.375	0.106	0.185	0.490	4.756
1000	0.4	0.101	0.187	0.401	4.758
1000	0.45	0.089	0.192	0.276	4.803
1500	0.275	**0.127**	0.183	0.801	4.068
1500	0.3	**0.127**	0.186	0.737	4.047
1500	0.35	0.122	0.192	0.549	4.001
1500	0.375	0.117	0.195	0.448	4.000
1500	0.4	0.110	0.198	0.361	4.001
1500	0.45	0.094	0.201	0.243	4.069

Table 5.1: PAN10 results for different thresholds

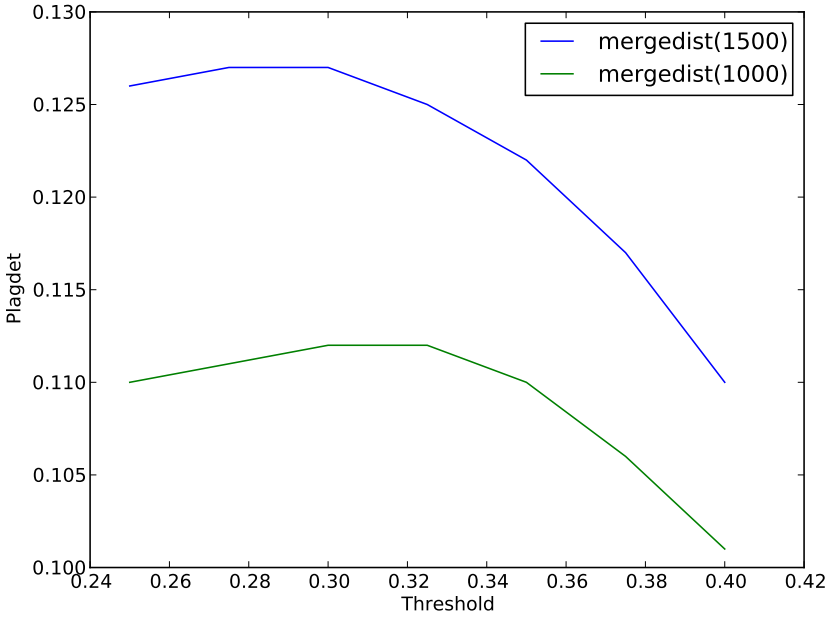


Figure 5.1: Plagdet values for different input parameters

The results are illustrated in Figure 5.1. Based on these results, the plagiarism threshold used for the PAN11 data set was set to **0.3**. The *mergedist* value was set to **1500**. There is potential for a great performance increase by tuning parameters more thoroughly.

5.2 Candidate retrieval phase

Getting good results in the candidate retrieval phase is not the goal of this thesis, but is still a necessity in order to get good results in the detailed analysis phase. The goal of the candidate retrieval phase is to achieve as high as possible recall, while reducing the problem space to a manageable size. Table 5.2 and 5.3 list the *plagdet* results for the candidate retrieval phase on the PAN10 and PAN11 data sets. The different values of n refer to the amount of sentence passages retrieved per document.

	Plagdet	**Recall**	**Precision**	**Granularity**
n=150	0.05	0.21	0.09	4.63

Table 5.2: PAN10 candidate retrieval results

Unfortunately, the recall is very low compared to the PAN10 results listed in Table 3.1. An improved candidate retrieval phase would definitely increase performance.

The candidate retrieval results for PAN11 are better, when compared to the contributions listed in Table 3.3. The other contributions to the PAN11 challenge achieve significantly lower recall, compared to the PAN10 results. As a result, a recall of 0.27 is better than 7 of 9 contributions, and can be considered satisfying.

	Plagdet	**Recall**	**Precision**	**Granularity**
n=50	0.09	0.21	0.20	3.88
n=150	0.06	0.27	0.13	5.64

Table 5.3: PAN11 candidate retrieval results

Due to time constraints, no other values of n were tested on the PAN10 corpus, so $n = 150$ was used for getting the detailed analysis scores.

