Towards Detecting Textual Plagiarism Using Machine Learning Methods

by

Rune Borge Kalleberg

Supervisors

Associate Professor, Morten Goodwin, Ph.D.

Professor, Ole-Christoffer Granmo, Ph.D.

This master's thesis is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education. However, this does not imply that the University answers for the methods that are used or the conclusions that are drawn.

The University of Agder

Faculty of Engineering and Science

Department of Information and Communication Technology

Grimstad, May 23, 2015

Abstract

Textual plagiarism is passing off someone else's text as your own. The current state of the art in plagiarism detection performs well, but often uses a series of manually determined thresholds of metrics in order to determine whether an author is guilty of performing plagiarism or not. These thresholds are optimized for a single data set and are not optimal for all situations or forms of plagiarism. The detection methodologies also require a professional familiar with the algorithms in order to be properly adjusted, due to their complexity. Using a pre-classified data set, machine learning methods allow teachers and censors without knowledge of the methodology to use a plagiarism detection tool specifically designed for their needs.

This thesis demonstrates that a methodology using machine learning, without the need to set thresholds, can match, and in some cases surpass, the top methodologies in the current state of the art. With more work, future methodologies may possibly outperform both the best commercial and freely available methodologies.

Acknowledgements

I would like to take this opportunity to thank my supervisors associate professor Morten Goodwin, and professor Ole-Christoffer Granmo for invaluable advice and guidance when I got stuck technically or had questions about thesis writing. I would also like to thank supporting supervisor assistant professor Christian Auby for rewarding meetings in earlier projects and discussion of ideas leading up to this thesis.

In times when motivation has been greatly lacking, the wonderful world of progressive rock music, especially the works of *Transatlantic*(which has been used for a few examples in this thesis) and *Spock's Beard*(Thanks Marcus!) have pulled me through.

I'd also like to thank my wonderful girlfriend Linn for keeping me level and motivated these last few weeks, and for her grammatical first-aid. I can't wait to see what the future brings!

Last but certainly not least I'd like to thank my wonderful parents Laila Johanne Borge and Ragnar Braar Kalleberg for their never-ending support when needed the most as well as their limitless faith in me. Without your invaluable support and fantastic parenting I would not be who, or where I am today! You are the best parents any kid could hope for! Dad: Don't trust the salesman, and always read the manual!

Now, witness the power of this fully operational battle sta.... I mean thesis!

Preface

This Master's thesis is submitted in partial fulfillment of the requirements for the degree Master of Science in Information and Communication Technology at the University of Agder, Faculty of Engineering and Science.

This work was carried out under the supervision of associate professor Morten Goodwin and professor Ole-Christoffer Granmo at the University of Agder, Norway.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the problem domain of automated plagiarism detection and the motivation for this thesis. Section 1 presents the problem at hand and the motivation for this thesis. Section 2 presents the thesis definition and what was done. Section 3 contains the claims made by the discoveries and results of this thesis. Section 4 present the contributions made to the relevant fields and the state of the art. Section 5 explains who this thesis is meant for and what prerequisites a reader should have before continuing. Section 6 provides an outline of this thesis.

1.1 Background and Motivation

Plagiarism means taking the work or ideas of someone else and passing them off as your own. The most common and well known form being textual plagiarism. For the purpose of this thesis, all references to plagiarism will be to textual plagiarism; copying the text from a source text and presenting it as your own work. Plagiarism comes in many forms. One can directly copy a text, but detecting pure verbatim plagiarism is a fairly easy task, and plagiarists are quickly caught doing this with current tools. In order to mask an act of plagiarism, the text is often

rewritten, words in a sentence rearranged, replaced with synonyms, or the text may be summarized. This makes it harder for automated systems to detect the plagiarised text. Detecting semantic meaning in a text is especially challenging to do with a computer algorithm. They are however very adept at lexical analysis. Most plagiarism detection tools use the structural and lexical similarities of documents.

In a world where the Internet has become a part of every day life and an integral part of education with its massive amount of freely available information, it has become easier than ever to steal text and pass it as your own without being detected. Students seldom use books when researching and writing text. It is far more common to use the Internet for these purposes. Plagiarists can simply copy the text from some unknown source from the Internet to save themselves hours of work, and in many cases this goes undetected.

Detecting textual plagiarism is a tedious and almost impossible process when done manually. With millions of possible sources online, this entire process benefits from being automated. A long line of commercial, automatic plagiarism detection tools have been made available to both academic institutions and individuals. The inner workings of these are seldom public due to the commercial nature of the products.

This is an active field of research, but the accuracy of these tools falters as the plagiarism becomes more complex. Verbatim copies are easy to detect, but as words are replaced with synonyms and shuffled around, detecting plagiarism becomes harder. Many different approaches have been designed attempting to handle these challenges, but few have yet to reach a very high level of accuracy in complex plagiarism methods.

Different levels in the education system require different amounts of your own work in order to be legitimate hand-ins. Children in high school or lower grades could potentially have more lenient plagiarism detection systems. These could allow for summaries and simple rewrites of passages as long as it is shown that

some work has been done on the text, and it is not a simple verbatim copy, their text should be passable as their own. Some classes may focus on purely the research and not the writing, while others may be vice versa, and should be very strict. In higher levels of education even summaries are often not acceptable, but a level of verbatim copying should perhaps be without triggering an alarm to allow for quotes. Finding the perfect settings for each of these scenarios is tedious and seldom produce optimal results. Tuning it also requires knowledge of the algorithm, which few users have.

Machine learning allows users to find optimal settings automatically based on statistics from a data set made up of pre-classified plagiarism and non-plagiarism cases. By defining passages, documents or sentences in a data set as plagiarism, or even the kind of plagiarism methods used on each passage, a system could potentially be tailored to each institution , or even teacher preference. An examiner or teacher can sit down with a data set of text to classify themselves what they consider to be examples of plagiarism and end up with a data set which can be used to train a detector for only the classes they teach.

These methods are not commonly seen in the plagiarism detection fields of research. As the review of the state of the art will show, many rely on thresholds tailored manually to a specific data set and will not work as well in real settings with different data. These are hard to adjust properly for anyone not familiar with the science behind the methodologies. By allowing users to say what they would label as plagiarism and what isn't, anyone could end up with their own tailored detector.

Creating a data set for the purpose of this Master's thesis would take far too long. However, a fairly suitable data set already exists allowing the methodology to be compared to the current state of the art.

In the last couple of years a workshop known as *PAN Workshop and Competition: Uncovering Plagiarism, Authorship and Social Software Misuse*(PAN)[1]

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

has held competitions in plagiarism detection and other related fields of research. They benchmark multiple contestants against each other using a standardised data set. This provides a good understanding of the performance level of the current state of the art. The entrants also provide papers explaining their methods and presenting their results. Overview papers are also provided. The published papers for the PAN workshop are all good representatives of the current state of the art. A freely available data set allows any new method to be easily benchmarked against the leading plagiarism detection techniques.

The literature shows examples of very good ideas designed mostly by academics in the field of linguistics. It is, however, apparent that many of these papers do not explore the possibilities opened up by machine learning as only a few make use of any form of machine learning methods. This is somewhat strange as the field has adopted methodologies from bio informatics. More specifically gene sequencing. Bioinformatics is a field that has embraced machine learning far more[3].

Many approaches make use of empirically found thresholds and only a couple of self-designed features for each approach. This can make each tool very specialized and give very good performance for a single or a few types of plagiarism in researched data sets, but they are not versatile enough to be applied to a real world scenario. Machine learning methods allow for joining several of these state-of-the-art approaches and for creating novel features based on existing ideas.

In contrast to plagiarism detection, the neighbouring research field of author profiling has adopted machine learning to a far greater extent. All entrants in the author profiling competition in the PAN 2014 workshop made use of a wide range of machine learning algorithms[4].

This thesis shows that a machine learning system that makes use of ideas from several of the top state of the art methodologies as well as some novel methods based on these methodologies is able to closely match the performance of the simpler methods in the current state of the art without the need for parameter

tweaking for each new data set.

1.2 Thesis Definition

In this thesis I will join techniques from some of the best approaches in the state of the art in plagiarism detection. These will be implemented as features in a machine learning pipeline. Alongside these, novel techniques based on their ideas will also be used as features or processing techniques.

The thesis research assumes that a given set of potential source documents is already available. Source retrieval is not a part of this research project but is an active field of research and an important part of any future research for this project.

Each part of the thesis methodology will be tested and all classification algorithm candidates compared to each other in an attempt to expose strengths and weaknesses throughout the pipeline. The feature set and classification algorithms producing the best overall result when trained and tested on a freely available plagiarism corpus will be chosen for implementation in a final system which will be benchmarked against the PAN contestants from 2012-2014 using their supplied data set.

The benchmarking results will confirm or disprove the following hypothesis:

"*Machine learning algorithms are capable of similar or better performance in terms of plagiarism detection than the current, manually optimized, state of the art methodologies in PAN 2012-2014*"

1.3 Claims

In this thesis I show that the current state of the art methods can be successfully combined using machine learning for a more powerful and versatile plagiarism detection tool. I also demonstrate that features can be constructed from similar fields of research and that these can aid in classifying plagiarism.

This method can take any suitable, pre-classified data set and produce results matching manually optimized state of the art methodologies without the need for threshold tailoring. The thesis shows that this method matches, and in some cases exceeds, the performance of the current state of the art in both simple and more complex forms of plagiarism.

1.4 Contribution

This thesis evaluates the use of some common machine learning techniques for use with plagiarism detection on a freely available data set to allow for accurate benchmarking against the current state of the art.

The work makes use of common techniques and joins them as features in a plagiarism detector. Some new ideas based on these techniques are also added. The machine learning algorithms used will give an indication as to which features and methodologies are most useful when classifying plagiarism or different types of plagiarism.

This thesis demonstrates that machine learning has a potential in plagiarism detection as it is capable of detecting more complex forms of plagiarism and match the current state of the art in terms of performance. Hopefully this will motivate future projects to further explore the possibilities of machine learning in the field of plagiarism detection.

1.5 Target Audience

The target audience for this thesis is anyone interested in either plagiarism detection or machine learning. Some knowledge of the aforementioned fields is required when reading this thesis, though most terms and techniques are either explained or an explanation is cited allowing anyone with a background in computer science to read and understand the contents. Basic understanding of probability theory and commonly used performance metrics is highly recommended.

1.6 Report Outline

Chapter 1 introduces the problem and provides a thesis definition as well as an explanation as to what this thesis adds to the field of plagiarism detection. It also explains who this thesis is meant for and what prerequisites are recommended prior to reading it. Chapter 2 explains the current state of the art and discusses the methodologies of the leading entrants in a conference workshop that focuses on plagiarism detection. Chapter 3 explains the methodology developed for this thesis. Chapter 4 presents the results from the methodology and compares it to entrants of the conference. Chapter 5 presents the conclusion for this thesis and suggests what should be done in the future both for this thesis and in the field of plagiarism detection in general.

Chapter 2

State of the Art

This chapter presents the current state of the art within plagiarism detection. It focuses mainly on entrants to a conference workshop including plagiarism detection as one of the tasks. The reasoning behind this focus was a freely available data set to allow for benchmarking and comparison.

Section 1 introduces the conference workshop. More specifically: the relevant subtask for this thesis. It also provides an overview of its entrants as well as discussing the data set used. Section 2 explains in detail many of the methodologies implemented by the leading entrants of the workshop.

There has been a fair bit of previous research on the topic of plagiarism detection and the surrounding areas of interest. Plagiarism detection, more specifically: *text alignment*, is a field of research where very few methodologies make use of any machine learning techniques. The main motivation for this thesis is to discover if machine learning techniques may be beneficial to accurately detect cases of plagiarism on a similar level to the current state of the art. Algorithms developed by these scientists use very promising methodologies, but often find the detector thresholds through empirical research. This may lead to sub-optimal thresholds when used with other data sets and real life data. They are also often

limited in the number of methods used to detect plagiarism. A few new ideas for each paper are tested and return fairly good, but at times specialized results.

2.1 PAN Workshop and Competition

The *PAN Workshop and Competition: Uncovering Plagiarism, Authorship and Social Software Misuse*(PAN)[1] proved to be a large source of information on the current state of the art in multiple branches of text analysis, including text alignment. This workshop provides a series of tasks and competitions covering text alignment, author profiling, and author identification. All of which may hold interesting methodologies that could be used in a machine learning based plagiarism detection tool. The workshop also provides freely available data sets in order to benchmark the methodology developed in this thesis.

None of the entrants in the text alignment task make use of machine learning with the exception of some clustering. However, all of the entrants in the author profiling task in 2014 utilized machine learning[4]. This may be due to a more suitable data set and very different methodologies.

2.1.1 Text Alignment

Text alignment is one of the tasks in the PAN contest. PAN defines this task as:

> In text alignment, given a pair of documents, the task is to identify all contiguous passages of reused text between them[2].

This pair consists of a potential source and a suspicious document. The source document contains the text that may have been plagiarised while the suspicious document is the one that will be scanned for acts of plagiarism.

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

Text alignment means that not only should plagiarism be classified on a document level, but that each passage of plagiarised text should also be located in both the source, and the suspicious document. In a use-case setting this would ease the task of manually confirming or discarding a document pair that has been classified as plagiarism, as the passages could be highlighted for easy comparison. Due to the current accuracy level of the state of the art, a user should not trust the detection system without doing a manual verification of the candidate passages.

Alongside the text alignment task was also a source retrieval task where the goal is to get all sources from a cached database of websites from which text has been reused in a suspicious document. This would be a first step in a plagiarism detection tool when there is a possibility that text from the Internet has been reused. No focus has been given to this task in this thesis as it is not directly related to the problem of text alignment, which is the main focus of this thesis. It is worth noting that more research is needed in this field as even the best results on an even smaller, cached database(Webis-TRC-2012 and 2013) still yield an F_1 score lower than 0.5[2].

The task overview papers and the papers on the entrants in this task was the main source of information when researching plagiarism detection. Many of these methodologies have influenced the approach in this thesis and provided ideas for improvement. The idea behind this research approach was to pick the best ideas from the best methodologies, extract features from these metrics and methodologies and apply a machine learning algorithm to them. Additional metrics would also be added to further aid the algorithm in discovering a potential pattern.

The text alignment subtask in PAN provides a varied corpus that tests the entrants in detection of " . . . *verbatim copies and quotations to paraphrases and translations to summaries*" [2].

2.1.2 Data Sets

It is important to discuss the data sets were were used in this thesis when benchmarking it against other contestants. It proved a valuable tool and also posed a series of challenges.

PAN makes use of a single large corpus that is withheld from the public. The first year the corpus was used a subset of this corpus was released as training data prior to the workshop. The entrants were benchmarked against each other with a smaller, withheld test set drawn from the large corpus. After the workshop the test data was released publically and could be used for future training. Each year PAN releases a smaller data set that is used for testing. The corpus was first used in 2012, with the latest conference being held in 2014, meaning all entrants since 2012 have been benchmarked using this data set. In order to keep the older entrants benchmarked against the newest entrants, all entrants since 2012 are tested with the latest test set each year.

The corpus is based on Webis-TRC-13[5]. Plagiarised passages are generated by creating a repository of source and suspicious documents from Webis-TRC-13. Document pairs are then generated. The suspicious documents from each of these pairs may receive one or more automatically generated plagiarised passages from the source document inserted into them. These passages use one of four different obfuscation strategies.

No obfuscation This is the simplest form of plagiarism. Verbatim copy without any alterations.

Random obfuscation This method shuffles both words and sentences around within the borders of a passage. This is done automatically and is not in any way guaranteed to be grammatically or semantically correct after obfuscation.

Translation-chain This method sends a passage through a series of translations

using various online translation tools such as Google Translate, Bing Translator, or other similar tools. All texts in the corpus are English. They are passed through these translators applying a series of translations to different languages before returning to English. The purpose of this is to attempt to preserve the semantic meaning while rewriting the text.

Summary This is a summary written from a part of the source document.

Comparing the documents in the data sets is not as simple as $D_{comp} = D_{plag} \times D_{src}$. A *pairs* file lists which documents that should be compared against each other. The reason for this is that many of the source and suspicious documents are generated from one another to increase the size of the data set without having to create new text. When doing exhaustive comparison of all document pairs this would yield a lot of false positives, which, is in fact, plagiarism. It could be said that the data set even plagiarises itself.

