NTNU – Trondheim
Norwegian University of
Science and Technology

Authorship Identification of Research Papers

Simen Skoglund

Master of Science in Informatics
Submission date: August 2015
Supervisor: Kjetil Nørvåg, IDI

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

Abstract

Authorship identification is a technique used to identify anonymous documents by identifying and extracting an authors stylometric features. The focus of this thesis is to apply an authorship identification technique, classification, to a set of research papers to determine the authorship. We go through theory and previous work of authorship identification before we present the implemented system. In the end, we perform two separate experiments and discuss their results.

The experiments show good results in specific cases, and we achieve an accuracy of 100% in the best case. The algorithms used are support vector machines, artificial neural networks, decision trees, random forests and the k-nearest neighbor. In our experiments support vector machines and artificial neural network had the best performance while decision trees performed worst.

Based on our results we propose caution when applying authorship identification before or after having performed a double-blind review, or for an author to use authorship identification to acquire an unbiased review of a research paper. Even though we state that authorship identification should be used with caution, it is still a great tool and gives a general idea of finding the authorship of an anonymous document.

Sammendrag

Forfatter-identifisering er en teknikk brukt til å identifisere forfatteren av et anonymt dokument. Identifiseringen skjer ved å identifisere og å ta i bruk forfatterens stylometriske trekk. Hovedfokuset i denne masteroppgaven har vært å ta i bruk en forfatter-identifiseringsteknikk, kalt klassifisering, og å bruke denne teknikken til å identifisere forfatteren av anonyme forskningsartikler. I oppgaven går vi gjennom nødvendig teori og tidligere arbeid utført i dette forskningsfeltet. Videre presenterer vi det implementerte systemet i oppgaven, og til slutt utfører vi to separate eksperimenter og diskuterer de respektive resultatene.

Eksperimentene viste gode resultater på spesifikke områder, og på det beste oppnådde vi en nøyaktighet på 100%. Algoritmene brukt i oppgaven er som følger: support vector machines, artificial neural networks, decision trees, random forests og k-nearest neighbor. I våre eksperiment hadde bruken av support vector machines og artificial neural networks de beste resultatene, mens decision trees hadde de dårligste.

Basert på resultatene oppnådd i denne masteroppgaven vil vi advare mot å stole på forfatter-identifisering til enten å finne forfatteren etter en blind review, eller for en fortfatter til å bruke systemet til å kamuflere sin forskningsartikkel. Til tross for at vi advarer mot dette, er forfatter-identifisering fortsatt et verktøy som kan brukes i eksperimentelle situasjoner, og kan gi en pekepinn for å finne forfatteren av et anonymt dokument.

Preface

This thesis is written by Simen Skoglund, and is the final result of a master's thesis at the Norwegian University of Science and Technology. This masters's thesis is the conclusion of the two-year master's degree study in Informatics at the Department of Computer and Information Science, Faculty of Information Technology, Mathematics, and Electrical Engineering at NTNU.

I would like to thank Professor Kjetil Nørvåg, my supervisor, for being available, helpful, and for giving constructive feedback during the period of writing this thesis.

Contents

Chapter 1

Introduction

In this chapter, an introduction to the thesis is given. First through the motivation for the thesis, and why this is relevant research.

1.1 Motivation

The ability to be able to determine the authorship of anonymous texts is increasing rapidly in today's society. Whether it is in a criminal investigation or trying to determine the authorship of a research paper authorship identification has to be used to some degree even if someone has to sit down and perform the identification manually. Authorship identification is the ability to identify anonymous authors based on their previous work and statements. The common technique in authorship identification is to look at and identify characteristics by an author's stylometric pattern. For instance, if an individual has been active in public forums and blogs posting messages with their real name, or with other means that can act as identification and suddenly starts posting threatening messages anonymously. In this case authorship identification could have been used to identify the perpetrator based on their previous activity because these two individuals, even though they where one person all along, would have used the same stylometric pattern.
The main motivation for this thesis is the ability to detect the authorship of a research paper by using different classification algorithms and see how they perform.
When a new journal or research paper is up for a peer review [1] normally a double-blind [2] approach is used. The double-blind approach operates with the

[1] http://libguides.lib.siu.edu/peerreviewedjournals
[2] http://www.arj.no/2010/09/04/double-blind-peer-review/

assumption that the author does not know the identity of the reviewer, and the reviewer does not know the identity of the author. The reason to do a double blind review is for the journal or research paper to get an unbiased opinion on the work. The likelihood of a respected author to get a biased review of their work is why the blind/double-blind review is standard when a research paper is up for evaluation.
There are two sides to the ability to detect the authorship of a research paper. The first one is unethical and gives the reviewer a chance to find out who wrote the research paper. In this scenario, a reviewer can use authorship identification software and give the author a biased review. Alternatively, the less unethical approach; to see who wrote the paper after the review has been completed. The second side is the ethical way and gives the author a chance to stay anonymous and prevent the reviewer from giving a biased review. In this scenario, the author can use authorship identification software to check whether or not the author can be identified. If identified the author can alter the contents of the paper rendering the software unable to correctly identify the authorship and, therefore, be able to get an unbiased opinion on the work.