5.3 Detailed analysis

Table 5.4 presents the results from the detailed analysis for PAN11. Two different *mergedist* values are listed, to demonstrate the difference of using the passage merging functionality described in Section 4.4.6.

mergedist	**Threshold**	**Plagdet**	**Recall**	**Precision**	**Granularity**
1500	0.3	0.224	0.223	0.479	1.56
0	0.3	0.146	0.204	0.646	3.38

Table 5.4: PAN11 detailed analysis results

As a summary, the *plagdet* score of 0.224 could have been better, but would have been enough to achieve 5th out of 10[1] contestants in the PAN11 challenge. Compared to the candidate retrieval results in Table 5.3, the recall was reduced by **0.033** and precision increased by **0.349**. The Granularity was increased by **4.08**. The total *plagdet* score improvement from the candidate retrieval phase was **0.164**.

5.4 Different graph representations and edit cost functions

In order to determine the best graph representation and edit cost variants, the different variants from Section 4.4.7 were tried out on the PAN11 data set.

Ideally this experimentation should have been done on the PAN10 training set, but due to some technical issues and time constraints, the only available comparison was made on the PAN11 data set. As a result, the different comparisons cannot be compared directly against the other contestants of the PAN11 challenge. However, the comparison is just as good to compare the different implementation variations against each other. The following variants were tried out:

[1]PAN11 had 9 contestants, plus one if this contribution had been made

1. Directed edges, deprel matching and weighted edit operations
2. Undirected edges, deprel matching and weighted edit operations
3. Directed edges, deprel matching, weighted edit operations and edge dependent matching
4. Directed edges, deprel matching and unweighted edit operations
5. Undirected edges, deprel matching and unweighted edit operations
6. Directed edges, deprel matching, unweighted edit operations and edge dependent matching
7. Undirected edges, deprel matching, unweighted edit operations and edge dependent matching

Variant	**Plagdet**	**Precision**	**Recall**	**Granularity**
1	0.224	0.479	0.223	1.56
2	0.221	0.462	0.223	1.57
3	0.184	0.59	0.140	1.34
4	0.189	0.71	0.133	1.28
5	0.181	0.703	0.126	1.26
6	0.105	0.690	0.067	1.24
7	0.101	0.681	0.065	1.24

Table 5.5: PAN11 detailed analysis with different implementation details

Variant 1 was the original version, and is the one used for the results in the Section 5.3. The input parameters are probably slightly biased in favour of variant 1, since the parameters from Section 5.1 were retrieved using this variant. Variant 4-7 with unweighted edit operations achieve a significantly higher precision, but much lower recall. The original variant with directed edges, only *deprel* edge matching matching and edit cost weights received the best *plagdet* score.

5.5 Execution speed

Figure 5.2 illustrates a comparison between the Munkres implementation described in Section 4.4.4, and a Java implementation of the VolgenantJonker assignment problem

algorithm.[2]

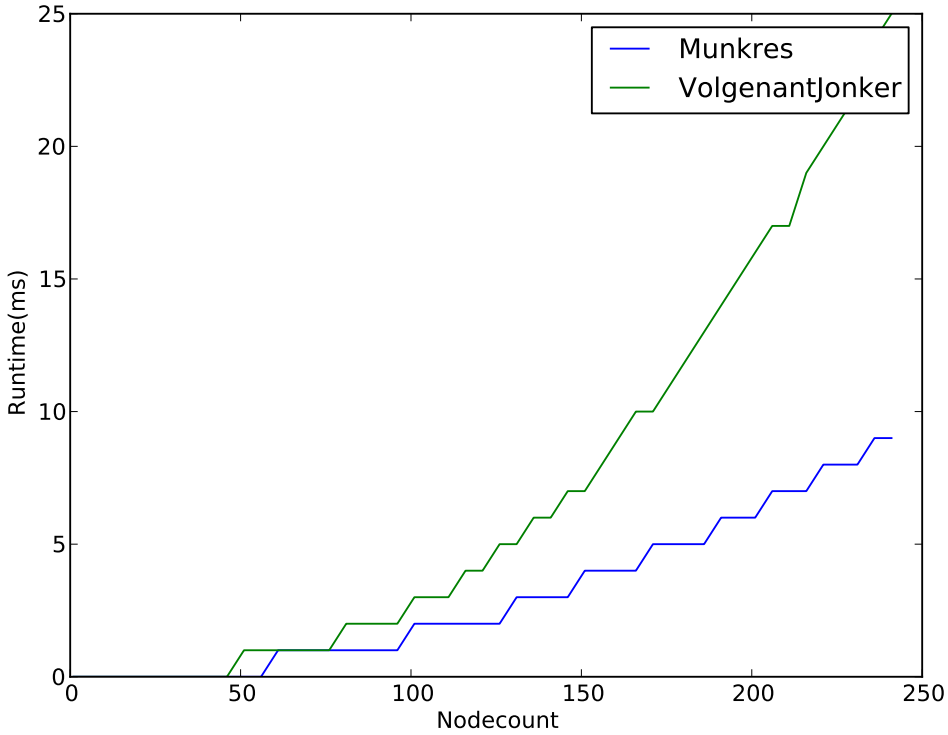


Figure 5.2: Runtime comparison of graph edit distance computations, using both Munkres and VolgenantJonker assignment algorithms, for graphs of size 1 to 250 nodes