Figure 2.1 shows how plagiarism is described for a document pair. An XML file named after the document pair it is describing contains plagiarism data. All pairs have an XML file containing the `document` tag, denoting the suspicious document it describes. In the event of plagiarism, a `feature` tag describes the plagiarism locations and document, as well as type and in some cases the level of obfuscation. Figure 2.1 shows a plagiarism case with no obfuscation. The plagiarised passage is 579 characters long and starts at character 6765 in the suspicious document and at character 7259 in the source document.

```
<document reference="suspicious-document00016.txt">
    <feature name="plagiarism" obfuscation="none"
    source_length="579" source_offset="7259"
    source_reference="source-document01836.txt"
    this_length="579" this_offset="6765"
    type="artificial" />
</document>
```

Figure 2.1: XML example from PAN data set (suspicious-document00016-source-document01836.xml).

2.1.3 Results and Comparisons

Before discussing the methodologies in detail, an overview of how they are evaluated, as well as their performance should aid in understanding their strengths and weaknesses individually.

PAN uses a custom performance metric, PlagDet, which balances recall, precision, and *granularity*. The granularity metric is designed to measure any overlaps and multiple detections of cases. A case is classified correctly if over half of the detected characters are within a plagiarised passage, and over half of the plagiarised passage has been detected. The formal definitions are shown and explained in detail in section 3.7.

Let S denote the set of plagiarism cases in the corpus. R denotes the set of plagiarism detections by a detector. Granularity is defined as:

$$gran(S, R) = \frac{1}{S_R} \sum_{s \in S_R} |R_S|$$

$S_R \subseteq S$ are detected cases in R while $R_S \subseteq$ R are detections in S

Along with recall and precision, these metrics make up the performance score designed for PAN. PlagDet is defined as:

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

$F_1(S, R)$ is the harmonic mean of precision and recall and is defined as:

$$F_1(S, R) = 2\frac{Prec(S, R) \cdot Rec(S, R)}{Prec(S, R) + Rec(S, R)}$$

The exact formulas for precision and recall are described later in section 3.7 and in further detail [6, 2]. They do not differ from normal precision and recall measures, but are designed to calculate these measures on a character level.

PlagDet can be considered an F_1 measure on a character level, which also considers granularity.

In general, the top entrants have a very solid performance on simpler forms of plagiarism such as *no obfuscation* and *random obfuscation*. They are found to struggle more with *translation chain* and *summary*.

		Obfuscation Strategies				
Submission	Year	None	Random	Translation	Summary	Overall Corpus
Sanchez-Perez	2014	0.900	0.884	0.887	0.561	0.878
Oberreuter	2014	0.920	0.868	0.881	0.368	0.869
Palkovskii	2014	0.960	0.865	0.858	0.276	0.868
Glinos	2014	0.962	0.806	0.847	0.624	0.859
Shresta	2014	0.892	0.866	0.844	0.156	0.844
Kong	2012	0.872	0.832	0.852	0.436	0.837
Torrejón	2014	0.932	0.754	0.859	0.353	0.830
Oberreuter	2012	0.942	0.750	0.846	0.132	0.827
Gross	2014	0.900	0.803	0.838	0.319	0.826
Torrejón	2013	0.926	0.747	0.851	0.341	0.822
Kong	2014	0.838	0.823	0.852	0.431	0.822
Kong	2013	0.827	0.823	0.852	0.434	0.819
Palkovskii	2012	0.882	0.797	0.740	0.275	0.792
Torrejón	2012	0.882	0.702	0.801	0.442	0.788
Suchomel	2013	0.818	0.753	0.675	0.610	0.745
Suchomel	2012	0.898	0.652	0.631	0.501	0.732

Table 2.1: Top 16 out of 29 2012-2014 PAN entrants with respects to, and sorted by, PlagDet [2]

As visible in table 2.1, the entrants are very adept at detecting the simpler forms of plagiarism, but struggle greatly with the *summary* type. Especially the verbatim, *no obfuscation* form of plagiarism seems to be easy to detect. Even further down the extended PlagDet table in the PAN overview paper [2], most of the entrants have a PlagDet score over 0.80. This form of plagiarism is the easiest to detect. With a PlagDet score of 0.88 overall, Sanchez-Perez holds a small lead over the other entrants. What is interesting is that this methodology is not the best on verbatim plagiarism. It is, however, quite adept overall and is second best in *summary* out of the top 10 entrants and third best overall with 0.56, following

Glinos 2014 at 0.62 and Suchomel 2013 at 0.61. This suggests that this method holds some features that work very well across all kinds of plagiarism.

Glinos 2014 is the best in the entire field in terms of both *summary* and *no obfuscation*, but struggles somewhat in both *random* and *translation*. The methodology here may definitely be of interest to enhance any performance of a detector in both of Glinos' strong obfuscation strategies. Studying why it struggles in the weak strategies may provide valuable insight which could lead to a stronger detector.

Palkovskii 2014 also has a high performance level in the 3 simplest forms of plagiarism but struggles in *summary*. The same can be said for Shresta 2014 and Oberreuter 2014 who both greatly improved their previous methodology entries. It is interesting is that Sanchez-Perez leads the state of the art as a first time entrant in PAN.

		Obfuscation Strategies				
Submission	Year	None	Random	Translation	Summary	Overall Corpus
Sanchez-Perez	2014	0.979	0.861	0.890	0.413	0.879
Oberreuter	2014	0.999	0.833	0.863	0.245	0.858
Shresta	2014	0.974	0.832	0.853	0.089	0.838
Palkovskii	2014	0.964	0.822	0.820	0.177	0.826
Kong	2012	0.948	0.780	0.850	0.299	0.824
Kong	2013	0.907	0.787	0.846	0.300	0.843
Kong	2014	0.895	0.781	0.845	0.296	0.807
Glinos	2014	0.960	0.725	0.762	0.486	0.793
Saremi	2013	0.954	0.689	0.804	0.102	0.771
Torrejón	2014	0.967	0.630	0.821	0.231	0.769
Oberreuter	2012	0.999	0.653	0.796	0.071	0.769
Gross	2014	0.907	0.719	0.784	0.206	0.766
Suchomel	2013	0.996	0.689	0.666	0.563	0.766
Torrejón	2013	0.953	0.634	0.811	0.216	0.762
Palkovskii	2012	0.994	0.751	0.667	0.169	0.762
Torrejón	2012	0.964	0.623	0.791	0.290	0.753

Table 2.2: Top 16 out of 29 2012-2014 PAN entrants with respects to, and sorted by, recall [2]

Table 2.2 shows that all entrants have a very high recall in *no obfuscation* plagiarism. This continues on beyond this table, with only 3 entrants below 0.8[2]. The *random* and *translation* obfuscation strategies are often quite similar to each other in terms of recall, but everyone seems to struggle greatly with summary, with the greatest recall is Suchomel as far down as 0.56. This number is interesting even though Suchomel struggles with the other obfuscation strategies compared to the other entrants.

		Obfuscation Strategies				
Submission	Year	None	Random	Translation	Summary	Overall Corpus
Glinos	2014	0.964	0.970	0.962	0.965	0.963
Nourian	2013	0.929	0.963	0.959	0.999	0.947
Jayapal	2012	0.985	0.960	0.896	0.833	0.945
Alvi	2014	0.919	0.948	0.960	0.880	0.934
Gross	2014	0.918	0.960	0.921	0.949	0.933
Palkovskii	2014	0.956	0.915	0.899	0.913	0.922
Torrejón	2014	0.899	0.938	0.900	0.898	0.904
Torrejón	2013	0.900	0.910	0.895	0.908	0.895
Oberreuter	2012	0.890	0.879	0.903	0.990	0.894
Gilliam	2014	0.881	0.952	1.000	0.000	0.886
Oberreuter	2014	0.852	0.906	0.900	0.936	0.886
Gilliam	2012	0.881	0.956	0.972	0.996	0.885
Gilliam	2013	0.881	0.960	0.973	0.996	0.885
Sanchez-Perez	2014	0.834	0.910	0.885	0.999	0.882
Jayapal	2013	0.920	0.923	0.857	0.688	0.879
Shresta	2013	0.809	0.923	0.880	0.905	0.875

Table 2.3: Top 16 out of 29 2012-2014 PAN entrants with respects to, and sorted by, precision [2]

Table 2.3 shows that in terms of precision, all of the entrants are mostly ranged from 0.8-1.0, with Glinos 2014 as overall best with 0.96 for all obfuscation strategies. Note that some entrants in table 2.3 were previously not visible in table 2.2 or 2.1. This is because of their very low recall. Jayapal 2013 has a corpus recall as low as 0.38 as well as a granularity of 2.9, resulting in a very poor PlagDet score of 0.271. Alvi has an overall PlagDet score of 0.65. High precision is useful to avoid insults to those falsely accused of plagiarism, but if very few plagiarists are caught, the algorithm provides little value. PlagDet is effectively the harmonic mean between precision and recall, while also considering granularity.

			Case Level			Document Level		
Entrant	Year	PlagDet	Prec	Rec	F_1	Prec	Rec	F_1
Sanchez-Perez	2014	0.88	0.90	0.91	0.90	0.92	0.91	0.91
Oberreuter	2014	0.87	0.84	0.89	0.87	0.89	0.89	0.89
Palkovskii	2014	0.87	0.90	0.85	0.87	0.90	0.84	0.87
Glinos	2014	0.86	0.90	0.83	0.87	0.93	0.88	0.91
Kong	2012	0.84	0.86	0.85	0.85	0.89	0.85	0.87
Shresta	2014	0.84	0.91	0.85	0.88	0.94	0.85	0.89
Gross	2013	0.83	0.90	0.86	0.88	0.93	0.85	0.89
Oberreuter	2012	0.83	0.81	0.79	0.80	0.83	0.80	0.81
Torrejón	2014	0.83	0.84	0.83	0.83	0.89	0.84	0.86
Torrejón	2013	0.83	0.83	0.83	0.83	0.87	0.84	0.85
Kong	2013	0.82	0.85	0.86	0.85	0.89	0.86	0.87
Kong	2014	0.82	0.86	0.85	0.85	0.89	0.85	0.87
Palkovskii	2012	0.79	0.80	0.80	0.80	0.82	0.80	0.81
Torrejón	2012	0.79	0.65	0.79	0.72	0.65	0.78	0.71
Suchomel	2013	0.74	0.66	0.83	0.73	0.67	0.82	0.74
Suchomel	2012	0.73	0.76	0.70	0.73	0.77	0.69	0.73

Table 2.4: Top 16 out of 29 2012-2014 PAN entrants with respects to case and document performance, sorted by PlagDet [2]

Due to the high character precision seen in table 2.3, the entrants have a very close link between their PlagDet score, and case/document F_1 scores, as seen in table 2.4. The case level is with most entrants equal or higher than their PlagDet score.

2.2 Plagiarism detection

Below follows a survey of the best performing and promising methodologies in the current state of the art, all of which are entrants from the PAN Text Alignment task from 2012 to 2014. These years were picked because they were all tested on the same data set in the 2014 competition. This corpus was made freely available and was used to benchmark the methodology in this thesis against the PAN entrants. The entrants mainly use a 3-step approach borrowed from the field of bioinformatics when doing *sequence alignment*[7, 2]. These steps form the basis for most methodologies.

2.2.1 Pre-processing

Before detection and the 3-step process begins, the texts are often pre-processed in some way in order to increase the chances of detecting similarities. The most common is simply converting the text to lower case. One reason for this is that if words within a sentence are shifted around, the first word in a sentence that is capitalized will not match if it appears in the middle of a sentence, and isn't a capitalized noun. The same holds true for a word that has been moved to the first word position in a sentence. Lowering the case allows both these words to match if they exist in the original and plagiarised sentence. However, some entrants attempt to detect named entities and must therefore also use text where the original text has not been converted to lower case[8].

Sanchez-Perez uses two forms of pre-processing prior to the seeding stage[1]. All words are converted to lower case, and stemming is applied. Stemming is the process of removing endings from words in order to reduce variations of a word to a single unambiguous term: the stem. The purpose of using a stemmer in plagiarism detection is to increase the possibility of detecting rewrites of a given word. Words like "*fish*", "*fishing*", "*fisher*", "*fishes*" will all be reduced to the

stem of the word: "*fish*".

Using a simple set of rules based on the language of the text that is to be stemmed, it strips all words where applicable. The simplest stemming rule in English is perhaps the common plural suffix "*-s*". Some words are not stripped of a suffix but are altered to transition to the root form of the word. "*Conditional*" will be rewritten to "*condition*" but "*relational*" will be rewritten "*relate*". The latter example has the suffix "*-ational*" while the former simply has "*-tional*", leading to different rules although the end of the suffixes are similar.

There are a range of different stemmers using different rules designed specifically for different languages. Sanchez-Perez uses a Python port of the Porter-stemmer algorithm[9] designed for English, as the entire data set is written in this language. This pre-processing technique is also seen in other entrants [10, 11] but which stemmer that has been applied has not been specified.

Some entrants also remove stop words[10, 12, 11] while others use the stop words specifically in part of their classification methodology[8]. Stop words are short *filler* words that add color and meaning semantically, but often add noise in natural language processing. Examples in English include, but are not limited to: *the, in, at, that, which*. No common stop word dictionary exists and is implementation specific.

Tokenization is the process of splitting a text up into certain some predetermined chunks, hereby referred to as tokens, fragments, or seed candidates. The entrants used a wide range of tokenization methods ranging from sentences to n-grams. This is tightly connected to the first step in the detection process and the line between the two is often blurry in some methodologies. Sanchez-Perez splits their seeds into sentences using the Natural Language Toolkit for Python, which has a tokenizer for english(usable as *tokenizers/punkt/english.pickle*). Kong also uses sentences when tokenizing[10, 12].

Sanchez-Perez also joined small sentences(3 words or shorter) with the next

sentence when generating seeds. The reasoning behind this was not shown, but it may have been an attempt to remove false positives in the *seeding* step when sentence similarity is measured. The statistical chance of a false similarity match increases as the sentences get shorter. Sanchez-Perez has released the source code for their methodology, thus allowing a finer study in the output of their algorithm. Studying the tokenization algorithm and comparing to the correct plagiarism cases in the data set, it seems that in some cases, the passages are too long; The splitting algorithm doesn't split up the fragments enough. This apparent minor overestimation of seed size seems to reduce the precision of the plagiarism detector somewhat. This may be because sentences are usually long enough to withstand the added three words and thus can still be classified as seeds. The similarity reduction that comes from the adding of short sentences is usually not big enough to get below the similarity metric threshold. However, if the similarity is low to begin with such as in translation or summary type plagiarism, the entire seed may be dropped and the recall will be affected.

Palkovskii also makes use of sentence splitting, although the details of their custom sentence splitting are not published[8].

```
Catch your breath as you watch your step.
```
(a) Phrase from [13]

```
catch your breath as you watch your step
```
(b) Converting to lower case and removing punctuation

```
catch your breath, your breath as, breath as you, as you watch
you watch your, watch your step
```
(c) 3-grams of the phrase, split by comma

Figure 2.2: Word 3-gram example

Palkovskii does, however, mainly make use of a different kind of token. N-

grams are chunks of size N. N can represent characters or words or any other means of splitting the text. Figure 2.2 shows a word 3-gram example where the sentence is split into all sequential phrases of 3 words. Entrants in PAN use both simple character and word n-grams of various N sizes as well as named entity n-grams, stop word n-grams, frequent word n-grams, odd even skip n-grams and contextual n-grams. When doing character n-grams spaces are often trimmed before generating the fragments. The text in figure 2.2 would be reduced to "*catchyourbreathasyouwatchyourstep*" before extracting all sequences N characters long. Given $N = 5$ the set (catch, atchy,tchyo,chyou,hyour, ...) would be generated. N varies greatly between methodologies and is an important setting when adjusting for optimal performance. Increasing the length of N will lower false positives, but also lower the recall when plagiarism becomes more complex. Lowering the length will increase overall recall but may include a lot of false positives, thus lowering precision. The optimal N will likely differ between plagiarism types. Palkovskii makes use of a wide range of n-gram forms in their detector [8]. Which N values that were used by Palkovskii is not stated.

The promising Glinos methodology uses word 1-grams as well as a frequent word supplement[14]. The tokenization is closely matched to the pre-processing. Besides a conversion to lower case, little is done to the text. Splitting is done using a custom tokenizer which preserves punctuation in numerical values and does not alter the possessive suffix *'s* nor the *n't* contraction. It separates all words, numbers, special characters and textual punctuation[14]. The lack of stemming may explain the precision of 0.96 on all forms of plagiarism[2], but may have affected the recall to some extent. One of their seeding methods excludes certain stop words when building the seeding model.