1.1.1 Early work in authorship identification

Authorship identification dates back to the middle of the 19th century when mathematician and logician Augustus De Morgan theorized that an author of a given text could be identified based on the number of longer words in a text. De Morgan's theory was put to the test in 1887 when T.C Mendenhall [27] experimented with the works of different authors. Mendenhall concluded that if you had about one hundred thousand words from the same author that authors curve of words would be uniquely different to another authors curve of words. Figure 1.1 shows two example curves each five thousand words long originating from the same author. The x-axis represents the number of letters in each word, and the y-axis represents the number of occurrences for each n-letter word.

1.1.2 Authorship identification using classification

Authorship identification using classification follows four very distinct steps and is what we have built the implemented system upon for this thesis. The first step when performing authorship identification is to collect data and construct a dataset. After the dataset is constructed, feature extraction is performed to extract stylometric features from the generated dataset. When the feature extraction process is completed, different classification algorithms can be applied

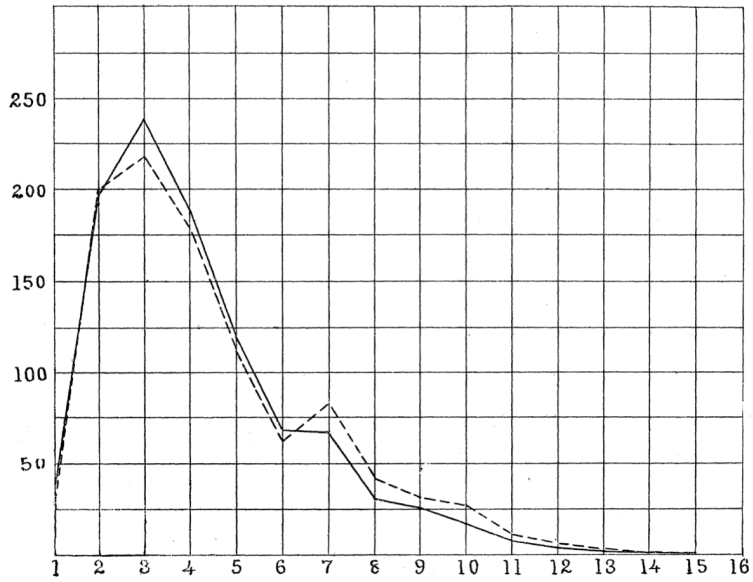

Figure 1.1: Mendenhalls example of a curve of words. [27]

to the feature set and display the author predictions. The entire process is is explained in Chapter 4 and the necessary theory is presented in Chapter 2.

1.2 Problem definition

From the motivation, the problem definition of this thesis will be to implement an authorship identification system based on previous research and use this system to identify the authorship of research papers. The systems goal is to be able to use several machine learning algorithm to be able to identify the authorship of a given research paper.

1.3 Research questions

From the problem definition, this thesis will answer the following research questions:

RQ1 What are the different features and techniques used in authorship identification field when looked at as a classification problem?

RQ2 Is it possible to perform authorship identification on research papers?

RQ3 If authorship identification of research papers is possible, does the amount of work contributed to a research paper by an author affect the degree of authors that can be identified?

To be able to answer these research questions, a survey on previous work related to authorship identification as a classification problem has been performed. A prototype has been developed to identify authors of research papers and we have tested out our system in two different experiments. The methods used in the implementation is explained in detail and evaluated as an author identification problem. The results have been discussed, and the experiments have been compared internally where ever it was possible, and the same experiments have been compared against each other.

To evaluate the different techniques used in the implementation, two separate datasets were created. The first dataset contains twenty different authors with between ten and twenty research papers per author divided in three different heuristics. The last dataset takes on a set of research papers presented at various information retrieval conferences and uses research papers prior to the year 2014 as training data and research papers after the year 2014 as test data.

1.4 Report outline

Chapter 2 introduces the necessary theory for understanding what classification is, which algorithms are going to be used and which metrics can be used to evaluate these algorithms.

Chapter 3 goes through the related fields of authorship identification, including authorship verification, plagiarism, authorship profiling and the PAN workshop.

Chapter 4 present the related work for this thesis and the techniques used in authorship identification, including how to perform data collection, feature extraction and model generation.

Chapter 5 gives a detailed explanation of the implemented system and talks about which choices had to be made and why.

Chapter 6 presents the experiments that have been done and their respective results.

Chapter 7 concludes this thesis and discusses the strengths and weaknesses as well as contributions and further work.

Chapter 2

Preliminaries

This chapter provides general information about machine learning, techniques and algorithms discussed in Chapter 4 to get a better understanding about how machine learning and the algorithms used in this thesis work. To learn more about classification, the algorithms or how to evaluate a classification model, please refer to the following literature [5] [28] [38] [41].