The Y-axis describes the runtime in milliseconds it takes to calculate[3] the graph edit distance of two graphs with n nodes, where n refers to the node count in the X-axis. The speed (Y-axis) for each node count value (X-axis) is the median of 1000 runs. This way the comparison is less affected by outside factors, such as other processes using system resources.

Figure 5.3 represents the runtime of the two assignment problem algorithms only. The assignment problem calculations are expected to have the most impact on the graph edit distance runtime. The reason for spiky graphs is due to the low run times. Arguably graphs of larger sizes should be compared, but considering very few sentences are above 100 words, a 1 to 250 size scope should be representative of actual running times.

The speed results are somewhat surprising, as Fankhauser et al. (2011) claims that the VolgenantJonker algorithm is the fastest of the two alternatives. The results in Figure 5.2 represent a specific implementation of the algorithm, and may not be optimal. It is natural to question the specific VolgenantJonker implementation used, and as a result, the Munkres implementation is used to generate the results for this thesis.

[2]http://code.google.com/p/java-k-best/

[3]The benchmark was done on a laptop with a Intel Core i7-3517U Processor.

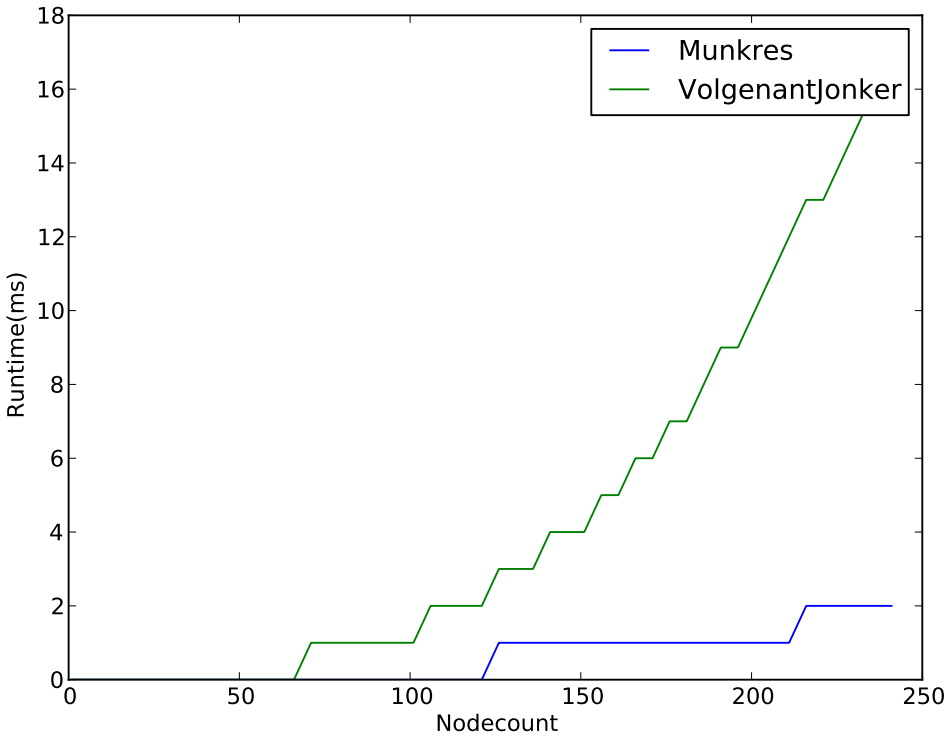


Figure 5.3: Runtime comparison of the Munkres and VolgenantJonker assignment algorithms, for graphs of size 1 to 250 nodes

Chapter 6

Discussion

6.1 PAN evaluation

As described in Section 2.3.2, the PAN results are ranked by the *plagdet* measure, which is a combination of precision, recall and granularity. Suchomel et al. (2012) identified a problem with the granularity measure, illustrated by two methods:

1. 100% recall, 100% precision, granularity 3
2. 33.33% recall, 33.33% precision, granularity 1

Method 1 correctly detects all cases of plagiarism, with no false positives. Unfortunately, method 1 has not implemented a proper adjacent passages merging function, and receive a bad granularity score. Method 2 only detects one third of the plagiarism cases, and only one third of the detections were correct. Based on the above examples, it is reasonable to claim that method 1 is preferred.