The use of n-gram tokens in Palkovskii may explain the very good PlagDet score in the simpler plagiarism forms, in particular the high *no obfuscation* score , as well as the low *summary* score. In the simpler forms of plagiarism, phrases and words may appear in the same order as they do in the source text. This is obviously the case in verbatim plagiarism, but also in the other two simple forms.

Consider the text used in figure 2.2: "*Catch your breath as you watch your step*" could for example be rewritten to "*Watch your step while you catch your breath*". In the event of a regular word 3-gram model, "catch your breath", and "watch your step" are common 3-grams between the two sentences. Passing this phrase through Google Translate in the chain English $\rightarrow$ Spanish $\rightarrow$ French $\rightarrow$ English resulted in the following phrase: "Breath while watching your step". Work is still needed in the world of automated translation as seen here, but some of the semantic meaning still remains. With word stemming *watching* would be reduced to *watch*, causing the 3-gram "*watch your step*" to be common with the original phrase. Some stemmers also reduce *your* to *you*, but the stemming is in this case irrelevant as the words are identical in both phrases.

Several of the other leading methodologies use skip-n grams, such as Palkovskii, Gross and Torrejón[8, 15, 16]. Skip-grams are most likely used in these methodologies to solve the more complex forms of plagiarism. These work by simply skipping over a given length of words. The phrase "*Catch your breath as you watch your step*" can be used to explain this concept. Skip-grams use two parameters. How many words to skip, and the regular N in length of the gram to be produced. Skip-1 3-grams for the mentioned phrase would look like this: (catch, breath, you), (your, as, watch), (breath, you,step). This would allow for some words to be shifted around and still detect seeds.

It is worth noting that Torrejón uses a special version of skip-grams they call *surrounding context n-grams*. The algorithm generates n-grams within sentences and shifts them around to create various permutations based on a set of rules[11]. This should increase the chance of picking up similarities within rewritten sentences and specifically target the *random* and *translation* obfuscation strategies. With a PlagDet score of approximately 0.75 for *random* and 0.86 for *translation* in their 2013 and 2014 entries, there are algorithms that produce better results specifically for these obfuscation strategies[2].

The same type of skip-grams were also seen in Suchomel [17], which bases

their contextual n-grams on CoReMo 1.9, the 2012 Torrejón entry[18], along with several other common types, such as 4-word n-grams and stop word 8-grams.

2.2.2 Seeding

Seeding is the process of detecting short plagiarism candidates. This can be as short as a couple of words or even a given number of characters up to a sentence, depending on the tokenization described in the previous section. Prior to seed classification the these candidates are referred to as fragments, passages, or seed candidates. After classification they are referred to as seeds.

Sanchez-Perez generates Bag-of-Words models(BoW) and calculate similarity metrics between these models. A BoW-model is a dictionary giving the number of occurrences of a word in a given sentence. Consider the example sentence: "*Day after day, so many fall away*." This produces the BoW-model seen in figure 2.3. The word *day* is the only word occurring more than once.

day	after	so	many	fall	away
2	1	1	1	1	1

Figure 2.3: Bag-of-Words example

BoW models are created for both the suspicious and source document. Seeding is then performed by doing an exhaustive comparison of similarity between all pairs of fragments in a suspicious, and a source document. This similarity is calculated by getting a tf-idf(term frequency - inverse document frequency). In this case the idf is called an isf(inverse sentence frequency) to clarify the sentence focus over document focus. The seeding metrics are calculated as follows:

$$tf(t, s)$$

where tf is the term frequency, or simply the number of times term t appears in

document s.

$$isf(t, D) = log(\frac{|D|}{|\{s \in D : t \in s\}|})$$

isf is the inverse sentence frequency, D is the set of sentences in both documents.

$$w(t, s) = tf(t, s) \times isf(t, D)$$

$w(t, s)$ is the t-th coordinate of sentence s.

A sentence pair $(susp_i, src_j)$ is considered a seed if the Cosine and Dice-coefficients are both over 0.33

$$Cos(susp_i, src_j) = \frac{susp_i \cdot src_j}{|susp_i||src_j|}, Cos(susp_i, src_j) \leq 0.33$$

$$Dice(susp_i, src_j) = \frac{2|\delta(susp_i) \cdot \delta(src_j)|}{|\delta(susp_i)|^2 + |\delta(src_j)|^2}, Dice(susp_i, src_j) \leq 0.33$$

with $\delta(x) = 1$ if $x \neq 0$, 0- otherwise[1].

Using the BoW example seen in table 2.3, we can compare this with a similar BoW-model, where day is only mentioned once. From the following sentence from [19] and a sentence modified from this:

$A =$ `Day after day, so many fall away`

$B =$ `Day after, so many fall away`

we can calculate the following metrics: $Cos(A, B) = 0.95$, $Dice(A, B) = 1.0$. The reason for this is that Dice does not look at the frequency of terms, just occurrences. If all words, regardless of frequency, in A are represented in B, then $Dice(A, B) = 1.0$. Cosine takes the frequency in to account as well as the occurrence of words. This yields different kinds of information that may be useful

in determining different forms of plagiarism. A fragment pair within a summary passage may yield a very different Dice value than the Cos value for the same fragment pair.

Some entrants also make use of the Jaccard similarity in various parts of their detector methodologies [14, 12] which is similar in behavior to the Dice metric.

$$Jaccard(susp_i, src_j) = \frac{|susp_i \cap src_j|}{|susp_i \cup src_j|}$$

Glinos [7] stands out from many of the entrants as they use a slightly different approach. By using a modified version of the Smith-Waterman dynamic programming algorithm[20, 21], seeds were generated from the word 1-grams. This algorithm is capable of recursively detecting multiple alignments and join adjacent subsequences. This method therefore also integrates the next step: *extension*. A notable downside to this algorithm is that it requires some programming parameters that need to be manually set. The use of dynamic programming also makes this method challenging to implement in a machine learning pipeline.

In addition to the dynamic programming approach, Glinos also uses multiple clustering algorithms on the most frequent words in the suspicious and source document. However, their method uses a long series of various thresholds, meaning that it is not clustering in the sense that is used in machine learning. They do cluster the n-grams, but by using predefined heuristics [7]. It does suggest that using a machine learning clustering algorithm could aid in the generation of seeds.

Seeding when using n-grams is often very simple. If identical n-grams are discovered in both the source, and suspicious document, then the n-gram pair is passed on to the extension stage as a valid seed along with its offset information. No similarity measurements are needed for this.

2.2.3 Extension

Seed matches are joined in this step using certain algorithms and parameters in order to generate a single passage instead of the many fragments that the seeds make up.

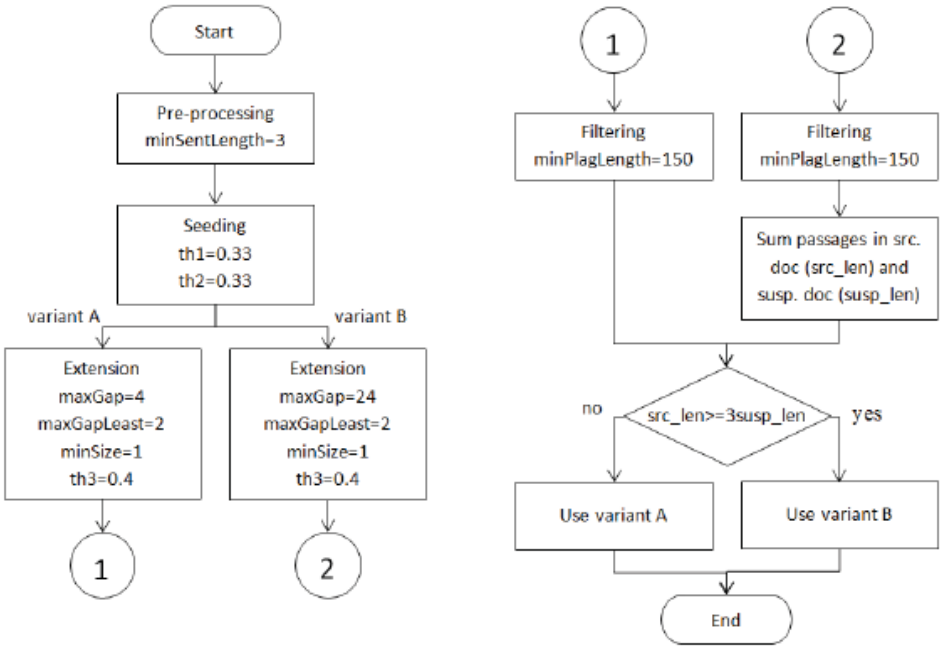

Figure 2.4: Overview and parameters of the Sanchez-Perez algorithm[1]

Some strategies among the entrants share a common idea; Attempt to identify different forms of plagiarism using different methods or parameters. This is visible in figure 2.4, which shows an overview of the entire Sanchez-Perez algorithm. After seeding, the extension is performed twice with different parameters in an attempt to identify plagiarism more correctly. Variant A focuses on the 3 simplest forms of obfuscation while variant B attempts to detect *summary* obfuscation. Both variant A and B can return passages but only one is selected and used per

case based on the threshold heuristic $source\_length > 3 \cdot suspicious\_length$[1]. The algorithm runs a similarity test using the Cosine metric described in section 2.2.2 on a passage generated by seeds within a given threshold of each other in terms of distance in character offset. Cosine must be above a given threshold(th4 in figure 2.4) in order to be classified as plagiarism.

Also visible in figure 2.4 are the 10 unique parameters used(14 adjustable). All of these may be highly data set dependent. Using a threshold like the one determining the extension variant may not work very well in a real setting. There are no guarantees that a summary will be a third of the length of the plagiarised source passage. This would indicate that a parameter-free method is desirable.

Shreshta uses two different approaches when classifying plagiarism. One they call a strict method and one more lenient. The strict method uses a machine translation evaluation metric called TER-p[22] to find an *edit distance* between two translations. This can of course be used to measure text similarity, as it is done here. The TER-p method is sentence-based, whereas the more lenient method is n-gram based. They combine passages where similarity is high in both TER-p score and occurrence n-grams within the same passage. Using an empirically discovered thresholds for merging and short-passage removal, they end up with very promising results.

The idea of mixing both N-grams and sentences is intriguing, but may not be necessary if the features used are good. Shreshta uses two different approaches based on a heuristic. It would be possible to split and use two different classifiers and two different feature sets based on BoW and n-grams respectively. These could be calculated into offsets/length combinations and merged for a final set of XML outputs as the PAN performance tool requires. However, with the right set of features, this approach should not be necessary. Similar results could potentially be produced using multiple classifiers, bagging or boosting.

2.2.4 Filtering

Filtering is the process of reducing granularity, and in some cases improve precision using some algorithm. Passages overlapping one another are joined into single passages in an attempt to remove any granularity. Passages shorter than a given threshold may also be removed. Overlap resolutions and short passage removal was done in Sanchez-Perez using a minimum passage length threshold and an overlap removal algorithm[1]. Minimum passage length threshold is also seen as a common filtering mechanism in other entrants. One example is Gross [16] who used a minimum of 15 words.

Kong 2013 uses an interesting approach in filtering that was not seen in other entrants. A sliding window is applied on a passage in order to maximise the Jaccard coefficient value[12]. The exact algorithm and parameters are not published, but the idea is interesting. This can also easily be applied to other similarity metrics such as Cosine or Dice used in Sanchez-Perez [1].

2.2.5 Summary

Selecting a pre-processing strategy is highly dependant on the tokenization, but conversion to lower case and stemming seems promising in an attempt to increase the accuracy in *random* and *translation* obfuscation.

In terms of seeding there are two major strategies that stand out. N-grams and sentence tokenization. The N-value is a parameter value which greatly can affect the result and which types of parameters a seeding mechanism is capable of detecting proficiently. As a goal of this thesis is to remove dependencies on manual parameters, using sentences as a basis for tokenization seems more natural and can easily be defended as the top methodology uses it[1, 2].

The extension algorithms differ greatly but they often use some form of allowed gap between seeds and can create new passages from these. The method

used in Sanchez-Perez is interesting and possible to implement in a machine learning pipeline. The similarity metrics are reused and a new threshold decides if the extension will be passed on to filtering for further processing[1].

In terms of filtering, the method used by Kong 2013 [12] which implements a sliding window seems very promising and can be easily modified to use other similarity measurements. Overlaps should also be resolved. Looking at minimum passage length is also of interest. This can be easily added in the extension stage by extracting a passage length feature to pass on to the classification algorithm.

Palkovskii attacks the task at hand with a somewhat similar approach overall as this thesis. Combine the known methods into an algorithm in an attempt to outperform each method separately. With an overall PlagDet score of 0.87, precision of 0.92 and a recall of 0.83[2], this method shows very good results.

The algorithm generates a range of n-grams, including regular, stop-words, named entities, frequent words as well as some expansion methods before calculating metrics similar to Sanchez-Perez[1] on these n-grams. After this, the data is clustered using an angled ellipse based graphical clustering method. This is done in an attempt to predict the type of plagiarism that is being performed. This allows them to tweak the parameters for the final detection.

Clustering seems like a good idea for this data set, but their method is parameter dependent, making it less useful in a more general scenario. It would require a lot of work for each new training data set. It does suggest attempting more automated clustering methods may be beneficial and should be looked into.

Torrejón 2014 describes how the *chunk length* needs to be tuned again every year to find optimal settings based on the training data set provided. It for example states that "*[...] optimum settings for the 2012 and 2013 corpora were quite different.*". The goal for the 2014 entry was to find a method of self-tuning the system [15]. With the right parameter settings the 2014 entry shows good results, although very similar to the 2013 entry. However, with poor settings, their results

are far from optimal. The need for parameter re-tuning every year to fit the current data set shows that the need for self-adjusting systems or systems capable of learning may prove very useful, even when the data sets are as similar as they are from year to year.

Chapter 3

Plagiarism Detector Design

This chapter presents the methodology of the plagiarism detector setups used in later experiments. Section 1 provides an introductory overview of the entire pipeline. Section 2 shows the details behind the pre-processing, Section 3 explains the seeding metric and classification process. Section 4 explains how seeds are extended to passages and classified. Section 5 details the filtering mechanism used to optimize results. Section 6 explains how output is generated. Section 7 introduces how the detector will be analysed.

As seen in chapter 2, many methodologies have borrowed a stepwise approach of seeding, extension and filtering from the field of bioinformatics. The approach applied in this thesis does not differ from this general methodology, although it adds extra steps to provide a full and highly modular pipeline. Most entrants in PAN perform the entire process in a single script. A more streamlined solution is suggested for any production ready implementation for easier usability of the pipeline developed in this thesis.

In general, the process makes use of two tools: Python scripts and KNIME Analytics Platform[1]. KNIME is a program that allows users to create data analysis

[1] Available at http://www.knime.org/knime

workflows very quickly and easily. It contains a lot of machine learning algorithms and other tools vital to parsing, analysing and representing various types of data. Parts of the Python code is based on the freely available source code of the Sanchez-Perez entry[1]. This is done for time efficiency and reducing the risk for creating unnecessary bugs. The code has not been optimized for execution time as this is not a vital issue in this setting. Some parts of the KNIME workspaces also use Java for data manipulation.

3.1 Overview

Figure 3.1 shows an overview of the approach. Preliminary and intermediate Python scripts are seen in rounded blue rectangles while the classification steps, currently using KNIME workspaces, are seen in red rectangles. Green, rounded rectangles are Python scripts used for metrics and cleanup. The *Performance Metrics* script is the script supplied by the PAN workshop. In order to avoid reinventing the wheel and thus introducing unnecessary bugs, some parts of the scripts, such as pre-processing, tokenization and basic seed metrics calculations based on PAN entrant metrics, use their source code where available.