2.1 Classification

Classifications ultimate goal is to label an object to one of several possible categories. An example in classification is to detect whether or not an email is considered as spam. In this case, we have two categories, spam and not spam. The object to be labeled as spam or not spam is the email. This thesis looks at the possibility to label anonymous research papers to one out of several possible candidate authors. A classifier uses a training set to build a model that fits the relationship between the feature attributes and the category label. When the classifier is satisfiable trained a test set can be applied to the model to evaluate the performance of the classifier based on the number of correct and incorrect labels the classifier has predicted.

2.1.1 Feature set

In classification, each instance in a dataset is represented as a feature set, also known as a feature vector. Each feature in the feature set represents a unique property to that instance. For example, a feature for a research paper can be the total number of characters in that research paper. Features are often numeric values, but can also be other values such as booleans or strings. A feature set

is either used as a part of the training- or test dataset to help the classification algorithms be able to learn.

2.1.2 Training- and test data

Training data is data where the class label is known. For example, the features of an email that is labeled as spam can be used in the training dataset to help the classifier identify unlabeled emails as spam or not spam.
Test data is data where the class label is unknown. For example an email that has not been labeled as either spam or not spam.

2.1.3 Test- and training set

A set of objects where the class label is exposed is referred to as the training set. The same goes for the test set except the class label is unknown. For training and testing you can provide your own separate training set, and your separate test set that are independent of each other. However, there exist other techniques that can be used such as percentage split and cross-validation. These techniques use one part of the data as the training set and the remaining part as the test set.

2.1.4 Percentage split

The percentage split method is the easiest one, were a percentage of the objects are used as training data, and the rest is used for testing. For example, given six instances, and 67% of these are used as training data, the remaining 33% of the instances are used as test data. A drawback of the percentage split method is that the training and test set are no longer independent of each other.

2.1.5 K-fold cross-validation

In the cross-validation technique, each object are used exactly once for testing and each object are used the same number of times for training. For example if we have six objects split equally into three parts, each part would consist of a pair of objects. The first pair is then used as the test set, and the two other pairs are used as training set. The next iteration the first and the last pair are used for training and the middle set for testing and so on. This is known as K-fold cross-validation where K is the number of parts the initial dataset is split into. K must also be an integer greater than one and less than or equal the number of instances in the initial data set.

2.2 Evaluating the model

As mentioned in Section 2.1 during the training of a classifier a model is generated. This section provides the theory on how such a model can be evaluated based on performance and randomness. The specific classification algorithms are presented in Section 2.3.

2.2.1 True- and false positives & negatives

A true (TP) and false positive (FP) in classification is the number of instances that are correctly classified. A true positive is the number of classifications that the classifier classified correctly as the answer. A true negative is the opposite and is the amount of classifications that the classifier correctly predicted *not* to be the answer.
The true (TN) and false negatives (FN) are the incorrectly classified instances. A false positive happens when an incorrect instance is classified as correct. A false negative occurs when a correct instance is classified as incorrect.

2.2.2 Precision

Precision is the accuracy of how many relevant documents that are retrieved compared to the total amount of document retrieved. Precision is calculated as follows

$$\text{Precision} = \frac{\text{Relevant documents retrieved}}{\text{Total number of retrieved documents}} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.1}$$

2.2.3 Recall

Recall is defined as the number of relevant documents retrieved compared to total number of relevant documents that exists in the document collection. Recall is calculated as follow

$$\text{Recall} = \frac{\text{Relevant documents retrieved}}{\text{Total number of relevant documents}} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.2}$$

2.2.4 F-measure

Sometimes we are not interested in only recall or only precision but we want to combine the two because they are both important. The F-measure score does this by using the harmonic mean between precision and recall. There is also a possibility to add weights to either precision and recall based on how important

	a	b	c
a	1	4	1
b	3	6	1
c	4	2	2

Table 2.1: Realistic confusion matrix.

	a	b	c
a	6	0	0
b	0	10	0
c	0	0	8

Table 2.2: Ideal confusion matrix.

they are. In the equation below we can see how F-measure is calculated with equal weight on both precision and recall, also known as the balanced F1-measure.

$$\text{F-measure} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} = \frac{2 * \text{TP}}{2 * \text{TP} + \text{FP} + \text{FN}} \tag{2.3}$$

2.2.5 Confusion matrix

In classification, a confusion matrix tells something about how the classifier performed. The confusion matrix says something about how many instances were classified correctly, and how many were not. The rows represents the class an instance belongs to, and the columns represent the class predicted by the classification algorithm. Ideally in a confusion matrix we want the left-to-right diagonal to be the only cells that have non-zero values, and if achieved, the classifier has predicted every instance correctly. Table 2.2 represent an ideal confusion matrix as seen, only the diagonal contains non-zero values. The other matrix, shown in Table 2.1 represents a confusion matrix with only 9 out of 24 instances classified correctly and classes a and c heavily misclassified compared to class b.