Potthast et al. (2012) stresses the fact that not only the *plagdet* score should be considered, and rankings based on the F_1 harmonic mean of precision and recall. Table 6.1 lists the PAN11 contributions ranked by harmonic mean. The two runs named Røkenes are the two runs from Section 5.3, with *mergedist* 0 and 1500, placing fourth when only recall and precision are considered.

Team	F_1	**precision**	**recall**	**granularity**
Grman and Ravas (2011)	0.561	0.94	0.40	1.00
Grozea and Popescu (2011)	0.479	0.81	0.34	1.22
Oberreuter et al. (2011)	0.367	0.91	0.23	1.06
Røkenes (mergedist 0)	**0.310**	0.646	0.204	3.38
Røkenes (mergedist 1500)	**0.304**	0.479	0.223	1.56
Torrejón and Ramos (2011)	0.268	0.85	0.16	1.23
Cooke et al. (2011)	0.248	0.71	0.15	1.01
Rao et al. (2011)	0.236	0.45	0.16	1.29
Palkovskii et al. (2011)	0.212	0.44	0.14	1.17
Nawab et al. (2011)	0.136	0.28	0.09	2.18
Ghosh et al. (2011)	0.02	0.01	0.00	2.00

Table 6.1: PAN11 results ranked by precision and recall

6.2 Standalone detailed analysis evaluation

The system described in this thesis is a full-fledged plagiarism detection system. There are many factors beyond the detailed analysis which may have affected the overall result in a negative or positive way.

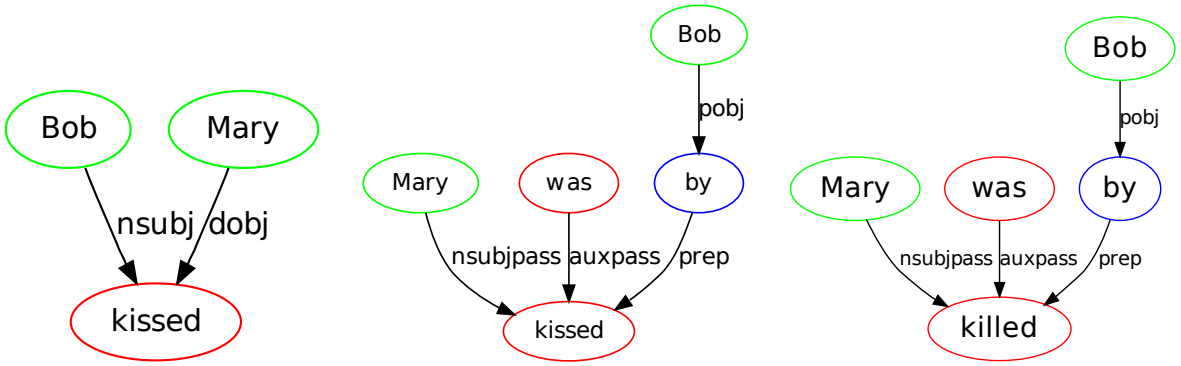
For starters, the sentence partitioning may falsely partition sentences, which leads to errors in the later stages of the system. The median character length of the PAN10 and PAN11 plagiarised passages are 3672 and 2688 respectively. One could also argue that plagiarism usually is done on larger sequences of text than sentences, which is an argument for partitioning text into larger pieces than sentences.

The candidate retrieval phase excludes many potential plagiarism passages, and is probably the phase with the most negative impact on the total score. The recall from the candidate retrieval phase is lower than many state-of-the-art systems, which means that the detailed analysis phase never will be able to achieve the same recall as the top contestants in the PAN challenge.

In order to fully evaluate the graph edit distance algorithm with unbiased results, it should be applied to a problem of detecting similarity between sentences pairs.