3.2 Pre-processing

Prior to any seed generation or metric calculations, the data set in question is prepared equal to the pre-processing performed in [1], using their available code[2]. Each document pair is converted to lower case where applicable before being stemmed and split up into lists of sentences. Sentences shorter than 3 words are joined with the next sentence. Upon completion of these processes, BoW-models are generated. Each step of the pre-processing methodology is explained in detail

[2]Source code available at: http://www.gelbukh.com/plagiarism-detection/PAN-2014/

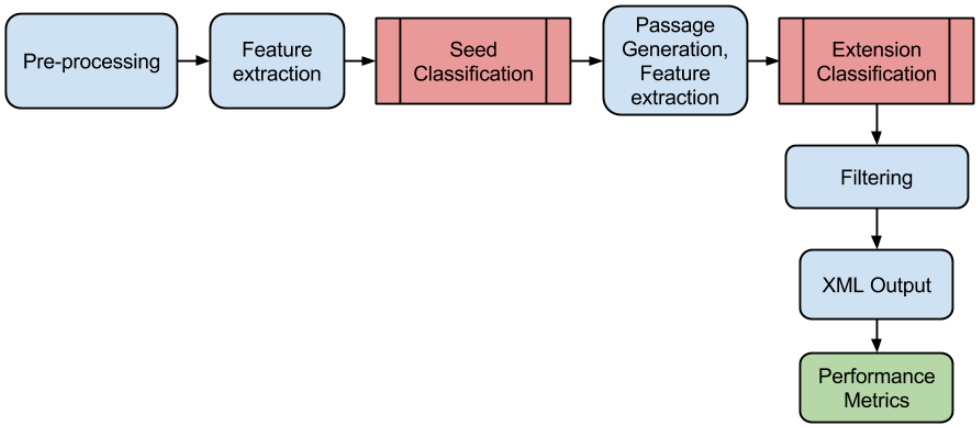

Figure 3.1: Overview of the plagiarism detection methodology

below.

3.2.1 Stemming

This project, like that of Sanchez-Perez[1], makes use of the Porter-stemmer in the Natural Language Toolkit for Python[3], an implementation of the algorithm explained in detail in [9].

The stemming algorithm will not be able to catch synonyms with different roots. This would require a taxonomy of synonyms. An interesting idea, but most likely very costly to compute and it could potentially produce a lot of noise through false similarity. There is also a potential risk of over-generalization of words.

[3]http://www.nltk.org/

3.2.2 Tokenization

The Natural Language Toolkit for Python has a tokenizer for english(usable as *tokenizers/punkt/english.pickle*), which is used by Sanchez-Perez [1]. This method is used in this detector without changes from the original source code except for adaptation to the pipeline.

3.2.3 Bag of Words

After pre-processing, Bags of Words(BoW)-models are generated for each sentence. This is done using the methodology and selected parts of the source code from Sanchez-Perez [1], again, adapted to fit the pipeline. BoW-generation is performed several times throughout the pipeline.

3.2.4 Labeling

In order to train the classifier, the seed candidates must be labeled as plagiarism or non plagiarism. A seed candidate contains 4 points of data to determine its location within the active document pair. Source offset and length as well as suspicious offset and length. An offset i combined with a length n creates a fragment containing a set of characters $F_{doc} = \{i, i+1, i+2, ..., i+n\}$, from a document doc.

Each seed candidate therefore contains a set of characters $C = F_{src} \times F_{susp}$ from sets of characters F_{src}, F_{susp} from a source and suspicious document respectively. C represents all character pairs within the candidate. A similar set $P = F_{src} \times F_{susp}$ is created to denote plagiarism in the active document pair based on similar data points from the data set.

Character intersection $I = C \cap P$ is marked as plagiarism if $I \neq \emptyset \cap \frac{|I|}{|C|} > t$, where t is a threshold. This threshold is not determined by experiments, but by

thresholds used in performance by PAN[2]. When determining cases, a case is correctly classified when recall and precision > 0.5 in terms of character classification. This means that half of the detected characters must be within the plagiarised passage and half of the passage must be detected for a case to be correctly classified. This resulted in $t = 0.5$. This provides a balanced labeling of fragments. Detailed explanation for performance testing follows in section 3.7. This is done here despite attempting to classify at a character level as a help to the seed classifier by increasing precision at a possible expense of recall.

3.3 Feature Extraction and Seed Classification

After the pre-processing is finished a range of metrics are calculated on each fragment pair. A fragment pair is defined as a fragment i from a set of fragments F from a suspicious document and a fragment j from a set of fragments F from a source document. These are compared against each other using two metrics: The Cosine and Dice coefficient as used in [1] and described in section 2.2.2. These metrics are applied to the BoW-models doing an exhaustive comparison of all possible fragment pairs $E_{ij} = F_i \times F_j$ in a given source and suspicious document.

Jaccard was also considered, but it does not give any more information than Dice does, at it also does not take term frequency into account.

In Sanchez-Perez [1] only the raw Cosine and Dice metrics were used for seeding, only calling it a seed if the following statement is true:

$$Cos(susp_i, src_j) > 0.33 \cap Dice(susp_i, src_j) > 0.33$$

i and j marks fragment or sentence indicies from suspicious and source documents $susp$ and src respectively.

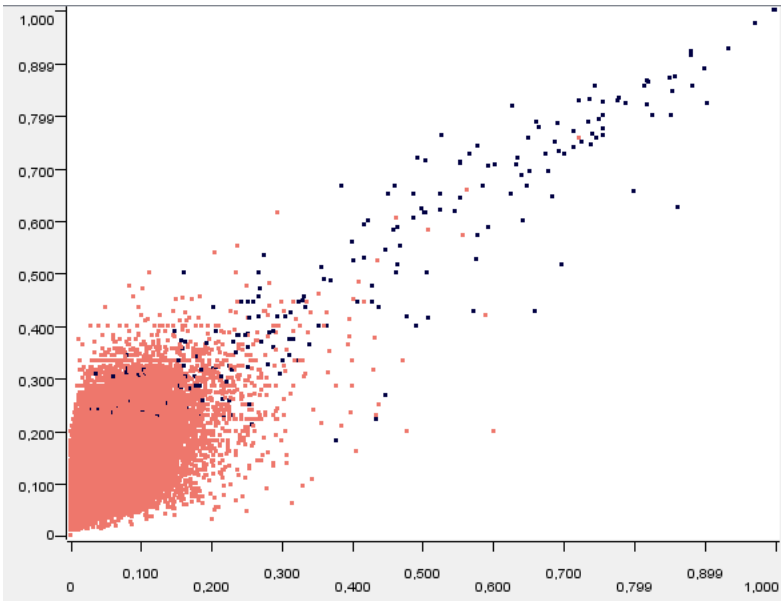

Figure 3.2: Cos(x) - Dice(y) plot over a set of fragments from multiple documents. Seeds within plagiarised passages in black, fragments outside in pink.

Figure 3.2 also shows some of the problem. Hidden among all the non-plagiarism seeds with $Cos(susp_i, src_j) < 0.33 \cap Dice(susp_i, src_j) < 0.33$ are quite a few seeds within plagiarised passages. Determining if these are plagiarism or not can prove difficult without more features. The graph shows that the threshold used by Sanchez-Perez provide a very high accuracy but could exclude plagiarism seeds.

This also shows that using clustering on these simple metrics is not possible without a cleaner data set. It may provide some benefits in terms of accuracy, but very little that other machine learning algorithms are unable to do.

The approach in this thesis expands on these two metrics by extracting more features from them by utilizing statistical and seed-relational metrics to further separate plagiarism from non-plagiarism.

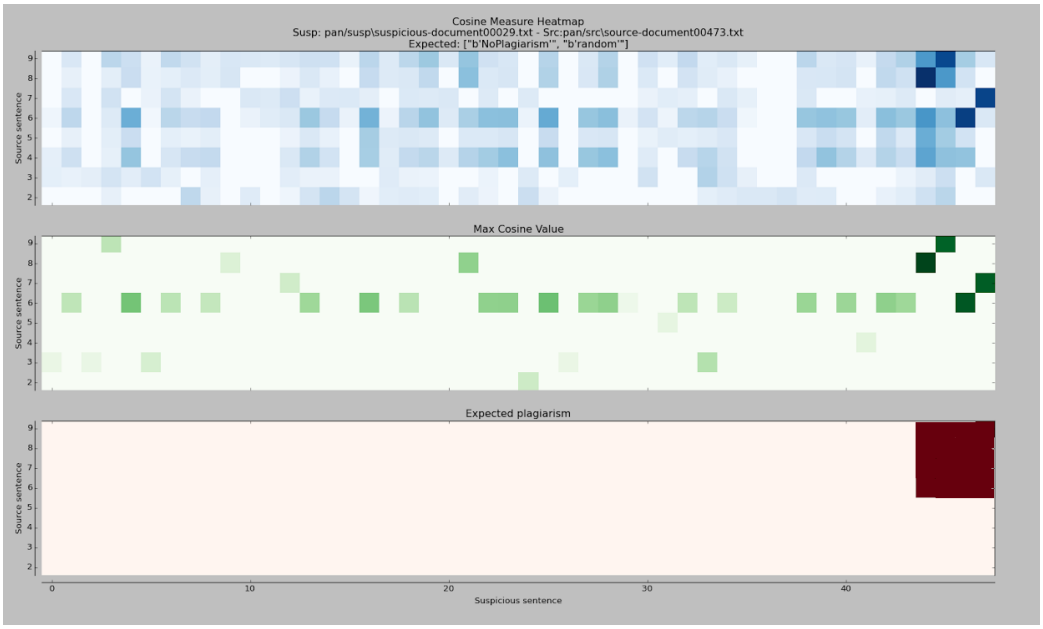

Figure 3.3: Randomized plagiarism example

Figure 3.3 shows the usefulness of metrics such as Cos and Dice. This 3-part graph shows the cosine values(darker is higher, more similar) for sentence pairs between 2 documents in blue, the maximum vertical cosine value for each suspicious sentence in green, and the area of known plagiarism in red. The idea of applying added metrics was conceived when it became apparent that the data set not only randomised words within a sentence but also mixed sentence order in most forms of plagiarism. Plagiarised fragments are not guaranteed to come *in order* within a known plagiarised passage.

This proposed a challenge in terms of training machine learning algorithms when applying it to the seeding stage. In the example in figure 3.3, the plagiarised passage is 4 sentences long in both the suspicious and source document. A fairly simple one to detect with the human eye using this graph, however more difficult to train on. Saying that all seeds within the entire plagiarised area marked in red is plagiarism is wrong and will result in very noisy training data. 4 out of 16

fragments are in this case plagiarised sentences leaving 12 that are not. As visible in the top sub-graph, there are seed candidates with little to no similarity at all, which would greatly confuse the machine learning algorithm.

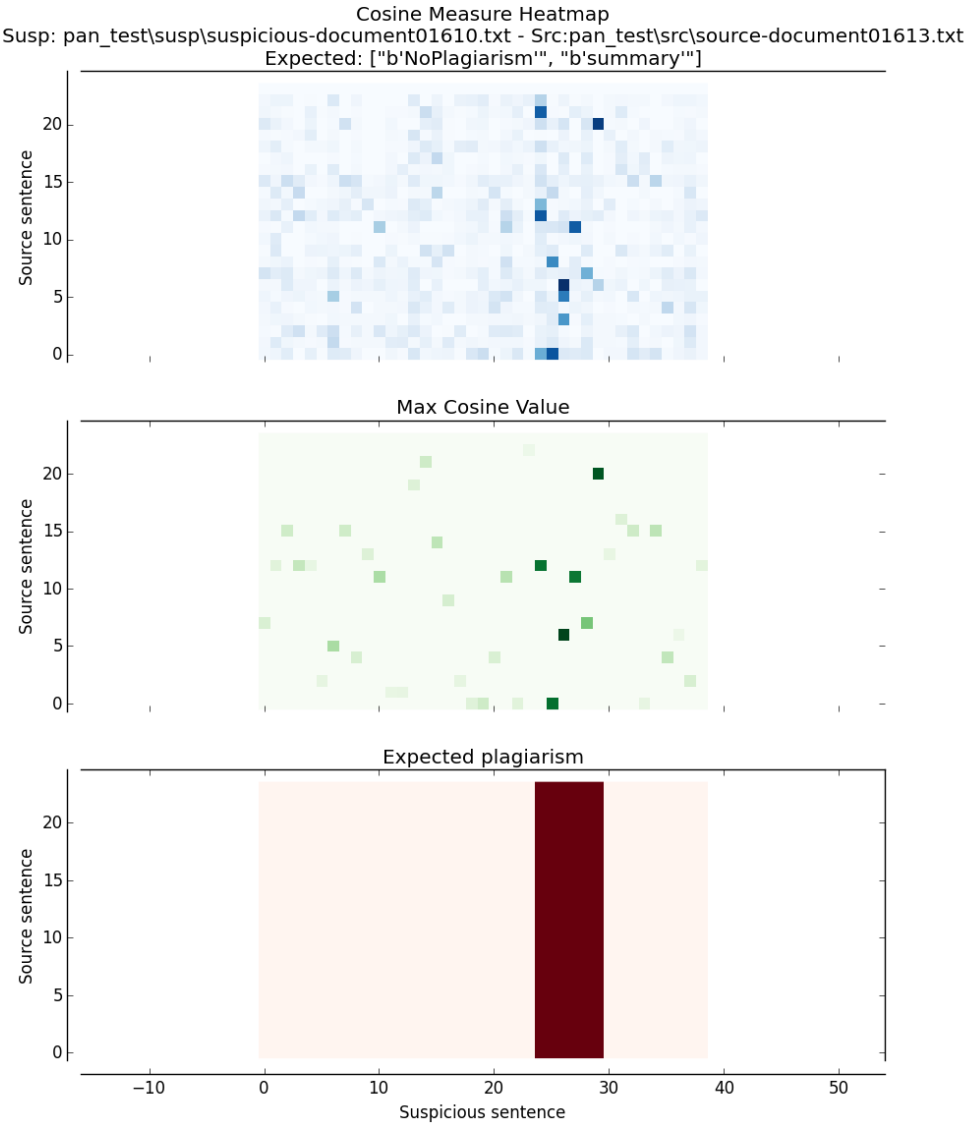

Figure 3.4: Summary plagiarism example

The challenge is even more visible in 3.4. This shows a case where part of

the suspicious document is a summary of the entire source document. The seeds are horizontally adjacent to each other in the suspicious document but vertically far apart in the source document. There are also a lot more seeds that should not be classified as plagiarism when comparing it to figure 3.3. Some seeds stand out although they are not dark blue, meaning that detecting seeds is not as easy as checking for a high threshold along the lines of $Cos_{i,j} > 0.8$. Several document pairs contain single scattered seeds that have a higher Cos or Dice similarity than many of the plagiarised seeds.

After conversing with the Task Chair for the Text Alignment task at PAN, Martin Potthast, it became clear that there is no direct way of extracting the plagiarised seeds from the PAN corpus. No logs from the data set creation were available. Finding a method of cleaning the data set or pointing the algorithm in the right direction using clustering or heuristics would perhaps prove useful.

A more lenient version of the Sanchez-Perez seeding method is used to filter out the worst cases of falsely labeled plagiarism. Using the heuristic $Cos(susp_i, src_j) > 0.1 \cap Dice(susp_i, src_j) > 0.1$ as a fragment filter prior to seeding classification, the data sent to the classifier will be less noisy. Using manual thresholds is far from optimal but with such a noisy data set it should improve the training data. Seeds with similarity metrics as low as 0.1 could with a low probability be summaries but are unlikely to be valid cases of simpler forms of plagiarism.

The value of the threshold should affect the classifier's performance but is not a parameter that should be tweaked in order to increase it, and is unlikely to be the optimal threshold. The main point is that it is substantially lower than the seed similarity threshold used by Sanchez-Perez [1] and is not the main data point seeds are to be classified on. The purpose of this thesis is to avoid thresholds in order to detect plagiarism. The threshold is not used for classification but to clear the training data set. It is so low that it should have little effect, but mitigate the noise issue to some extent. The mean values of Cos and Dice throughout the data set are lower than 0.02 and 0.09 respectively. Along with figure 3.2, this suggests

that the mentioned cleaning thresholds will not remove a lot of plagiarism cases, as the data set mainly consists of non-plagiarism in terms of characters.

When first studying these graphs, it became apparent that the maximum Cosine, and Dice values were often the plagiarised seed, as seen in figures 3.3 and 3.4. A novel feature was created for both of the similarity metrics. The boolean values $IsMaxCos$ and $IsMaxDice$ are defined as *true* if:

$$IsMaxCos(susp_i, src_j) = True \iff Cos(susp_i, src_j) = \operatorname{argmax} Cos(susp_i, src_{0...n})$$

$$IsMaxDice(susp_i, src_j) = True \iff Dice(susp_i, src_j) = \operatorname{argmax} Dice(susp_i, src_{0...n})$$

, where n denotes the number of fragments on the vertical axis(source document). These booleans are *false* if their respective requirements do not hold.