2.2.6 Kappa statistics

When working with classification getting the success rate of the classifier is often not good enough. Let us say that for a given confusion matrix 65% of the instances are classified correctly. But how many of these instances are left to chance? With the Kappa statistics, also known as Cohen's Kappa, we take the instances classified correctly by chance into account and provides a more robust measure than the simple success rate. Equation 2.4 shows how to calculate the Kappa value from a confusion matrix. P(a) is the agreement between the classifier (predicted class) and the actual class. P(e) is the chance agreement.

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)} \tag{2.4}$$

A score between -1 and 1 is achievable when calculating Kappa, -1 equals perfect disagreement, 0 indicates that instances are classified by chance and 1 equals

perfect agreement. For values greater than 0 a Kappa score between 0 and 0.20 are considered poor agreement, between 0.20 and 0.40 are considered fair agreement, between 0.40 and 0.60 are considered moderate agreement, between 0.60 and 0.80 are good agreement, and between 0.8 and 1 are excellent agreement. Let us calculate the Kappa statistics for the confusion matrix shown in Table 2.1, since we already know that the overall success rate for this matrix is 37.5% we can assume that the Kappa is going to be fairly low, somewhere between poor and fair agreement. For class a the actual and predicted class agrees that 1 of the instances are correct, for class b the actual and predicted class agree that 6 of the instances are correct and for class c they agree that 2 of the instances are correct. The agreement between the predicted and actual class is $\Pr(a) = \frac{1+6+2}{24} = \frac{9}{24}$. To calculate the probability of chance agreement $\Pr(e)$ we have to compare the number of times the actual and predicted class chooses the classes a, b, c. The actual class chooses a 6 out of 24 times, b 10 out of 24 times and c 8 out of 24 times. The predicted class chooses a 8 out of 24 times, b 12 out of 24 times and c 4 out of 24 times. The probability that both the actual class and predicted class choose a by chance is $\frac{6}{24} * \frac{8}{24}$, b by chance is $\frac{10}{24} * \frac{12}{24}$ and c by chance is $\frac{8}{24} * \frac{4}{24}$. The overall probability by chance is the sum of all individual classes being chosen by chance and is expressed as $\Pr(e) = \frac{6}{24} * \frac{8}{24} + \frac{10}{24} * \frac{12}{24} + \frac{8}{24} * \frac{4}{24} = \frac{25}{72}$ Now we have calculated both $\Pr(a)$ and $\Pr(e)$ and can insert these values directly into Equation 2.4 and we get the Kappa score as seen in Equation 2.5.

$$\kappa = \frac{\frac{9}{24} - \frac{25}{72}}{1 - \frac{25}{72}} = 0,04 = 4\% \tag{2.5}$$

A Kappa score of 4% is considered very poor performance and only slightly better than what is considered to be random.

2.3 Algorithms

Classification can be performed with several different classification algorithms, in this section we describe the algorithms used in this thesis.

2.3.1 Support vector machines

In 1995, Cortes and Vapnik introduced an algorithm known as the support vector machine [9]. In its simplest form, a support vector machine is visualized in two-dimensional space with a dataset that consists of two different classes, e.i a square and a circle. These squares and circles are separated by a hyperplane, and since the support vector machine is two-dimensional, the hyperplane is represented as a one-dimensional line. Since there exists an infinite amount of hyperplanes that

can separate the circles and squares, the support vector machine is interested in the hyperplane that provides the maximal margin, also know as the maximal margin hyperplane. What this means is that we want to find a hyperplane that separates the two classes as far away from each other as possible. Figure 2.1 shows an example of the maximal margin hyperplane.

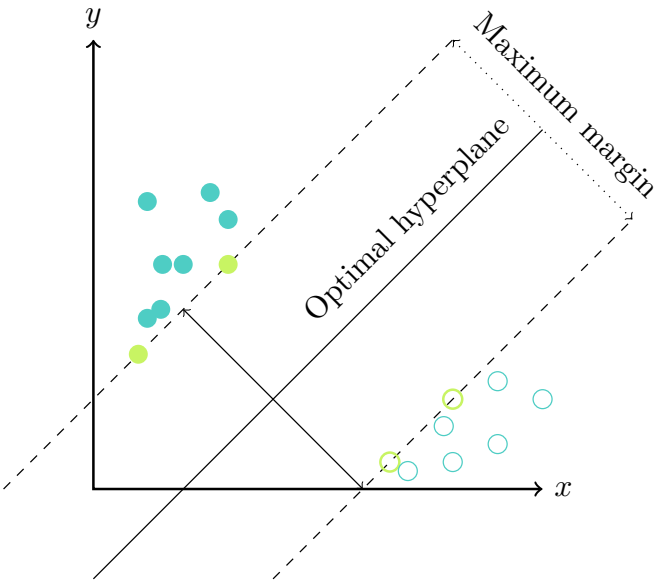

Figure 2.1: Example of an SVM.

When training the SVM classifier to build a model, any hyperplane found that separates the two data classes can have zero error after the training is completed. However, a model with a small hyperplane tend to perform poorly when classifying data from an unseen test set. Therefore, support vector machine models with large margin usually have better generalization errors than models with a smaller margin. There are also cases where the data points (squares and circles) cannot be separated by a straight line. If this happens to be the case, we can apply something called the kernel function. What the kernel function does is to project the data points to a higher dimension to be able to separate them. Support vector machines are also popular in authorship identification because the algorithm does not suffer from the curse of dimensionality problem.