6.3 Edge difference

The edge difference function described in Section 4.4.2 is based on the edge labels only. The dependent of the edge is not considered at all, and the question is whether the edge dependent should be added to the equation. Consider the sentences in Figure 6.1.



1) *Bob kissed Mary* 2) *Mary was kissed by Bob* 3) *Mary was killed by Bob*

Figure 6.1: Example of sentences with similar structure, but different meaning

Needless to say, sentence 1 and 2 have the same meaning, while sentence 3 is very different. Table 6.2 shows the edit distances between the three sentences.

The edit distance between sentence 2 and 3 is the lowest. This might indicate that the algorithm favours structural similarity over semantic content, which can be considered a weakness.

Sentence pair	Edit distance	Normalised edit distance
1 & 2	2.5	0.625
1 & 3	2.75	0.687
2 & 3	**0.25**	**0.05**

Table 6.2: Edit distances where structural similarity is favoured over semantic similarity

6.4 Preprocessing output size

When searching for plagiarism, the source documents will typically be analysed multiple times. To avoid dependency parsing multiple times, the dependency graphs are saved to disk. The dependency graphs contain more information for each words, which leads to larger file sizes. As explained in Table 4.1, each token has an id, word, lemma, POS-tag, relation and a deprel tag. In addition, each sentence has information about sentence number, offset and length.

Needless to say, the file sizes increase with a minimum of 6 times the original file size, and a lot more in practice. This can cause problems for large text corpora. Since the candidate retrieval phase reduces the number of sentences, dependency parsing on the fly might be a good idea.

The system implemented performed the dependency parsing first, and saved the output to a database. This way the latter stages of the system ran faster, which was useful during development. For a production system, this should not be a motivation, so parsing on the fly is the recommended solution.

Chapter 7

Conclusion & Future Work

The main focus of this thesis was to explore the usage of graph-based representations of text. A graph edit distance algorithm was implemented and applied to the problem of detecting plagiarism in text.

A full-fledged plagiarism detection system was built, based on comparing sentences represented as graphs. The system included a pre-processing, candidate retrieval and detailed analysis phase, which applied the graph edit distance algorithm. By comparing the system against state-of-the-art systems, an empirical evaluation was made.

The algorithm achieved the highest scores with weighted edit costs. The edge cost function was best calculated based on deprel similarity. Dependency graphs were best represented by directed edges.

The approach would have scored 5th out of ten contestants in the PAN11 challenge, or 4th by only considering precision and recall. The algorithm has many areas of improvement, especially when it comes to parameter tuning. As a conclusion, it is reasonable to claim that the graph-based approach is competitive with state-of-the-art systems, but require some fine tuning. Especially the candidate retrieval and passage merging phase have potential for improvement.

7.1 Future Work

The following sections describe improvements to the system which were not implemented due to time constraints, and are likely to improve the plagiarism detection performance.

7.1.1 Synonym node matching

A common way to plagiarise text is to replace words with synonyms (Stamatatos, 2011). A plagiarism detection system should be able to determine the similarity between two words, or at least detect closely related synonyms. Gustafson et al. (2008) use a *word-correlation factor*, which determine how often two words correlate, based on 880,000 Wikipedia documents. By using a measure like *word-correlation factor*, each word the algorithm would be able to determine the relative similarity between two words. As a result, the algorithm would be better equipped to deal with cases where a plagiariser has replaced words in a sentence.

7.1.2 Edit cost weight tuning

The weights used for POS and edge label edit costs described in Section 4.4 were very simple. In order to achieve better edit cost, the weights can be tuned on a training corpus, such as PAN10.

By inserting random values for each weight, and comparing results, the weights can be tuned. Due to the size of the corpus, it is probably best to work with a subset of the data set.

7.1.3 Cross-lingual detection

Currently, the system only supports English, but this is mostly due to the models used for dependency parsing and POS-tagging. In order to apply the system to other languages, one simply has to replace the English models, with models for the given language.

As described in Section 2.3.1, the PAN11 corpus consists of English, German and Spanish documents. By performing automatic translation, the documents can be compared. Needless to say, this is a hard task, and outside the scope of this thesis. Due to the added complexity by automatic translation, the task of detecting similarity cross languages becomes difficult.

Another solution is to simply match POS-tags in the Graph Edit Distance algorithm, but given different syntactic structures in different languages, this is most likely an inaccurate solution as well.