Using the heuristic $withinPlagPassage(susp_i, src_j) \cap (isMaxCos(susp_i, src_j) \cup isMaxDice(susp_i, src_j))$ to clean the data set to some extent without using thresholds the most obvious candidates could be extracted. This would likely mean discarding a great deal of valid candidates. This could especially be the case in the summary plagiarism type, as seen in 3.4. They will not include all cases of plagiarism in the training data, but should provide a fair assistance in cleaning the data and increase precision. Seeds are labeled as plagiarism if this heuristic holds *true*, and is *false* otherwise.

In addition to the simple $IsMax$ feature other novel features were created, such as $MaxDiff$ and $MeanDiff$ for both Cosine and Dice. These are based on the maximum values for metrics generated along the vertical axis, and the document pair mean of these maximum values. Distances to these values are calculated for all source/suspicious sentence pairs.

$$MaxCosDiff(susp_i, src_j) = \operatorname{argmax} Cos(susp_i, 0 \ldots n) - Cos(susp_i, src_j)$$

$$MaxDiceDiff(susp_i, src_j) = \operatorname{argmax} Dice(susp_i, 0 \ldots n) - Dice(susp_i, src_j)$$

$$MeanCosDiff(susp_i, src_j) = \frac{\sum_{k}^{m} \operatorname{argmax} Cos(susp_k, src_{0\ldots n})}{m} - Cos(susp_i, src_j)$$

$$MeanDiceDiff(susp_i, src_j) = \frac{\sum_{k}^{m} \operatorname{argmax} Dice(susp_k, src_{0\ldots n})}{m} - Dice(susp_i, src_j)$$

m and n note the last sentence index of the suspicious and source documents containing sentence indicies i and j from documents $susp$ and src respectively.

As visible in figure 3.4, the length of the plagiarised passage is much longer in the source document than in the suspicious document, as is often the case of a textual summary. This means that given just the $isMax$ method, all seeds can not be found. A maximum of two seed candidates can be produced for each suspicious sentence given that the same sentence pair is not the maximum value for both Cosine and Dice. In many cases they are the same, and will produce only one.

Even though a seed may not be the best candidate for plagiarism, the nearby neighbours should be considered. Especially in cases seen in figure 3.4 where nearby seeds to the most obvious plagiarism have increased metric similarities. Features that indicate neighbouring plagiarism may increase the likelihood of classifying nearby seeds with lower, but still significantly high metrics as relevant seeds.

Different novel features were designed both horizontally(between suspicious

seeds) and vertically(source seeds), even though they differ in order to solve different challenges. Selected features are extracted from each of the base metrics to represent the maximum Cos/Dice values for the immediate neighbour in either horizontal direction.

$$\begin{aligned} &MaxCosNeigbour(susp_i, src) \\ &= \text{argmax}(\text{argmax}\, Cos(susp_{i-1} src_{0\ldots n}), \text{argmax}\, Cos(susp_{i+1}, src_{0\ldots n})), i \\ &> 0, i \\ &< m \end{aligned}$$

$$\begin{aligned} &MaxCosNeigbour(susp_i, src) \\ &= \text{argmax}(\text{argmax}\, Dice(susp_{i-1}, src_{0\ldots n}), \text{argmax}\, Dice(susp_{i+1}, src_{0\ldots n})), i \\ &> 0, i \\ &< m \end{aligned}$$

, where m and n again denote the last sentence indicies of suspicious and source document sentences respectively. In the event that $i = 0$ or $i = m$, the only available neighbour is used to calculate these values, using $i + 1$ or $i - 1$ as neighbour index, so that this index is guaranteed to exist in the fragment pair array.

Using $where(x, S) = y$ as a function that returns the location y of x in an ordered set S.

$$\begin{aligned} &VerticalCosMaxDist(susp_i, src) \\ &= |where(\text{argmax}\, Cos(susp_i, src_{0\ldots n}), Cos(susp_i, src_{0\ldots n})) - i| \end{aligned}$$

$$\begin{aligned} &VerticalDiceMaxDist(susp_i, src) \\ &= |where(\text{argmax}\, Dice(susp_i, src_{0\ldots n}), Dice(susp_i, src0 \ldots n)) - i| \end{aligned}$$

The vertical distance features return the distance from the maximum metric value in terms of sentence index difference. This is always an absolute, positive value. The interest lies in the vertical distance from the maximum metric value.

Whether it is above or below is of little interest to a classifier. These features are designed in an attempt to fill out the plagiarism *rectangle* and classify more relevant seeds that may not necessarily be plagiarism. It adds information as to the relative location of the most likely seed. This may also help segregate non-plagiarism seeds with fairly high metric values from plagiarism seeds with similar values. A seed which is near a high metric value should be more likely to be a plagiarism seed than a fragment pair 50 sentences away with a similar metric value.

This is designed with the summary plagiarism type in mind. Figure 3.4 demonstrates the need for this. The number of seeds that confuse the classifier are numerous and the line between plagiarism and non-plagiarism within the plagiarised passage is very fine, as seen in the top sub-graph.

In an attempt to further aid the classifier to detect *summary* plagiarism, a ratio feature describing the ratio between the length of the suspicious and source passage was created.

$$SrcSuspLenRatio(susp_i, src_j) = \frac{|susp_i|}{|src_j|}$$

The output from the extension is passed on to a workspace in KNIME for further processing and classification. This workspace is set up to be capable of using multiple classifiers in order to find the optimal algorithm for this task.

The classifiers are trained using the complete data set available after PAN 13, which is the same the PAN 14 entrants used for their optimizations. The classifiers are then tested on the PAN 14 test set. Both data sets are from a larger corpus where some data is withheld from the general public. A new subset from the large corpus is released each year for testing. The corpus is discussed in detail in section 2.1.2. The training/testing data split described here is done for all classifiers in all classification steps of the methodology in this thesis. The reason for this is to avoid training on any data that it is tested against.

Bayesian spam filtering is a common method of classifying e-mail spam [23] and is a classifier that takes both little time to train and run. With history in a linguistic field, naïve Bayes [24] is an obvious candidate to be tested. It will however be used differently in this thesis as it here works on numerical features instead of word occurrence probabilities. This classifier has seen use in a similar field to this thesis, in an attempt to detect source code plagiarism[25]. This is however a different problem as the features used here are very different due to the differences in human languages and programming languages.

A challenge with naïve Bayes is that it requires some form of feature selection for optimal performance. Passing all features through this algorithm may cause noise and confuse the classifier. The available subsets of features should be tested. With the number of available features and a large training data set, testing for the optimal feature set takes a series of runs. Development testing also showed that it had a tendency to return a large number of false positives.

Decision trees [24] often strike a good balance between recall and precision. Development testing showed a decrease in recall when compared to naïve Bayes, but a higher precision. An advantage with decision trees is that they by nature perform a form of feature selection. All features are not always used, and they attempt to use the features that separate the data the best way possible.

An extension to decision trees are *random forests*. These consist of sets of decision trees using randomised sets of features. Majority voting among the generated decision trees decide how the data should be classified.

An overview of all seeding features with a brief explanation is shown in table 3.1.

Feature	**Value**	**Description**
Cosine	Float	BoW similarity measurement. Frequency dependent
Dice Coefficient	Float	BoW similarity measurement. Frequency invariant
IsMax(Cos/Dice)	Boolean	Returns TRUE if it is source fragment with the highest Cos/Dice value for that suspicous fragment
MaxDiff(Cos/Dice)	Float	Difference between the current source fragment and the source fragment with the highest maximum similarity value of the same type for that suspicious fragment.
MeanDiff(Cos/Dice)	Float	Difference between the current source fragment and the document mean maximum values of the same type for that suspicious fragment.
MaxNeighbour(Cos/Dice)	Float	The highest value of the given similarity type for immediate suspicious fragment neighbours.
VerticalMaxDist(Cos/Dice)	Integer	Distance in fragments between the current source fragment and the fragment with the highest similarity value of the given type.
SrcSuspLenRatio	Float	The length ratio between the source and suspicious passages.

Table 3.1: Overview of features extracted from fragments before seeding classification

3.4 Passage Generation and Extension Classification

In the state of the art and this thesis, extension is the process of joining seeds to form a passage and evaluating this as a whole to determine a case of plagiarism. The process here is fairly simple. For a document pair, create lists of seeds that are horizontally adjacent to each other in the suspicious document, or share the same offset in the suspicious document.

A seed $S(i, j)$ is added to the ordered set G ordered by i if: $S(i, x) \in G$, or $S(i - 1, x) \in G$, where x can hold any value. G is referred to as a *group* of horizontally adjacent seeds.

A passage may contain several groups if a seed somewhere within the plagiarised passage has been falsely classified as non-plagiarism. To solve this problem, a novel approach is suggested:

All detected groups within a document pair are paired in order to generate all possible pairs of groups so that multiple candidate passages can be generated. This method ensures that all gaps are covered. Only seeds missing before the first classified seed in the first group and after the last seed in the last group can no longer be detected.

A set of passages P is generated by $P = G \times G$. For all passages $p \in P$ the minimum and maximum character offsets horizontally and vertically of all groups within p are calculated to form the total passage.

In Sanchez-Perez [1] they allow a gap below a certain threshold of characters to still pass it off as a single passage. Allowing no gap at all horizontally may increase granularity but could also increase precision. If seeds are missed there is a risk of lowering recall. Not requiring adjacency would likely do the opposite. This is another manual threshold that should be avoided to allow for flexibility when training with new data sets. In a real life setting, having a slightly increased granularity resulting in a split passage or false negative seeds mid-passage is not a

big issue when detected plagiarism cases are checked by teachers or other hand-in censors.

To attack the problem of such a threshold, a novel method is created in this thesis. Instead of using an absolute threshold, a normalized ratio based on the length of the passage is used. A group $g \in G$ holds a set of character offsets C_g vertically ($C_{g_{src}}$) and horizontally ($C_{g_{susp}}$). $GroupLenSrc$ and $GroupLenSusp$ are defined as follows:

$$GroupLenSrc = \sum_{g \in G} |C_{g_{src}}|$$

$$GroupLenSusp = \sum_{g \in G} |C_{g_{susp}}|$$

These two metrics are not used on their own as new ones, but lead to two new features: $SuspGapRatio$ and $SrcGapRatio$. These are defined as follows:

$$SrcGapRatio = \frac{GroupLenSrc}{|C_{p_{src}}|}$$

$$SuspGapRatio = \frac{GroupLenSusp}{|C_{p_{susp}}|}$$

, where C_p are the set of characters in the passage p generated from joining the two groups used to calculate $GroupLenSrc$ and $GroupLenSusp$.

The passages are compared in much the same way as seeds are generated, but using the passages instead of sentence pairs. BoW-models are created from the passages and Cos as well as Dice are used again to measure passage similarity. These definitions are equal to those used in section 3.3.

Three new features are generated based on these two similarity measurements.

$$CosDiceAvg = \frac{Cos(susp_i, src_j) + Dice(susp_i, src_j)}{2}$$

$$CosDiceDiff = Cos(susp_i, src_j) - Dice(susp_i, src_j)$$

$$AbsCosDiceDiff = |Cos(susp_i, src_j) - Dice(susp_i, src_j)|$$

In addition to these, the $SrcSuspLenRatio$ is calculated again for the passage, as defined in section 3.3.

The state of the art often is seen to sometimes use a specified length threshold to remove passages that are too short in the filtering stage[1, 16]. This is performed in the extension classification stage in this thesis by analysing lengths of the source and suspicious passages. These features are defined simply as:

$$SrcLen = |src_j|$$

$$SuspLen = |susp_i|$$

In addition to these, the minimum length of the passage is also extracted as a feature. $MinLen$ is defined as:

$$MinLen = argmin(SrcLen, SuspLen)$$

The passage also now needs to be re-labeled as plagiarism or non-plagiarism for the classifier. The same condition as for seeding applies. If the ratio of correctly classified characters is over 0.5, the passage is labeled as plagiarism.

After these features are extracted and labeling is completed, the candidate passages are passed through the final classifier. Like the seed classifier, a KNIME workspace has been created containing a series of machine learning algorithms in order to find the optimal classifier. Any passage classified as plagiarism is passed on to the filtering and output process.

Because most of the seed candidates are removed in the seeding step, the extension classifier receives fairly few true negatives to begin with, making it a challenge to classify non-plagiarism. During test runs on low amounts of data, some classifying algorithms returned error messages since they were unable to run when there was only a single class to classify. The seeding stage had in some test cases yielded a precision of 1.0, meaning that all passages were plagiarism. Given enough model data, this is no longer a big issue.

An overview of all extension features with a brief explanation is shown in table 3.2.

The same classifiers used for seed classification were tested for extension classification using a similar data set split.

Feature	Value	Description
Cosine	Float	BoW similarity measurement. Frequency dependent.
Dice Coefficient	Float	BoW similarity measurement. Frequency invariant.
CosDiceAvg	Float	Average of Cos and Dice.
CosDiceDiff	Float	Difference between Cos and Dice.
AbsCosDiceDiff	Float	Absolute difference between Cos and Dice.
SrcSuspLenRatio	Float	Length ratio between source and suspicious passages.
SrcGapRatio	Float	Ratio of passage not covered by seeds on the source axis.
SuspGapRatio	Float	Ratio of passage not covered by seeds on the suspicious axis.
SrcLen	Integer	Length of the source passage
SuspLen	Integer	Length of the suspicious passage suspicious passages.
MinLen	Integer	Shortest length of the source and suspicious passages.

Table 3.2: Overview of features extracted from fragments before extension classification

3.5 Filtering

Filtering is the process of removing any surplus or erroneous classifications of plagiarism where possible. The goal of overlap removal is to decrease granularity. Any other filtering steps are applied to increase precision without significantly lowering recall.

Overlaps are removed by checking for offset overlaps between any passages within the same document pair. For all passages $p \in P$ that contain the data $p =< offset_{src}, offset_{susp}, length_{src}, length_{susp} >$, character sets $c = \{offset_{susp}, \ldots, offset_{susp} + length_s$ are generated and put in C. Any sets c in C which overlap are joined.

In the case of a seeding stage with low precision, passages are sometimes too large, thereby classifying many seeds falsely as plagiarism. Development testing shows that this is especially true for the vertical source document axis, perhaps due to the vertical distance features. Finding a case of plagiarism is not too difficult. Finding the *edges* of the case in both suspicious and source documents is far more trying task. The idea is that in the event that the detected plagiarism exceeds the boundaries of the true plagiarism case, the filtering algorithm will attempt to remove misclassified seeds from the edges of the passage in an attempt to optimize the location of the start and end offsets on both the horizontal and vertical axis.

Filtering in this project is inspired by the sliding window in approach used by Kong[26]. The exact details of the algorithm are not specified, but a simple approach based on the understanding of the concept is used to determine whether it had a positive effect on the results.

A plagiarised passage has an offset and a length both horizontally and vertically. The goal is to minimize the square generated on these two axes in order to maximise the Cos and Dice metrics. Seeing a plagiarised passage as a square, as shown in figure 3.5, aids in understanding the algorithm.

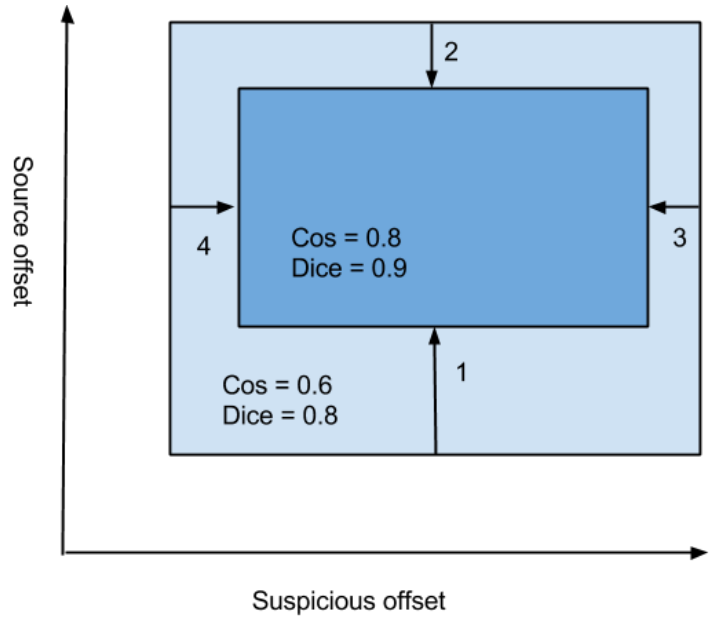

Figure 3.5: Graphical demonstration of sliding window filtering mechanism

More optimal approaches will most likely exist by looking for better global maximums, or by performing exhaustive searches. Another possibility would include attempting to expand the square to determine whether it caught all seeds near the edges, in order to improve recall. This is highly suggested for future research.