2.3.2 Artificial neural networks

An artificial neural network, ANN for short, is a family of learning algorithms heavily inspired in its design to function much like the human brain. However, instead of neurons, axon, dendrites and synapses an ANN is composed of an

assembly of dependent nodes that are linked together in a network. The simplest ANN known is the perceptron-model, a perceptron uses two kind of nodes, the input nodes and an output node. The input nodes represent the input attributes and transmit the value to the outgoing link, the output node represent the model output and performs all the calculations. Like the human brain each node in an ANN is known as a neuron, and in the perceptron model the edges are known as the synaptic connection between the nodes. Each edge between the input node and the output node is given a weight, this is known as the weighted link and is used to emulate the synaptic strength between the nodes. Figure 2.2 illustrates how the perceptron-model may look like.

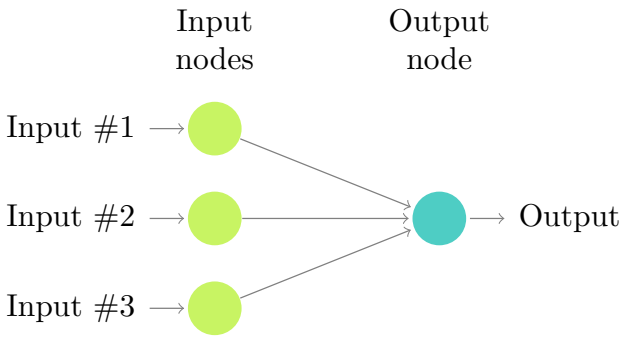

Figure 2.2: Example of the perceptron-model.

To calculate the output of the perceptron model a weighted sum operation is performed on the weights and attributes, and then a bias factor t is subtracted from the sum resulting in Equation 2.6. In the perceptron-model, a positive value will yield the value +1 and a negative value yields the value -1.

$$\text{Result} = \text{weight}_d\text{attribute}_d + \text{weight}_{d-1}\text{attribute}_{d-1} + ... + \text{weight}_1\text{attribute}_1 - t \tag{2.6}$$

Under training of the perceptron model the weights on the edges are adjusted until the outputs of the perceptron-model are consistent with the actual output from the training data. A drawback with the perceptron-model is that it only works on linearly separable classification problems, if a perceptron-model is used on a problem that is not linearly separable the perceptron-algorithm will never converge. For classification problems that are not linearly separable a more complex model has to be used. This model is known as the multilayered ANN and can have several hidden layers between the input node layer and the output node layer. And instead of being a feed-forward neural network, where nodes in one layer are only connected to nodes in the next layer, it can be recurrent, which means that nodes can be connected within the layer or be connected to a previous layer as well. As in the perceptron-model the input node layer is the attributes

to be evaluated, and the output node-layer represents the available classes from the training data. The hidden node-layer can, as the output node layer use an activation function to find a hyperplane and combine this with the output node to be able to produce output values that are non-linear. A figure showing a neural network with the hidden layer is shown in Figure 2.3.

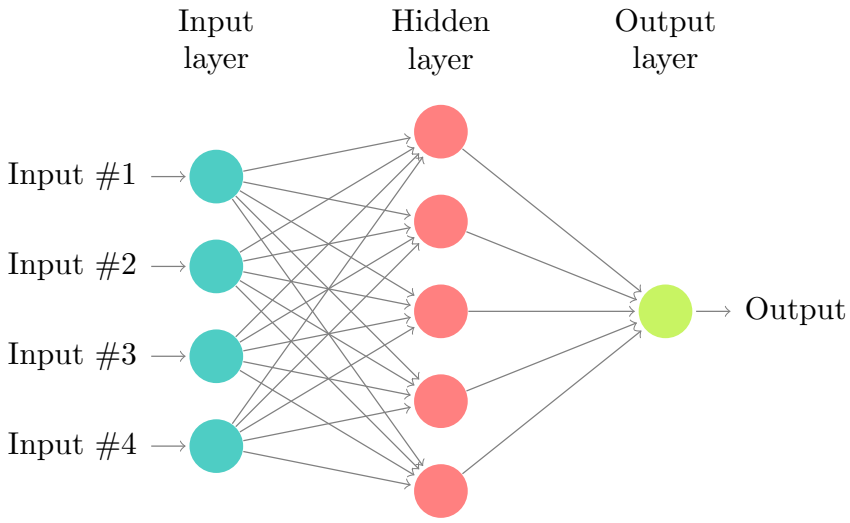

Figure 2.3: Example of multilayered perceptrons.