7.1.4 Adjacent passage detection

Very few plagiarised passages in the PAN11 corpus consist of a single sentence. Most plagiarised passages are on paragraph level. Due to this, adjacent passages to a plagiarised passage should also be analysed, in case they were excluded in the candidate retrieval phase.

After detecting a plagiarised passage, the adjacent sentences can be analysed for plagiarism. Since plagiarism usually is done on paragraph level, it increases the likelihood that a sentence is plagiarised if it is adjacent to a plagiarised sentence. Due to this, the plagiarism threshold should be slightly higher for these adjacent sentences.

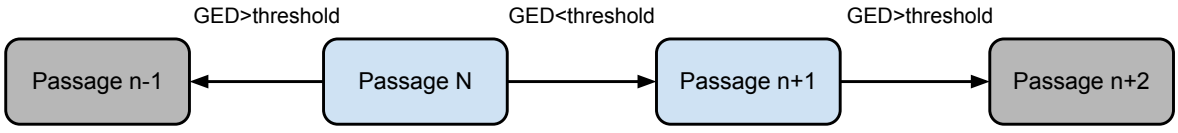


Figure 7.1: Adjacent passages analysis

Figure 7.1 illustrates how the expanding process can be done. Blue passages are added to the passage, grey passages are discarded. The search for adjacent sentences continues until a passage with too high graph edit distance is encountered, or the last/first sentence is reached.

Appendix A

A.1 List of Part-of-speech tags

The following is a list of part-of-speech tags from the Penn Treebank project.

CC	Coordinating conjunction	RB	Adverb
CD	Cardinal number	RBR	Adverb, comparative
DT	Determiner	RBS	Adverb, superlative
EX	Existential there	RP	Particle
FW	Foreign word	SYM	Symbol
IN	Preposition or subordinating conjunction	TO	to
JJ	Adjective	UH	Interjection
JJR	Adjective, comparative	VB	Verb, base form
JJS	Adjective, superlative	VBD	Verb, past tense
LS	List item marker	VBG	Verb, gerund or present participle
MD	Modal	VBN	Verb, past participle
NN	Noun, singular or mass	VBP	Verb, non-3rd person singular present
NNS	Noun, plural	VBZ	Verb, 3rd person singular present
NNP	Proper noun, singular	WDT	Wh-determiner
NNP	Proper noun, plural	WP	Wh-pronoun
PDT	Predeterminer	WP$	Possessive wh-pronoun
POS	Possessive ending	WRB	Wh-adverb
PRP	Personal pronoun	PRP$	Possessive pronoun

Table A.1: List of POS-tags

Bibliography

Alzahrani, S. and N. Salim (2010). Fuzzy semantic-based string similarity for extrinsic plagiarism detection - lab report for PAN at clef 2010. See Braschler et al. (2010).

Braschler, M., D. Harman, and E. Pianta (Eds.) (2010). *CLEF 2010 LABs and Workshops, Notebook Papers, 22-23 September 2010, Padua, Italy.*

Cooke, N., L. Gillam, P. Wrobel, H. Cooke, and F. Al-Obaidli (2011). A high-performance plagiarism detection system - notebook for PAN at clef 2011. See Petras et al. (2011).

Costa-Jussà, M. R., R. E. Banchs, J. Grivolla, and J. Codina (2010). Plagiarism detection using information retrieval and similarity measures based on image processing techniques - lab report for PAN at clef 2010. See Braschler et al. (2010).

De Marneffe, M. and C. Manning (2008). Stanford typed dependencies manual. *URL http://nlp. stanford. edu/software/dependencies_manual. pdf*.

Devi, S. L., P. R. K. Rao, R. V. S. Ram, and A. Akilandeswari (2010). External plagiarism detection - lab report for PAN at clef 2010. See Braschler et al. (2010).

Fankhauser, S., K. Riesen, and H. Bunke (2011). Speeding up graph edit distance computation through fast bipartite matching. *Graph-Based Representations in Pattern Recognition*, 102–111.

Forner, P., J. Karlgren, and C. Womser-Hacker (Eds.) (2012). *CLEF 2012 Evaluation Labs and Workshop, Online Working Notes, Rome, Italy, September 17-20, 2012.*

Gao, X., B. Xiao, D. Tao, and X. Li (2010). A survey of graph edit distance. *Pattern Analysis & Applications 13*(1), 113–129.

Ghosh, A., P. Bhaskar, S. Pal, and S. Bandyopadhyay (2011). Rule based plagiarism detection using information retrieval - notebook for PAN at clef 2011. See Petras et al. (2011).