This algorithm is *greedy*, meaning that it will stop immediately when the results are worse than the metrics calculated from the current best passage. As long as the shortened candidate passage has a better Cosine or Dice metric value, it will continue to shorten in the given direction. When both of them are worse than the best, it moves on to the next direction in an attempt to improve the result further here.

It shortens the vertical source axis from the front of the passage before shortening from the back. After this it moves on to the suspicious front, then back. The

order in which this is done may be of some significance due to the greedy nature of the algorithm. Development testing showed that it seemed more willing to strip seeds if the vertical source axis is minimized first.

3.6 Output

The methodology currently outputs XML that specifies the offset and length of each plagiarised passage in both suspicious and source documents. PAN supplies a python script that compares a standardized XML output against their own in order to extract performance metrics on a character level.

Figure 2.1 shows the correct plagiarism for the document pair `suspicious-document00016` and `source-document01836`. Some of these fields can not be filled. The plagiarism tool is not required, nor is it set up to detect the type of plagiarism, so the `obfuscation` field can be omitted in the output. This simply states the obfuscation type used on the document pair. The `type` field can also be omitted. Figure 3.6 shows an example output for the same document pair as in figure 2.1. It is visible here that it has not hit all the seeds in the passage; It seems to be missing a sentence or two at the beginning and at the end of the suspicious passage while only missing seeds at the beginning of the source passage. The cases overlap with maximum precision and some reduced recall.

```
<document reference="suspicious-document00016.txt">
    <feature name="detected-plagiarism"
    source_length="539" source_offset="7384"
    source_reference="source-document01836.txt"
    this_length="448" this_offset="6897"/>
</document>
```

Figure 3.6: XML output ready for comparison with PAN data sets (suspicious-document00016-source-document01836.xml).

3.7 Comparison and Performance Analysis

The detector will be benchmarked against the other PAN entrants using the performance metrics supplied by PAN[2].

They use mainly common performance metrics such as recall, and precision, but also create new ones to solve problems of overlapping. Detailed information about the design of these metrics is available in [2].

Performance is measured on three levels: Character, case and document level. A brief explanation may be beneficial before defining these levels formally.

The character level calculates performance based on the character offsets of a detection and plagiarism case. A character has been correctly aligned with a plagiarism case if its offset within a source or suspicious document is within the offsets described by the XML file in the data set.

A case has been correctly classified if more than half of the characters within the detected passage are within the offsets of the plagiarism case, and more than half of the plagiarism case has been covered by the detected plagiarism.

A document pair has been correctly classified if at least one case within the document pair has been classified correctly, as stated above.

All of the formal definitions below were designed by, and for PAN[6, 2].

3.7.1 Character Performance Comparison

A plagiarism case $s =< s_{susp}, d_{susp}, s_{src}, d_{src} >, s \in S$ is a set of references to the characters of d_{susp} and d_{src}, in passages s_{susp} and s_{src}. Detected cases $r \in R$ are represented similarly. r detects s if $s \cap r \neq \emptyset$ and s_{susp} overlaps with r_{susp} and s_{src} overlaps with r_{src}. Recall and precision are therefore denoted as:

$$Prec(S, R) = \frac{1}{|R|} \sum_{r \in R} \frac{|\bigcup_{s \in S}(s \cap r)|}{|r|}$$

$$Rec(S, R) = \frac{1}{|S|} \sum_{s \in S} \frac{|\bigcup_{r \in R}(s \cap r)|}{|s|}$$

For clarity, the definitions previously defined in section 2.1.3 are repeated.

Let S denote the set of plagiarism cases in the corpus. R denotes the set of plagiarism detections by a detector. Granularity is defined as:

$$gran(S, R) = \frac{1}{S_R} \sum_{s \in S_R} |R_S|$$

$S_R \subseteq S$ are detected cases in R while $R_S \subseteq$ R are detections in S

$F_1(S, R)$ is the harmonic mean of precision and recall and is defined as:

$$F_1(S, R) = 2\frac{Prec(S, R) \cdot Rec(S, R)}{Prec(S, R) + Rec(S, R)}$$

F_1 is not used on its own in character level performance analysis as it does not account for granularity. Instead, along with recall and precision, these metrics make up the performance score designed for PAN. PlagDet is defined as:

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

3.7.2 Case Performance Comparison

Notations are reused from previous definitions where applicable. R' represents a subset of R where precision and recall are larger than a given threshold. S' denotes a similar subset from S which includes all plagiarism cases detected from S with character recall and precision larger than a predetermined threshold. This threshold has in [2] been set to 0.5 for both recall and precision to give a good balance between fraud and insult. This means that for a case to be correctly classified, over half of the detected characters must be within the plagiarised passage, and more than half of the plagiarised passage must be covered by the detection. Precision and recall for cases are described as:

$$Prec_{case}(S, R) = \frac{|R'|}{|R|}$$

$$Rec_{case}(S, R) = \frac{|S'|}{|S|}$$

F_1 is used also here as defined in 3.7.1 and is used instead of PlagDet in case level performance analysis.

3.7.3 Document Performance Comparison

A document detection occurs when at least one plagiarism case in a document pair has been correctly detected. A set of document pairs $D_{pairs} = D_{susp} \times D_{src}$

provides the set of possible comparisons the detector can make. S denotes the set of plagiarism cases in the corpus. R denotes the set of plagiarism detections.

$D_{pairs|S} = \{(d_{susp}, d_{src})|(d_{susp}, d_{src}) \in D_{pairs} \cup \exists s \in S : d_{susp} \in s \cup d_{src} \in s\}$ represents the subset $D_{pairs|S} \in D_{pairs}$ which contains the plagiarism cases S while $D_{pairs|R} = \{(d_{susp}, d_{src})|(d_{susp}, d_{src}) \in D_{pairs} \cup \exists r \in R : d_{susp} \in r \cup d_{src} \in r\}$ represents the subset $D_{pairs|S} \in D_{pairs}$ which contains detected plagiarism cases in R.

The 2014 PAN plagiarism task overview paper [2] denotes an R' which represents a subset of R containing detections that are over a certain threshold in terms of recall and precision, but this is not necessary in document level. Only a single case detection is required for this, so a simplified version of precision and recall is as follows:

$$Prec_{doc}(S, R) = \frac{|D_{pairs|S} \cap D_{pairs|R}|}{|D_{pairs|R}}$$

$$Rec_{doc}(S, R) = \frac{|D_{pairs|S} \cap D_{pairs|R}|}{|D_{pairs|S}}$$

As with case performance, F_1 is defined as it is in section 3.7.1 and used instead of PlagDet in document level performance analysis.

Chapter 4

Experiments and Results

This chapter presents the experiments performed and why they were designed the way they were, as well as their results. Section 1 presents the results from preprocessing. Section 2 shows the efficiency of the seeding stage. Section 3 presents the performance of the extension stage. Section 4 shows the efficiency of the sliding window filtering algorithm. Section 5 shows the most accurate algorithm combination from the previous results and the final detector methodology. Section 6 compares the best results of this methodology to the entrants of PAN 2012-2014.

All experiments are tested using the PAN 14 test data set as described in section 3.3. Performance metrics are calculated as described in section 3.7

The purpose of the module experiments are to test each step of the algorithm in order to determine the effectiveness and to potentially discover any flaws or points that can be improved upon.

The overall experiments are used to build a benchmark against the existing state of the art, more specifically against the entrants of PAN 12-14.

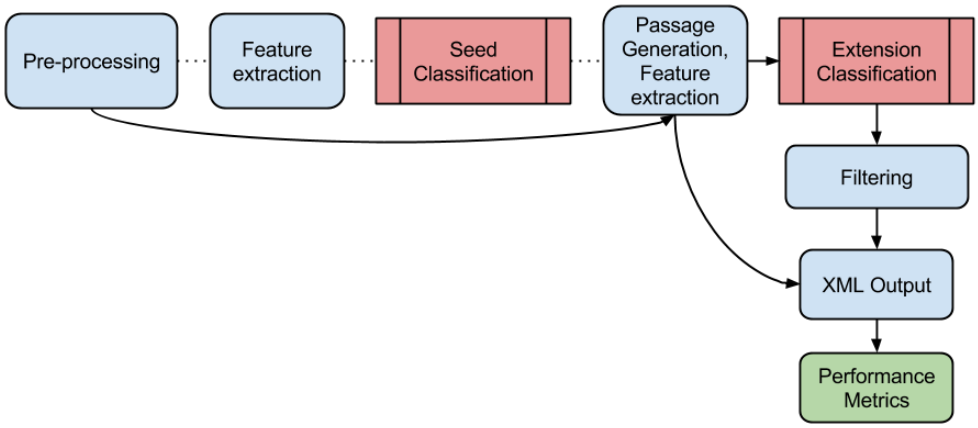

Figure 4.1: Active modules during pre-processing tests

4.1 Pre-processing

Certain potential issues regarding the pre-processing steps Sanchez-perez [1] were previously mentioned in section 2.2.1. In order to establish whether these suggestions are correct, experiments should be run already here, in order to determine if this step can be improved upon.

This is done by performing the pre-processing as suggested earlier. Any seed candidates that come from this pre-processing that overlap with known plagiarism in the data set are labeled as plagiarism and passed on to the XML-output. Using the PAN 2014 performance script the recall and precision on a character level are extracted. Granularity is overlooked in order to disregard any potential bugs from a filtering mechanism. Case and document level performance is calculated using a script developed for this thesis as a PAN developed script for this purpose was not discovered.

Any result here should be the absolute best any later step of the algorithm is capable of performing in terms of recall, as this is a waterfall system. Any poor classification in one step can easily propagate to the next step. The precision may

in some cases increase, but likely at the expense of recall.

Precision is the most interesting number in this experiment. Imperfect precision is expected due to observed issues with the sentence segmentation as explained in chapter 2.2.1.

Obfuscation method	Recall	Precision	Granularity
No obfuscation	.99	.88	1.0
Random	.99	.93	1.0
Translation	.98	.90	1.0
Summary	.99	.99	1.0
Overall	.99	.91	1.0

Table 4.1: Pre-processing PAN performance metrics

Table 4.1 displays some very interesting results. All obfuscation strategies have imperfect recall. This is suspected to come from a series of off-by-one errors in terms of passage length. It seems that it in some cases is one character too short. Exactly how these passages are divided by PAN is unknown, but the sentence splitting done by the pre-processing could potentially be missing end punctuations. This is a bug of very little importance as it should not dramatically affect the results. It should also affect the results of the top PAN entrant Sanchez-Perez as well, since this particular piece of code is shared.

The results also show that the precision is far from optimal. After examining the XML output from this experiment it became apparent that the pre-processing method often overshoots the size of the plagiarism, both where it starts and where it ends. Seeds are generated by a given rule set but this does not guarantee that plagiarism follows this rule set. Plagiarism does not necessarily start at the beginning of a sentence. The seed generation that follows from pre-processing will label a fragment as plagiarism if more than half of the characters within it intersects with a known plagiarised passage based on the data set.

The most interesting result here is the *summary* form of plagiarism. The recall is similar to the others, but the precision is notably higher and close to perfect. This would allow later stages to have a slightly higher chance of detecting this form of plagiarism compared to what presumably would be the simplest to detect, *no obfuscation*.

4.2 Seed Classification

Testing the seeding step is a difficult test due to the nature of the data set. It is very difficult to correctly classify all seeds due to the way the plagiarism is labeled. Some of the seeds within the marked plagiarism area is not really plagiarism as it is not the correct sentence pair. This means that a lot of cases of non-plagiarism should be classified as plagiarism and causes a great deal of confusion for a machine learning algorithm, and could lead to low precision. The noise problem has been attempted mitigated using a heuristic in combination with the labeling in order to provide a cleaner data set for the classifier.

Classification results are interesting but do not provide a full picture seeing as these data have been tampered with. No plagiarism fragments have been added, but quite a few have been taken away. Some may therefore be missing from these results. However, these numbers will however be interesting to compare to character, case and document level performance

The interesting number in this experiment is the precision throughout the levels. Passing this directly on to XML output and PAN performance tester will not give results that shows its true efficiency. This part of the pipeline is designed to extract the key points of plagiarism. It will not return the entire *rectangle* of plagiarism, meaning that a low recall may in reality mean a far higher recall in terms of characters.

Testing the true performance requires the seed classification to be run through

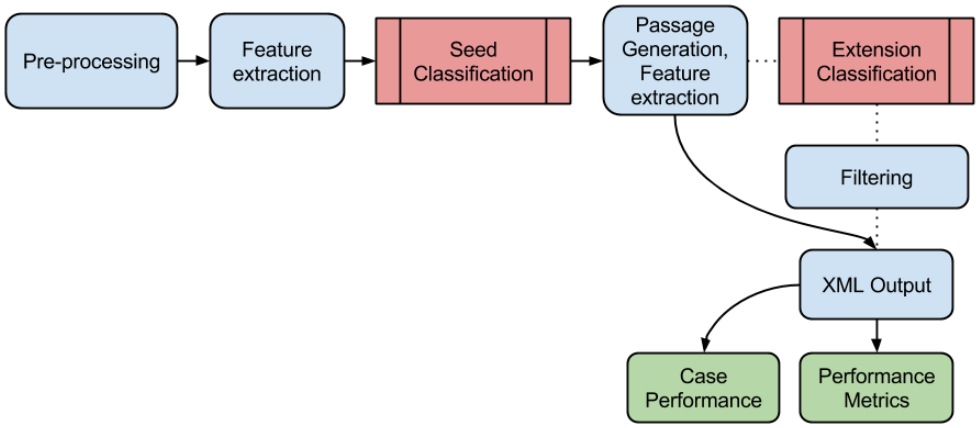

Figure 4.2: Active modules during seeding classification tests

the passage generation step, and assuming that all cases passed through the extension step are plagiarism cases. Filtering will be bypassed and the result post-extension passed directly to XML output and performance testing. This requires a bug-free passage generation step, which of course can not be guaranteed. These two steps are very intertwined and can not be tested efficiently independent of each other.

The tests will be run with all classifiers showing classification, character, case and document level performance.

4.2.1 Naïve Bayes

IsPlag	Recall	Precision	F_1
False	.991	.998	.994
True	.950	.794	.865

Table 4.2: Naïve Bayes seed performance on seeds

Table 4.2 shows great promise in terms of recall but does not show an optimal precision. Also note that this table shows the accuracy in discovering the plagiarised seed processed with a heuristic. This does not represent all the seeds within the plagiarised passages. It does, however, suggest that the heuristic is easily detectable and can stand out from the non-plagiarised passages. However, it is far from optimal and may suggest that the data set is still somewhat noisy. The heuristic may not have been optimal.

	Char Rec	Char Prec	Granularity	PlagDet
No obfuscation	.961	.564	1.0	.710
Random	.919	.651	1.0	.762
Translation	.938	.628	1.0	.753
Summary	.552	.559	1.0	.555
Overall	.917	.594	1.0	.721

Table 4.3: Naïve Bayes character performance

	Case Rec	Case Prec	F_1	Doc Rec	Doc Prec	F_1
No obfuscation	.963	.963	.963	.982	.982	.982
Random	.919	.971	.944	.946	.986	.966
Translation	.945	.968	.957	.969	.984	.976
Summary	.417	.263	.323	.667	.421	.516
Overall	.912	.886	.899	.949	.915	.932

Table 4.4: Naïve Bayes case and document performance

Tables 4.3 and 4.4 show that naïve Bayes is quite adept at detecting plagiarism in terms of recall with very high scores. However, the precision is lacking, resulting in a poor overall PlagDet score of 0.721. This is not so apparent in the case and document scores, as seen in table 4.4, where precision is far higher, with the exception of summary. With an F_1 case score of 0.899, this is a very decent result without any extension.

This also shows that the heuristic used to clean the data set seems fairly efficient, but not optimal. The data set may not be possible to fully clean up using heuristics. It may also be possible that the feature set for naïve Bayes is not optimal or that the algorithm is not the right algorithm for this task.

4.2.2 Decision Tree

IsPlag	Recall	Precision	F_1
False	.998	.996	.997
True	.892	.935	.913

Table 4.5: Decision tree seed performance on seeds

Table 4.5 shows that the decision tree seed classifier performs better in terms of plagiarism seed precision than naïve Bayes, at some expense of recall.