2.3.3 Decision trees

A decision tree [32] is a tree-like structure that is able to classify (i.e an author). Starting from the root node the decision tree proposes several choices depending on the number of values for each feature. Each value leads to a child node, also known as a chance node with a new feature. Already after the first choice several leaf nodes are unreachable from the decision tree and thus is eliminated as a possible candidate for classification. The child node chosen in the classification process has another feature similar to the root node, and again there is a choice between several values for the attribute. After reaching a leaf node (a node without children), the process is completed, and a prediction has been found (i.e. an author). A huge advantage with decision trees is that after they are generated (see the C4.5 algorithm) classification can be performed very fast. Figure 2.4 shows a possible decision tree in authorship identification.

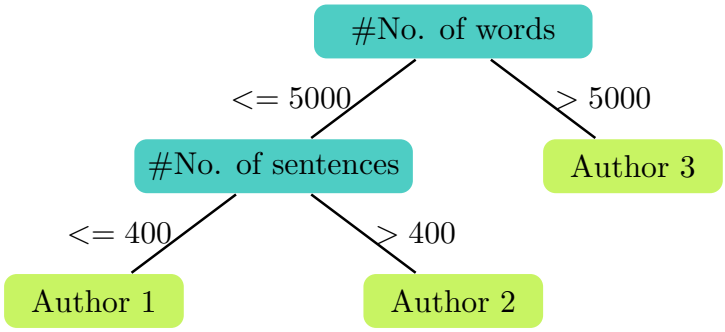

Figure 2.4: Example of a decision tree.

2.3.4 Random forests

Random forests combines decision trees and bootstrap aggregation (bagging) it was introduced in 2001 by Breiman [7]. Bagging is a technique in classification used to improve the accuracy of machine learning algorithms. Bagging is done by creating k bootstrap samples, train the classifier of your choice on these bootstrap samples and aggregate the result by taking a majority vote on the predictions made by each classifier. The bootstrap sample is a randomly chosen distribution of instances from the original training data. Each bootstrap sample is also of the same size as the original set, and because each instance in the sample is chosen at random, some data from the original training set can occur several times, and some data doesn't occur at all. What random forests does is to generate a set of B decision trees using bagging and at each node in the decision tree use a random subspace of the features for splitting. For a new instance to be classified the result is given by taking a majority vote on the result given by each decision tree. Going trough the entire procedure in more detail, first choose a B which is the number of decision trees in your random forest, for each tree choose a bootstrap sample from the training data. Construct the decision tree based on this bootstrap sample by using a random subspace of the features. A common number of features is $log_2(d) + 1$, and choose the best split for this node from the random subspace chosen. When all the B trees are generated and you want to classify a new instance of data x, input x to each decision tree and classify x based on a majority vote from the result for each decision tree. For example lets say you want to classify x, where x is either +1 or -1, four of nine decision tree in the random forest predicts -1, and the remaining five decision trees predict +1, x is classified as +1.

2.3.5 K-nearest neighbor

The k-nearest neighbor algorithm is one of the simplest classification algorithms that exists. Each instance of a training set is represented in a d-dimensional space where d is the number of attributes (or features) for each instance. For a new entry, i.e., from a test set, the position in the d-dimensional space is calculated. To classify this new instance the kNN-algorithm looks for the k nearest neighbors of d where k is the number of instances already represented in the d-dimensional space based on the training data. If the value of k is 3, then the algorithm checks the three nearest neighbors of the new instance and is classified according to the majority of instances that represents the same class. If there is a tie between the classes, a class may be chosen randomly to classify the new instance.

Chapter 3

Related Fields

This chapter aims to present fields that are related to the topic of this thesis, including a brief introduction to authorship identification.

3.1 Authorship analysis

Authorship analysis is defined as using an author's stylometric features to be able to conclude something about the authorship of a document. In 1997 Gray et al.[15] defined four different principal aspects of authorship analysis that they could use to determine the authorship of source code. Researchers responsible for the PAN workshop[1] use four different fields of authorship analysis that are presented below. These are *authorship identification*, also known as authorship attribution, *authorship verification*, *plagiarism detection* and *authorship categorization*.

3.2 Authorship identification

Authorship identification is to determine a probable author for an anonymous text. A detailed explanation can be seen in Chapter 4.

3.3 Authorship verification

In authorship verification, we are given a set of documents from an author along with an anonymous document. The goal in authorship verification is to be able

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

to tell whether or not the anonymous document were written by that author with a yes/no answer. Koppel & Schler[24] identified authorship verification as a one-class classification problem and they concluded that this problem is harder to solve than the problem of authorship identification. Koppel et al.[25] proposed in 2007 an efficient method on how to solve authorship verification known as the unmasking method. Since authorship verification is beyond the scope of this thesis we refer readers to Stein et al.[37], Kestemont et al. [21].