Gillam, L., N. Newbold, and N. Cooke (2012). Educated guesses and equality judgments: Using search engines and pairwise match for external plagiarism detection. See Forner et al. (2012).

Gottron, T. (2010). External plagiarism detection based on standard ir technology and fast recognition of common subsequences - lab report for PAN at clef 2010. See Braschler et al. (2010).

Grman, J. and R. Ravas (2011). Improved implementation for finding text similarities in large sets of data - notebook for PAN at clef 2011. See Petras et al. (2011).

Grozea, C. (2012). Brainsignals submission to plant identification task at imageclef 2012. See Forner et al. (2012).

Grozea, C. and M. Popescu (2010). Encoplot - performance in the second international plagiarism detection challenge - lab report for PAN at clef 2010. See Braschler et al. (2010).

Grozea, C. and M. Popescu (2011). The encoplot similarity measure for automatic detection of plagiarism - notebook for PAN at clef 2011. See Petras et al. (2011).

Gupta, P., S. Rao, and P. Majumder (2010). External plagiarism detection: N-gram approach using named entity recognizer - lab report for PAN at clef 2010. See Braschler et al. (2010).

Gustafson, N., M. Pera, and Y. Ng (2008). Nowhere to hide: Finding plagiarized documents based on sentence similarity. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, Volume 1, pp. 690–696. IEEE.

Hu, X., T. Chiueh, and K. Shin (2009). Large-scale malware indexing using function-call graphs. In *Proceedings of the 16th ACM conference on Computer and communications security*, pp. 611–620. ACM.

Jayapal, A. (2012). Similarity overlap metric and greedy string tiling for plagiarism detection at pan 2012. See Forner et al. (2012).

Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation 28*(1), 11–21.

Kasprzak, J. and M. Brandejs (2010). Improving the reliability of the plagiarism detection system - lab report for PAN at clef 2010. See Braschler et al. (2010).

Kennedy, M. (2010, April). Mongodb vs. sql server 2008 performance showdown. `http://blog.michaelckennedy.net/2010/04/29/mongodb-vs-sql-server-2008-performance-showdown/`. Last visited: 27th November 2012.

Kong, L., H. Qi, S. Wang, C. Du, S. Wang, and Y. Han (2012). Approaches for candidate document retrieval and detailed comparison of plagiarism detection. See Forner et al. (2012).

Koppel, M., N. Akiva, and I. Dagan (2006). Feature instability as a criterion for selecting potential style markers. *Journal of the American Society for Information Science and Technology 57*(11), 1519–1525.

Küppers, R. and S. Conrad (2012). A set-based approach to plagiarism detection. See Forner et al. (2012).

Marcus, M., M. Marcinkiewicz, and B. Santorini (1993). Building a large annotated corpus of english: The penn treebank. *Computational linguistics 19*(2), 313–330.

Micol, D., Ó. Ferrández, F. Llopis, and R. Muñoz (2010). A textual-based similarity approach for efficient and scalable external plagiarism analysis - lab report for PAN at clef 2010. See Braschler et al. (2010).

Muhr, M., R. Kern, M. Zechner, and M. Granitzer (2010). External and intrinsic plagiarism detection using a cross-lingual retrieval and segmentation system - lab report for PAN at clef 2010. See Braschler et al. (2010).

Nawab, R. M. A., M. Stevenson, and P. Clough (2010). University of sheffield - lab report for PAN at clef 2010. See Braschler et al. (2010).

Nawab, R. M. A., M. Stevenson, and P. D. Clough (2011). External plagiarism detection using information retrieval and sequence alignment - notebook for PAN at clef 2011. See Petras et al. (2011).

Nedas, K. A. (2008, May). Munkres java implementation. `http://konstantinosnedas.com/dev/soft/munkres.htm`. Last visited: 8th December 2012.

Nivre, J. (2005). Dependency grammar and dependency parsing. Technical report, Technical Report MSI report 05133, Växjö University: School of Mathematics and Systems Engineering.

Nivre, J. and J. Hall (2010). A quick guide to maltparser optimization. Technical report, Citeseer.

Oberreuter, G., G. L'Huillier, S. A. Ríos, and J. D. Velásquez (2010). Fastdocode: Finding approximated segments of n-grams for document copy detection - lab report for PAN at clef 2010. See Braschler et al. (2010).