	Char Rec	Char Prec	Granularity	PlagDet
No obfuscation	.957	.764	1.0	.850
Random	.858	.820	1.0	.838
Translation	.860	.796	1.0	.826
Summary	.430	.956	1.0	.593
Overall	.867	.801	1.0	.833

Table 4.6: Decision tree seed character performance

	Case Rec	Case Prec	F_1	Doc Rec	Doc Prec	F_1
No obfuscation	.949	.956	.952	.971	.978	.974
Random	.829	.902	.863	.892	.952	.921
Translation	.836	.915	.873	.895	.958	.925
Summary	.250	.261	.255	.583	.609	.596
Overall	.837	.884	.860	.901	.940	.920

Table 4.7: Decision tree seed case and document performance

The decision tree seems to have a slightly lower recall but a higher accuracy than naïve Bayes, resulting in a higher PlagDet score, as can bee seen in tables 4.6 and 4.7. However the overall case F_1 score is lower than with naïve Bayes. Also note the substantial character precision jump in *summary* obfuscation. The case recall is worse, resulting in naïve Bayes having a higher F_1 score on a case level, with decision tree marginally beating it on a character level.

The seed classification and character, case, and document level classifications seem to match up fairly well. This suggests that the data set may have been cleaned up sufficiently despite previously observed low precision by naïve Bayes. Decision trees may be more suitable for this task.

4.2.3 Random Forest

IsPlag	Recall	Precision	F_1
False	.997	.997	.997
True	.931	.917	.924

Table 4.8: Random forest seed performance on seeds

Table 4.8 shows great promise. All scores are over 0.9. A high level of recall as well as a high level of precision results in an F_1 of 0.924 in terms of seeding.

	Char Rec	Char Prec	Granularity	PlagDet
No obfuscation	.970	.741	1.0	.840
Random	.905	.825	1.0	.863
Translation	.912	.737	1.0	.815
Summary	.528	.999	1.0	.691
Overall	.907	.779	1.0	.838

Table 4.9: Random forest seed character performance

Table 4.9 tells a slightly different story. The precision here is slightly lower than what is suggested by table 4.8. The PlagDet score on *summary* obfuscation is higher than any entrant in PAN without any extension algorithm applied. The overall PlagDet score is among the top scores.

	Case Rec	Case Prec	F_1	Doc Rec	Doc Prec	F_1
No obfuscation	.985	.993	.989	.989	.992	.989
Random	.919	.962	.940	.923	.962	.942
Translation	.922	.967	.944	.926	.967	.946
Summary	.333	.364	.348	.333	.364	.348
Overall	.907	.938	.922	.911	.938	.924

Table 4.10: Random forest seed case and document performance

The case and document performance in table 4.10 is better overall than what has been demonstrated this far. A case F_1 score of 0.922 overall is a very high score, which already at the seeding stage is rivaling the top PAN entrants in terms of case detection. The same is true for document detection.

Random forest seeding will be used as the seeding classification algorithm due to the high overall accuracy on character, case and document levels.

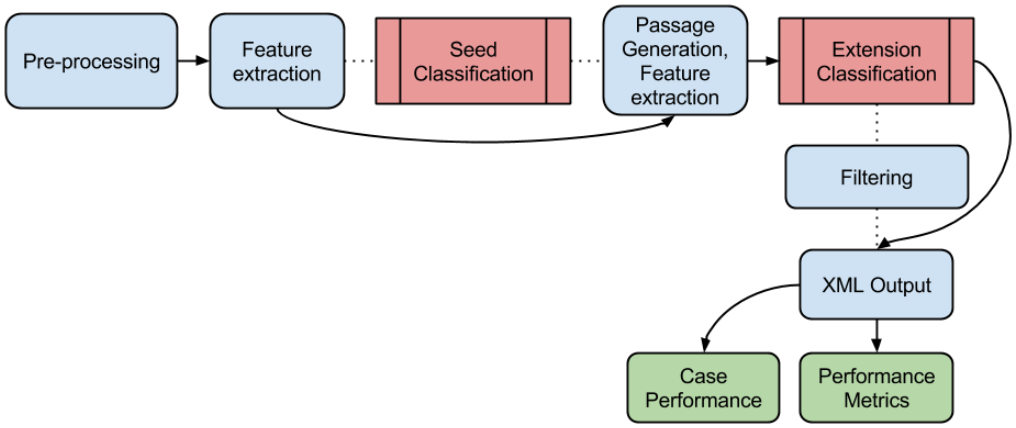

Figure 4.3: Active modules during extension tests

4.3 Extension Classification

Testing the extension step can be done in two ways. The first method would be creating a confusion matrix on the input from the seeding step and calculating recall and precision from this data as well as continuing to test with the PAN performance tool. These numbers will each provide valuable data about the performance of this step. Basing the test on the seeding data may not provide a complete picture, but at most a *best case* scenario, as some seeds may be missing, resulting in false recall and precision for this step. A second method would avoid this problem: the seeding data could be created based directly of the data set, ensuring an absolutely correct seeding algorithm. This would result in a very poor test as well, since the classifier would not have any non-plagiarised passages to train itself on. It would in this case either refuse to run due to a single classification class, or simply call all seeds plagiarism, resulting in a perfect score regardless.

The test is therefore performed on the output from the best seeding test previously performed, providing the *best case* scenario using real data. As with seeding, classification, character, case and document performance will be listed.

4.3.1 Naïve Bayes

IsPlag	Recall	Precision	F_1
False	.744	.793	.768
True	.911	.887	.899

Table 4.11: Naïve Bayes extension performance

Do note that table 4.11 does not show true extension performance. If seeds in plagiarised passages have been missed in the seeding stage, they are not passed on to the extension stage. These data are based on the passages passed to the extension classifier after passage creation. Nevertheless it shows how the extension classifier performs with the data provided from the seeding step. Naïve Bayes struggles to extract non-plagiarism. It is also not very good in terms of plagiarism passage detection.

	Char Rec	Char Prec	Granularity	PlagDet
No obfuscation	.946	.754	1.0	.840
Random	.889	.829	1.0	.858
Translation	.846	.763	1.0	.802
Summary	.167	1.0	1.0	.286
Overall	.851	.783	1.0	.816

Table 4.12: Naïve Bayes extension character performance

	Case Rec	Case Prec	F_1	Doc Rec	Doc Prec	F_1
No obfuscation	.956	.985	.970	.967	.992	.980
Random	.901	.962	.930	.923	.981	.951
Translation	.851	.948	.897	.891	.974	.931
Summary	.083	.250	.125	.208	.625	.313
Overall	.855	.947	.899	.885	.972	.926

Table 4.13: Naïve Bayes extension case and document performance

In terms of character, case and document performance, the results are fairly good, as seen in tables 4.12 and 4.13. However it seems to struggle greatly with summary detection. Very few cases have been discovered. This does, however, lead to a very high precision for this obfuscation strategy.

4.3.2 Decision Tree

IsPlag	Recall	Precision	F_1
False	.689	.912	.785
True	.970	.872	.918

Table 4.14: Decision tree extension performance

Decision tree seems to be able to detect plagiarism slightly better than naïve Bayes. With an F_1 score of 0.918, as seen in table 4.14, compared to naïve Bayes' 0.899, this should yield better character, case and document performance.

	Char Rec	Char Prec	Granularity	PlagDet
No obfuscation	.970	.741	1.0	.840
Random	.905	.825	1.0	.863
Translation	.920	.740	1.0	.820
Summary	.528	.999	1.0	.691
Overall	.909	.779	1.0	.839

Table 4.15: Decision tree extension character performance

	Case Rec	Case Prec	F_1	Doc Rec	Doc Prec	F_1
No obfuscation	.985	.993	.989	.989	.993	.990
Random	.919	.962	.940	.923	.962	.943
Translation	.930	.967	.948	.934	.968	.950
Summary	.333	.363	.347	.333	.364	.348
Overall	.909	.938	.924	.913	.938	.926

Table 4.16: Decision tree extension case and document performance

Tables 4.15 and 4.16 show a better performance than naïve Bayes. Looking at *summary*, the recall is far higher with precision barely affected. With a PlagDet score of 0.839, F_1 case and document score of 0.924 and 0.926 respectively, these are very good results. The *summary* PlagDet score of 0.691 is higher than any of the PAN entrants, which is demonstrated later.

A very interesting result here is that these two tables are near identical to those of the best seeding algorithm, random forest, as seen in tables 4.9 and 4.10. It seems that decision tree extension classification does not take away or add much to the result. The reason for this could lie in the way the seeds are labeled and extension passages are created. As explained in section 3.4, the passages are labeled as plagiarism if more than half of the passage is within a plagiarised area. After studying the output from the extension classification algorithm, it became clear that the false positives were actually surrounding plagiarised areas but had

a plagiarism coverage of less than half of the detected passage, leading to little change in plagiarism detection results for these cases.

In the event of false negatives, these were covered by other true positives, causing them to often override the bad classification. True negatives tended to be very short.

4.3.3 Random Forest

IsPlag	Recall	Precision	F_1
False	.972	.446	.612
True	.451	.973	.616

Table 4.17: Random forest extension performance

The results in table 4.17 are rather surprising. Random forests outperformed decision trees during development testing on all stages, and as well as seeding after development completion. With such a low recall and F_1 score, character, case, and document performance is not expected to surpass decision trees.

	Char Rec	Char Prec	Granularity	PlagDet
No obfuscation	.826	.847	1.0	.837
Random	.254	.978	1.0	.404
Translation	.197	.933	1.0	.325
Summary	.006	1.0	1.0	.013
Overall	.416	.890	1.0	.567

Table 4.18: Random forest extension character performance

	Case Rec	Case Prec	F_1	Doc Rec	Doc Prec	F_1
No obfuscation	.831	.966	.893	860	.983	.918
Random	.234	.839	.366	.275	.924	.424
Translation	.195	.862	.318	.242	.939	.385
Summary	0	0	N/A	0	0	N/A
Overall	.411	.921	.568	.449	.962	.612

Table 4.19: Random forest extension case and document performance

Tables 4.18 and 4.19 confirm the findings in table 4.17. Random forest is not suitable for extension classification in this thesis. It was unable to detect any *summary* cases and struggles greatly with all obfuscation strategies besides *no obfuscation*. This results in the very low PlagDet score of 0.567.

4.4 Sliding Window Filtering

The filtering test is designed to test the efficiency of the sliding window algorithm described in 3.5. The goal of this algorithm is to increase precision without great cost to recall. A simple way to test this is to observe the change in recall and precision running a complete detector test on the previous optimal setups, with and without the filtering algorithm.

This test is run with the complete system as the filtering algorithm is designed to work with an imperfect set of data. If the correct data was to be sent through the filtering system, the performance results would likely be misleading. This test is meant to see what the algorithm brings to a fully operational detector.

The best seed/extension setup prior to filtering is using random forest for seeding and decision tree for extension. These classifiers will be used for this experiment.

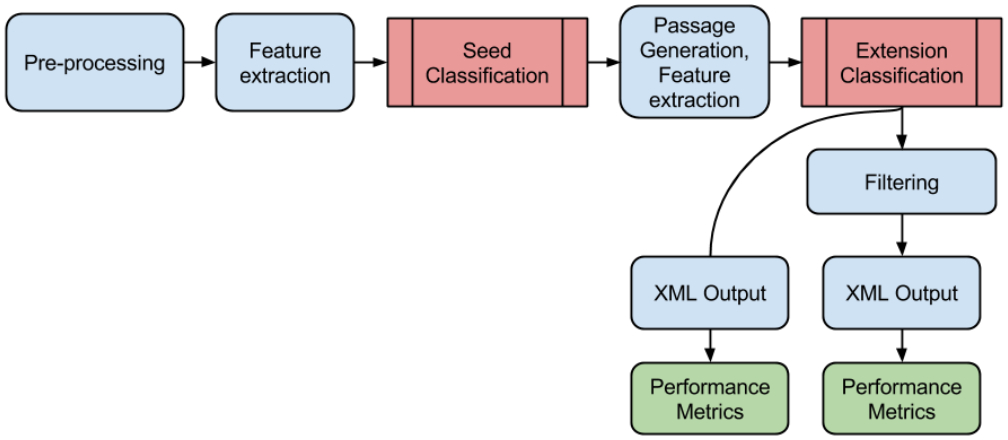

Figure 4.4: Active modules during filtering tests

	Char Rec	Char Prec	Granularity	PlagDet
No obfuscation	.910	.762	1.0	.829
Random	.854	.833	1.0	.844
Translation	.845	.751	1.0	.795
Summary	.522	.999	1.0	.686
Overall	.850	.793	1.0	.820

Table 4.20: Random forest seeding, decision tree extension with filtering character performance

	Case Rec	Case Prec	F_1	Doc Rec	Doc Prec	F_1
No obfuscation	.890	.996	.893	.945	.847	.947
Random	.847	.887	.866	.905	.945	.924
Translation	.836	.877	.856	.902	.929	.920
Summary	.333	.364	.348	.333	.364	.348
Overall	.827	.855	.841	.901	.927	.914

Table 4.21: Random forest seeding, decision tree extension with filtering case and document performance

As seen in table 4.20, the precision has increased, but at the expense of recall. This causes a decrease in PlagDet scores across all obfuscation strategies. In terms of case and document precision, this is lowered overall, as seen when comparing tables 4.21 and 4.20 to tables 4.154.16. This means that the sliding window algorithm does not add to the algorithm and is not a part of the optimal solution.

There may be parts of this algorithm that were misunderstood as the details surrounding it were not published. Some parameters may also be tweaked to improve performance. As of now it stops if the similarity metrics are lower. It may easily hit local minimums. In addition, a sliding window that increases the window size may also be tested in future research, as the recall seemed to lower due to the shrinking window.

4.5 Optimal Detector

	PlagDet				
Detector	None	Random	Translation	Summary	Overall
NB Seed	.710	.762	.753	.555	.721
DT Seed	**.850**	.838	**.826**	.593	.833
RF Seed	.840	**.863**	.815	**.691**	.838
NB Ext	.840	.858	.802	.286	.816
DT Ext	.840	**.863**	.820	**.691**	**.839**
RF Ext	.837	.404	.325	.013	.567
Sliding Window	.829	.844	.795	.686	.820
Optimal	.840	**.863**	.820	**.691**	**.839**

Table 4.22: Thesis methodology comparisons with respect to PlagDet. Column winners in bold

	Case Level			Document Level		
Detector	Rec	Prec	F_1	Rec	Prec	F_1
NB Seed	**.912**	.886	.899	**.949**	.915	**.932**
DT Seed	.837	.884	.860	.901	.940	.920
RF Seed	.907	.938	.922	.911	.938	.924
NB Ext	.855	**.947**	.899	.885	**.972**	.926
DT Ext	.909	.938	**.924**	.913	.938	.926
RF Ext	.411	.921	.568	.449	.962	.612
Sliding Window	.827	.855	.841	.901	.927	.914
Optimal	.909	.938	**.924**	.913	.938	.926

Table 4.23: Thesis methodology comparisons with respect to case and document performance. Column winners in bold

Tables 4.22 and 4.23 summarise and compare all the combinations of classifiers and algorithms previously tested. The winner in terms of overall PlagDet is the combination of random forest seeding with decision tree extension and no filtering. The extension classification barely increases the PlagDet score and the case level F_1 score. Decision tree and random forest are very close in the seeding stage but random forest struggles quite a bit with extension. It is interesting to note that naïve Bayes seeding has the best F_1 score on a document level, and fairly closely follows the optimal algorithm in terms of case level F_1. The substantial difference between case levels and PlagDet scores suggests that the methodology is capable of detecting cases but has a difficulty finding the start and/or end points of these cases. PAN entrants have a notably higher overall precision.

Despite a less than optimal data set, the methodology demonstrated here is clearly able to detect plagiarism at a high level. With a data set more suited to detecting seeds, an even higher detection level should be possible.