3.4 Plagiarism detection

Plagiarism is theft of someone else's work and in authorship analysis plagiarism detection is to find out if someone has stolen work he or she has published as his or her own. Maurer et al. [26] propose a list of six points that can be considered plagiarism. In research, plagiarism can be difficult to distinguish because most of the work is based on what other researchers have done before. A quote by Maurer et al.[26] describes this very well *"The border-line between plagiarism and research is surprisingly murky. After all, advanced research is only possible by "standing on the shoulders" of others, as it is often said." - Maurer et al.*. Also Mauer et al. propose three main categories for plagiarism detection, **document source comparison** is to pre-process the document in question and compare the similarity with other pre-processed documents via a search engine, e.i Google and highlight possible plagiarism parts. **Manual search of characteristic phrases** is another category which is to select a phrase and use this phrase as a query and search for matches on the internet. The last category is stylometry which is to use an author's writing style and comparing it within the document in question of previous work done by that author. The chances are that if, i.e., one paragraph of a paper does not fit that authors writing style this part may be questioned as plagiarism.

3.5 Authorship profiling

Authorship profiling is a technique used to find out information about an author given his or hers writing style. This type of authorship analysis is very similar to authorship identification in how it can be performed. For example to find out if an author is male or female can be considered a two-class classification problem. This relates to authorship identification between two authors. As in authorship identification, we can apply feature extraction to the text to generate a set of features and use machine learning to get a probable gender based on these features. Agramon et al.[3] used this technique to determine an author's age, gender, native language and neuroticism level. For authorship profiling, they

achieved 76.1% accuracy on classifying genders. 77.7% accuracy on classifying age divided into three classes(13-17, 23-27 and 33-47). 82.3% accuracy on their native language(Russian, Czech, Bulgarian, French and Spanish) and 65.7% accuracy on neuroticism (tendency to worry yes/no).

3.6 The PAN workshop

The PAN workshop is a bi-annually competition/workshop hosted at different conferences(SIGIR, CLEF, FIRE, SEPLN, and ECAI) all over the world. Participants in the PAN workshop can compete in three to four different categories in authorship analysis such as authorship identification, verification, profiling and plagiarism detection.

3.7 Summary

This chapter has introduced the reader to the field of authorship analysis and fields related to authorship identification.

Chapter 4

Authorship identification

This chapter illustrates the process of modern authorship identification by using classification. A general approach is presented along with previous work in the field alongside a more detailed presentation of authorship identification, including their data collection phase, feature extraction, model generation and results from 9 different papers featuring the authors of , Abbasi & Chen [1], Baraka et al. [6], Corney et al. [8], de Vel et al. [10], Hurtado et al. [18], Nizamani & Memon [30], Tas & Gor [39], Zhang et al. [42], and Zheng et al. [43]

4.1 Introduction

Modern authorship identification can be divided into two different areas. In the first area we have the similarity-based approach, this approach uses an author's writing style, also known as the stylometric features to determine the similarity between an anonymous text and an author. For example, one approach to this problem is to concatenate a large number of texts from an author and extract the stylometric features. These features are used to calculate the distance between the anonymous text and the different candidate authors. When the calculation is completed, a prediction can be made to determine the authorship of the anonymous text in question [36]. This approach has stood the test of time and has been in use since the mid-1960s when Mosteller and Wallace used bayesian statistical analysis on a set of small common words on the "Federalist Papers". The "Federalist Papers" is a collection of 85 articles and essays authored by Jay, Hamilton and Madison in 1964 [29]. Both Madison and Hamilton claimed authorship of twelve of these articles and by the work of Mosteller and Wallace, they credited Madison with the authorship of these. Later studies has also showed that the twelve articles were written by Madison, for example in [20] where Jockers and

Written applied several classification algorithms, including k-NN and support vector machines to the disputed papers.
The second area in authorship identification is the instance based approach and can be looked at as a multi-class single label text categorization task [35]. This approach uses training data to build a model using the extracted stylometric features as mentioned in Chapter 2. The model is generated by using a classification algorithm such as support vector machines, k-NN, random forests, artificial neural network or decision trees. A test set is then applied to the model to evaluate the performance of the classifier. In this thesis, we consider the second area and perform our authorship identification with the classification algorithms mentioned above.

4.2 Framework

The approach to authorship identification by classification has developed over the last few decades. However, the entire process can be boiled down to four distinct steps, also known as a framework [43]. The first step is known as the data collection phase. In this phase a corpus is created, this corpus is used as the data set for authorship identification. The second step is the feature extraction phase. In the feature extraction phase, a set of stylometric features defined by the people performing the authorship identification is used to extract data from the corpus established in the first step. Extracted data can be used as either training- or test data. The third step is the model generation phase. In the model generation phase, classification algorithms are used to generate a model based on the training data created in the previous step. Testing data is used to evaluate the performance of the model. The last step of the process is the authorship identification phase. In the authorship identification phase, an anonymous text is used as input to the generated model from the previous step and a prediction on the authorship represents the output. The entire process can be seen in Figure 4.1.