Oberreuter, G., G. L'Huillier, S. A. Ríos, and J. D. Velásquez (2011). Approaches for intrinsic and external plagiarism detection - notebook for PAN at clef 2011. See Petras et al. (2011).

Palkovskii, Y. and A. Belov (2012). Applying specific clusterization and fingerprint density distribution with genetic algorithm overall tuning in external plagiarism detection. See Forner et al. (2012).

Palkovskii, Y., A. Belov, and I. Muzika (2010). Exploring fingerprinting as external plagiarism detection method - lab report for PAN at clef 2010. See Braschler et al. (2010).

Palkovskii, Y., A. Belov, and I. Muzyka (2011). Using wordnet-based semantic similarity measurement in external plagiarism detection - notebook for PAN at clef 2011. See Petras et al. (2011).

Papineni, K., S. Roukos, T. Ward, and W. Zhu (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics.

Pereira, R. C., V. P. Moreira, and R. Galante (2010). Ufrgs@pan2010: Detecting external plagiarism - lab report for PAN at clef 2010. See Braschler et al. (2010).

Petras, V., P. Forner, and P. D. Clough (Eds.) (2011). *CLEF 2011 Labs and Workshop, Notebook Papers, 19-22 September 2011, Amsterdam, The Netherlands*.

Potthast, M., A. Barrón-Cedeño, A. Eiselt, B. Stein, and P. Rosso (2010). Overview of the 2nd international competition on plagiarism detection. *Notebook Papers of CLEF 10*.

Potthast, M., A. Eiselt, A. Barrón-Cedeno, B. Stein, and P. Rosso (2011). Overview of the 3rd international competition on plagiarism detection. In *Notebook Papers of CLEF 2011 LABs and Workshops*.

Potthast, M., T. Gollub, M. Hagen, J. Kiesel, M. Michel, A. Oberländer, M. Tippmann, A. Barrón-Cedeño, P. Gupta, P. Rosso, and B. Stein (2012). Overview of the 4th international competition on plagiarism detection. See Forner et al. (2012).

Potthast, M., B. Stein, A. Barrón-Cedeño, and P. Rosso (2010). An evaluation framework for plagiarism detection. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pp. 997–1005. Association for Computational Linguistics.

Rao, S., P. Gupta, K. Singhal, and P. Majumder (2011). External & intrinsic plagiarism detection: Vsm & discourse markers based approach - notebook for PAN at clef 2011. See Petras et al. (2011).

Riesen, K. and H. Bunke (2009). Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing 27*(7), 950–959.

Sánchez-Vega, F., M. M. y Gómez, and L. V. Pineda (2012). Optimized fuzzy text alignment for plagiarism detection. See Forner et al. (2012).

Stamatatos, E. (2011). Plagiarism detection using stopword n-grams. *Journal of the American Society for Information Science and Technology 62*(12), 2512–2527.

Suárez, P., J. C. G. Cristóbal, and J. Villena-Román (2010). A plagiarism detector for intrinsic plagiarism - lab report for PAN at clef 2010. See Braschler et al. (2010).

Suchomel, S., J. Kasprzak, and M. Brandejs (2012). Three way search engine queries with multi-feature document comparison for plagiarism detection. See Forner et al. (2012).

Torrejón, D. A. R. and J. M. M. Ramos (2010). Coremo system (contextual reference monotony) - lab report for PAN at clef 2010. See Braschler et al. (2010).

Torrejón, D. A. R. and J. M. M. Ramos (2011). Crosslingual coremo system (contextual reference monotony) - notebook for PAN at clef 2011. See Petras et al. (2011).

Torrejón, D. A. R. and J. M. M. Ramos (2012). Detailed comparison module in coremo 1.9 plagiarism detector. See Forner et al. (2012).

Toutanova, K., D. Klein, C. Manning, and Y. Singer (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pp. 173–180. Association for Computational Linguistics.

Vania, C. and M. Adriani (2010). Automatic external plagiarism detection using passage similarities - lab report for PAN at clef 2010. See Braschler et al. (2010).

Zeng, Z., A. Tung, J. Wang, J. Feng, and L. Zhou (2009). Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment 2*(1), 25–36.

Zou, D., W. jiang Long, and Z. Ling (2010). A cluster-based plagiarism detection method - lab report for PAN at clef 2010. See Braschler et al. (2010).