4.6 PAN Comparison

		Obfuscation Strategies				
Submission	Year	None	Random	Translation	Summary	Overall Corpus
Sanchez-Perez	2014	0.900	**0.884**	**0.887**	0.561	**0.878**
Oberreuter	2014	0.920	0.868	0.881	0.368	0.869
Palkovskii	2014	0.960	0.865	0.858	0.276	0.868
Glinos	2014	**0.962**	0.806	0.847	0.624	0.859
Shresta	2014	0.892	0.866	0.844	0.156	0.844
Kalleberg	2015	0.840	0.863	0.820	**0.691**	0.839
Kong	2012	0.872	0.832	0.852	0.436	0.837
Torrejón	2014	0.932	0.754	0.859	0.353	0.830
Oberreuter	2012	0.942	0.750	0.846	0.132	0.827
Gross	2014	0.900	0.803	0.838	0.319	0.826
Torrejón	2013	0.926	0.747	0.851	0.341	0.822
Kong	2014	0.838	0.823	0.852	0.431	0.822
Kong	2013	0.827	0.823	0.852	0.434	0.819
Palkovskii	2012	0.882	0.797	0.740	0.275	0.792
Torrejón	2012	0.882	0.702	0.801	0.442	0.788
Suchomel	2013	0.818	0.753	0.675	0.610	0.745
Suchomel	2012	0.898	0.652	0.631	0.501	0.732

Table 4.24: Thesis comparison against top 16 PAN entrants with respect to PlagDet. Top obfuscation scores in bold

Table 4.24 shows the PlagDet score of the best methodology in this thesis against the top 16 PAN entrants. Overall, it ranks 6th in the cross-year comparison on the latest data set. It stands out somewhat, as the *no obfuscation* score is a fair bit lower than that of the nearest neighbours in either direction. In terms of *summary* obfuscation it ranks first, which pulls it up a fair bit overall. The variance in PlagDet score is lower than that of the PAN entrants. At some cost of *no ob-*

fuscation detection, the *summary* detection is noticeably higher than the average and a few points ahead of the runner up in *summary*, Glinos at 0.624. A 10% improvement over the top entrant in the obfuscation strategy that is most difficult to detect is a very good result.

		Obfuscation Strategies				
Submission	Year	None	Random	Translation	Summary	Overall Corpus
Kalleberg	2015	0.970	**0.905**	**0.920**	0.528	**0.909**
Sanchez-Perez	2014	0.979	0.861	0.890	0.413	0.879
Oberreuter	2014	**0.999**	0.833	0.863	0.245	0.858
Shresta	2014	0.974	0.832	0.853	0.089	0.838
Palkovskii	2014	0.964	0.822	0.820	0.177	0.826
Kong	2012	0.948	0.780	0.850	0.299	0.824
Kong	2013	0.907	0.787	0.846	0.300	0.843
Kong	2014	0.895	0.781	0.845	0.296	0.807
Glinos	2014	0.960	0.725	0.762	0.486	0.793
Saremi	2013	0.954	0.689	0.804	0.102	0.771
Torrejón	2014	0.967	0.630	0.821	0.231	0.769
Oberreuter	2012	0.999	0.653	0.796	0.071	0.769
Gross	2014	0.907	0.719	0.784	0.206	0.766
Suchomel	2013	0.996	0.689	0.666	**0.563**	0.766
Torrejón	2013	0.953	0.634	0.811	0.216	0.762
Palkovskii	2012	0.994	0.751	0.667	0.169	0.762
Torrejón	2012	0.964	0.623	0.791	0.290	0.753

Table 4.25: Thesis comparison against top 16 PAN entrants with respect to character recall. Top obfuscation scores in bold

By a small margin, the methodology of this thesis has the highest overall recall, as seen in table 4.25. Even though the PlagDet score was the highest in *summary*, it does not lead this obfuscation strategy in recall. Suchomel 2013 has a

slightly higher recall. However, with a precision of 0.874 in *summary*, Suchomel has a lower PlagDet score for this obfuscation strategy, as table 4.24 shows.

In terms of character precision, the methodology is 3rd last out of the 29 entrants. Far below any of the top entrants. This is what caused the 6th place in PlagDet, despite a higher recall. Pre-processing showed a problem with precision and could be a potential culprit to the lowered precision. However, since Sanchez-Perez used the same methodology, this may not be the case. Their methodology may have focused more on precision when executing their extension algorithm. The labeling of the data set may also be a culprit due to the noise potentially still present in the corpus. If these two issues are addressed, the precision may increase.

			Case Level			Document Level		
Entrant	Year	PlagDet	Prec	Rec	F_1	Prec	Rec	F_1
Sanchez-Perez	2014	0.88	0.90	0.91	0.90	0.92	**0.91**	0.91
Oberreuter	2014	0.87	0.84	0.89	0.87	0.89	0.89	0.89
Palkovskii	2014	0.87	0.90	0.85	0.87	0.90	0.84	0.87
Glinos	2014	0.86	0.90	0.83	0.87	0.93	0.88	0.91
Shresta	2014	0.84	0.91	0.85	0.88	**0.94**	0.85	0.89
Kalleberg	2015	0.84	**0.94**	**0.91**	**0.92**	**0.94**	**0.91**	**0.93**
Kong	2012	0.84	0.86	0.85	0.85	0.89	0.85	0.87
Gross	2013	0.83	0.90	0.86	0.88	0.93	0.85	0.89
Oberreuter	2012	0.83	0.81	0.79	0.80	0.83	0.80	0.81
Torrejón	2014	0.83	0.84	0.83	0.83	0.89	0.84	0.86
Torrejón	2013	0.83	0.83	0.83	0.83	0.87	0.84	0.85
Kong	2013	0.82	0.85	0.86	0.85	0.89	0.86	0.87
Kong	2014	0.82	0.86	0.85	0.85	0.89	0.85	0.87
Palkovskii	2012	0.79	0.80	0.80	0.80	0.82	0.80	0.81
Torrejón	2012	0.79	0.65	0.79	0.72	0.65	0.78	0.71
Suchomel	2013	0.74	0.66	0.83	0.73	0.67	0.82	0.74
Suchomel	2012	0.73	0.76	0.70	0.73	0.77	0.69	0.73

Table 4.26: Thesis comparison against top 16 PAN entrants with respect to case and document performance. Top column scores in bold

Table 4.26 shows something very interesting. The methodology of this thesis matches or surpasses the best entrants in all columns despite a lower PlagDet score. This indicates that the methodology is quite adept at catching plagiarism cases, but with its low precision, this suggests that too many neighbouring characters are plagiarism as well. Further work is suggested to remove this problem. Only 6% of the cases detected as plagiarism are not. 9% of the cases in the data set were not detected.

Chapter 5

Conclusion and Future Work

Section 1 provides a conclusion based on the previously shown results. Section 2 suggests what work that may follow from this thesis or what work that should be done in the field in general.

5.1 Conclusion

In this thesis a textual plagiarism detector has been designed using machine learning techniques. It has been compared with the current state of the art using a freely available data set and results from the *PAN Workshop and Competition: Uncovering Plagiarism, Authorship and Social Software Misuse*(PAN)[2]. The goal was to match the performance level of the entrants of PAN without the need to set manual thresholds based on empirical data set dependent research. By researching the leading PAN entrants' methodologies, a pipeline based on common approaches and techniques used in the current state of the art was developed. Novel features and processing methods were developed in addition to using or modifying known techniques in order to further enhance the performance of the methodology. After analysis of the state of the art, weaknesses of leading methodologies were ex-

posed and attempted corrected by adding these novel features. The data set used by PAN was shown not to be very suitable for machine learning purposes. Using a heuristic based on some extracted features, the data set was cleaned and made more usable.

Each part of the pipeline was tested meticulously in order to find the most suitable methodology for this task. By testing 3 common machine learning algorithms: naïve Bayes, decision tree and the decision tree extension random forest, a complete methodology was created to benchmark against the PAN data set.

With a 6th place in terms of PlagDet score overall, a 1st place in *summary* PlagDet score and 1st in overall case and document performance scores, the thesis results confirm the hypothesis: "*Machine learning algorithms are capable of similar, or better performance in terms of plagiarism detection than the current, manually optimized, state of the art methodologies in PAN 2012-2014*".

The random forest and decision tree algorithms have been shown to be capable of classifying plagiarism of all complexities with a similar level of performance when comparing to current techniques. Along with the novel passage extension algorithm and the novel features, the detector is able to classify plagiarism with *summary* obfuscation at a higher level of accuracy than any detector in the state of the art compared in this thesis, with a PlagDet score of 0.691. It is able to closely match the top PAN entrant in terms of PlagDet score ending in an overall 6th place out of 29 entrants with a PlagDet score of 0.839. It is better than all entrants in case and document level detection with F1 scores of 0.924 and 0.926 respectively. It has the highest overall character recall of all entrants with 0.909 but one of the worst precision scores with 0.779. The high case scores along with the slightly lower PlagDet score and low precision suggest that the methodology still has potential for improvement in detecting the start and end points of plagiarism passages.

5.2 Future Work

5.2.1 Seeding and Filtering

The low character precision score combined with the high case score suggest that too many seeds near the edge of passages are passed on to the extension stage. Novel features attempting to detect just this may increase the character performance for this methodology.

The sliding window algorithm proved inefficient in increasing precision.

There are methodologies in the PAN subtasks *author profiling*, and *author identification* which may be of interest in plagiarism detection. Both for text alignment and source retrieval. This step is a natural next step for this thesis in order to design a full system that includes source retrieval.

By analysing changes throughout the document in terms of writing styles, vocabulary, and punctuation usage, plagiarised areas may be suggested without the need for a source document. Several such features have been used by entrants in these subtasks[4, 27]. Features like these may aid in source retrieval by suggesting what passages of a text that should be analyzed further and have potential sources retrieved for.

5.2.2 Suggestions to PAN

As this thesis shows that machine learning is fully capable of detecting plagiarism despite sub optimal data sets, it is suggested that future PAN workshops generate data sets where plagiarism information is available on a finer level than long passages. This would open up more for machine learning for this specific task.

Machine learning techniques are already being used extensively in the PAN subtask of author profiling. Most likely due to a more fitting data set.

5.2.3 Future Publications and PAN Entry

The methodology in this thesis may possibly be refined and entered into PAN 2015 or 2016 workshop with corresponding papers. This would also likely include a source retrieval entry, in order to create modules in a complete plagiarism detection solution. This could also at a later date be joined with other ideas of the author for a commercial system that could aid educational institutions in limiting and detecting cheating of several kinds in exams and hand-ins.

Bibliography

[1] M. Sanchez-Perez, G. Sidorov, and A. Gelbukh, "A winning approach to text alignment for text reuse detection at pan 2014," *Notebook for PAN at CLEF*, pp. 1004–1011, 2014. [Online]. Available: http://www.uni-weimar.de/medien/webis/research/events/pan-14/pan14-papers-final/pan14-plagiarism-detection/sanchezperez14-notebook.pdf

[2] M. Potthast, M. Hagen, M. Tippmann, P. Rosso, Beyer, Anna, Stein, Benno, and Busse, Matthias, "Overview of the 6th International Competition on Plagiarism Detection." 2014. [Online]. Available: http://www.uni-weimar.de/medien/webis/publications/papers/stein_2014k.pdf

[3] P. Baldi and S. Brunak, *Bioinformatics: The Machine Learning Approach*. MIT Press, 2001.

[4] F. Rangel, P. Rosso, I. Chugur, M. Potthast, M. Trenkmann, B. Stein, B. Verhoeven, and W. Daelemans, "Overview of the 2nd Author Profiling Task at PAN 2014." [Online]. Available: http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-RangelEt2014.pdf

[5] M. Potthast, M. Hagen, M. Vlske, and B. Stein, "Crowdsourcing Interaction Logs to Understand Text Reuse from the Web." in *ACL (1)*, 2013, pp. 1212–1221. [Online]. Available: http://www.aclweb.org/anthology/P13-1119

[6] M. Potthast, B. Stein, A. Barrn-Cedeo, and P. Rosso, "An evaluation framework for plagiarism detection," in *Proceedings of the 23rd international conference on computational linguistics: Posters*. Association for Computational Linguistics, 2010, pp. 997–1005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1944681

[7] D. Glinos, "Discovering Similar Passages within Large Text Documents," in *Information Access Evaluation. Multilinguality, Multimodality, and Interaction*, ser. Lecture Notes in Computer Science, E. Kanoulas, M. Lupu, P. Clough, M. Sanderson, M. Hall, A. Hanbury, and E. Toms, Eds. Springer International Publishing, Sep. 2014, no. 8685, pp. 98–109. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-319-11382-1_10

[8] Y. Palkovskii and A. Belov, "Developing High-Resolution Universal Multi-Type N-Gram Plagiarism Detector." [Online]. Available: http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-PalkovskiiEt2014.pdf

[9] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980. [Online]. Available: http://www.emeraldinsight.com/doi/abs/10.1108/eb046814

[10] K. Leilei, Q. Haoliang, W. Shuai, D. Cuixia, W. Suhong, and H. Yong, "Approaches for candidate document retrieval and detailed comparison of plagiarism detection," *Notebook for PAN at CLEF 2012*, 2012. [Online]. Available: http://www.uni-weimar.de/medien/webis/research/events/pan-12/pan12-papers-final/pan12-plagiarism-detection/kong12-notebook.pdf

[11] D. A. R. Torrejón and J. M. M. Ramos, "Text Alignment Module in CoReMo 2.1 Plagiarism Detector." [Online]. Available: http://ims-sites.dei.unipd.it/documents/71612/430938/CLEF2013wn-PAN-Rodr%C3%ADguezTorrej%C3%B3nEt2013.pdf

[12] K. Leilei, Q. Haoliang, D. Cuixia, W. Mingxing, and H. Zhongyuan, "Approaches for Source Retrieval and Text Alignment of Plagiarism Detection," *Notebook for PAN at CLEF 2013*, 2013. [Online]. Available: http://ceur-ws.org/Vol-1179/CLEF2013wn-PAN-LeileiEt2013.pdf

[13] Transatlantic, "Overture - The Whirlwind."

[14] D. Glinos, "A Hybrid Architecture for Plagiarism Detection." [Online]. Available: http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-Glinos2014.pdf

[15] D. A. R. Torrejón and J. M. M. Ramos, "CoReMo 2.3 Plagiarism Detector Text Alignment Module." [Online]. Available: http://www.uni-weimar.de/medien/webis/research/events/pan-14/pan14-papers-final/pan14-plagiarism-detection/rodrigueztorrejon14-notebook.pdf

[16] P. Gross and P. Modaresi, "Plagiarism Alignment Detection by Merging Context Seeds." [Online]. Available: http://www.uni-weimar.de/medien/webis/research/events/pan-14/pan14-papers-final/pan14-plagiarism-detection/gross14-notebook.pdf

[17] Š. Suchomel, J. Kasprzak, M. Brandejs *et al.*, "Diverse queries and feature type selection for plagiarism discovery," *Notebook for PAN at CLEF 2013*, 2013.

[18] D. A. R. Torrejón and J. M. M. Ramos, "Detailed comparison module in coremo 1.9 plagiarism detector." in *CLEF (Online Working Notes/Labs/-Workshop)*. Citeseer, 2012, pp. 1–8.

[19] Transatlantic, "Shine - Kaleidoscope."

[20] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022283681900875

[21] O. Gotoh, "An improved algorithm for matching biological sequences," *Journal of molecular biology*, vol. 162, no. 3, pp. 705–708, 1982. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022283682903989

[22] M. G. Snover, N. Madnani, B. Dorr, and R. Schwartz, "TER-Plus: paraphrase, semantic, and alignment enhancements to Translation Edit Rate," *Machine Translation*, vol. 23, no. 2-3, pp. 117–127, 2009. [Online]. Available: http://link.springer.com/article/10.1007/s10590-009-9062-9

[23] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos, "An evaluation of naive bayesian anti-spam filtering," *arXiv preprint cs/0006013*, 2000. [Online]. Available: http://arxiv.org/abs/cs/0006013

[24] N. Ye *et al.*, *The handbook of data mining*. Lawrence Erlbaum Associates, Publishers, 2003, vol. 24.

[25] U. Bandara and G. Wijayarathna, "A Machine Learning Based Tool for Source Code Plagiarism Detection," *International Journal of Machine Learning and Computing*, pp. 337–343, 2011. [Online].

Available: http://www.ijmlc.org/index.php?m=content&c=index&a=show&catid=25&id=242

[26] Kong Leilei, Qi Haoliang, Du Cuixia, Wang Mingxing, and Han Zhongyuan., "Approaches for Source Retrieval and Text Alignment of PlagiarismDetection," *Notebook for PAN at CLEF*. [Online]. Available: http://ceur-ws.org/Vol-1179/CLEF2013wn-PAN-LeileiEt2013.pdf

[27] E. Stamatatos, W. Daelemans, B. Verhoeven, M. Potthast, B. Stein, P. Juola, M. A. Sanchez-Perez, and A. Barrn-Cedeo, "Overview of the Author Identification Task at PAN 2014," *analysis*, vol. 13, p. 31, 2014. [Online]. Available: http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-StamatosEt2014.pdf