4.3 Document collection

Before the authorship identification process can begin enough data has to be collected. The researcher can choose whatever topic he or she would like to perform authorship identification on, imagination is the limit here. However choosing a specific area with a limited number of authors will most likely yield better results. Something to keep in mind is that some of the collected documents must specify the author to be used as training data. In previous studies due to the increasing amount of text published online popular topics for authorship

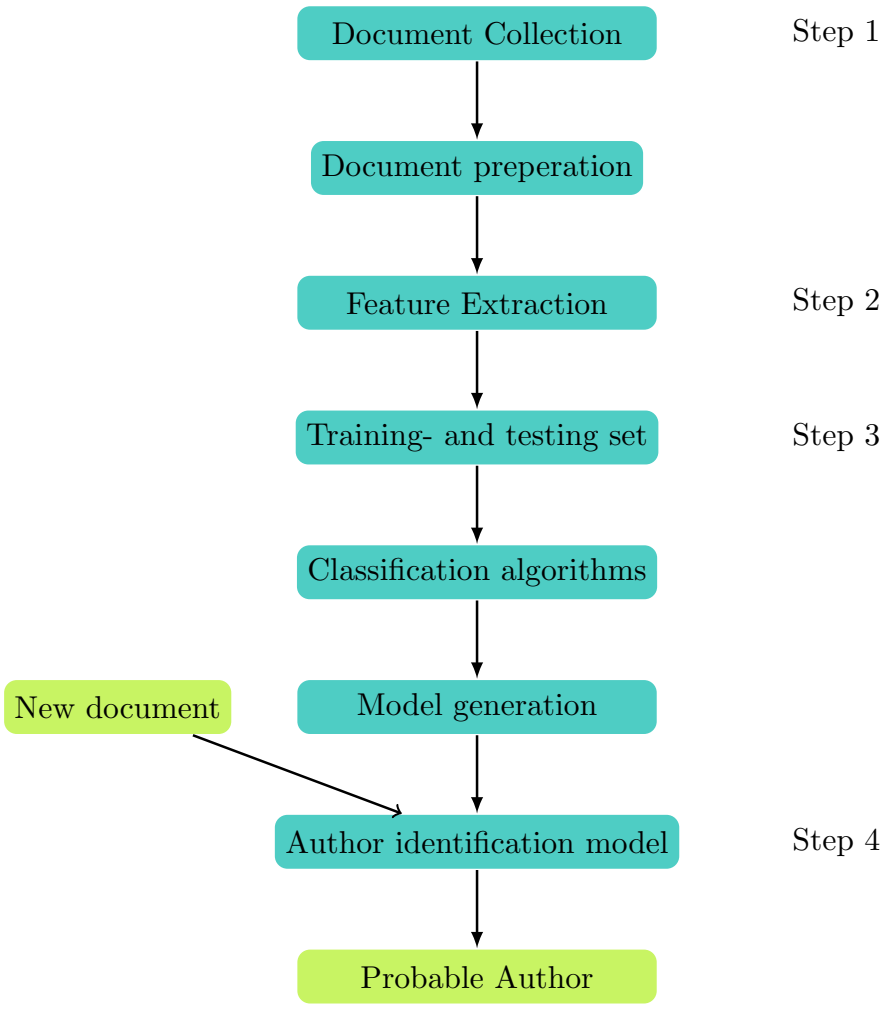

Figure 4.1: Author Identification process.

identification has been emails [8] [10] [30], newsgroup messages [43] and forum messages [1]. Other areas of authorship identification range from source code identification [12] to the identification of research papers based on only using the abstract [18]. In Table 4.1 the total number of authors for each research is shown, as well as the number of documents per author and the total number of documents.

Reference	# of authors	# of documents pr author	# of documents total
[18]	5+5	36 and 34	349
[39]	20	25	500
[30]	157+7	N/A for 157 and 30	620000
[1]	20	20	400
[42]	RCV1 corpus [1] and 10	N/A	RCV1 copus + 21
[10]	3	N/A	156
[8]	4	N/A	253
[6]	3+5	85 and 12	313
[43]	20	40	800

Table 4.1: Overview of the number of authors, documents per author and total number of documents in previous research.

4.4 Feature extraction

As seen in Chapter 2, a set of pre-defined features is needed to be able to classify data using classification algorithms. In authorship identification, these features are known as style markers and helps quantify an author's writing style. There are four groups of style markers that are used for authorship identification; these are the *lexical features*, the *syntactic features*, the *structural features* and the *content-specific features*. The lexical features are split into two sub-categories: the *character specific features* and the *word specific features*. The *character specific feature* considers a text as a sequence of characters, and the *word specific features* considers a text as a sequence of tokens (words). The *syntactic features* focus on an author's writing style. The *structural features* are based on how a text is built and the *content-specific* features are special features depending on the area authorship identification is performed on.

4.4.1 Character specific features

The character specific features are as mentioned that a text is parsed as a sequence of characters. Possible features that can be extracted in this group are the

[1] http://www.daviddlewis.com/resources/testcollections/rcv1/

total count of each letter in the alphabet i.e 26 different features in the English language, and 29 features in the Norwegian language. The total count of occurring digits can also be used providing ten more features from 0-9. The number of upper and lower case letters can also be used as features. Other character specific features range from the total count of special characters to the total number of characters used in the text [10] [43]. Character N-grams can also be used as a character specific feature. An N-gram is a sequence of characters or words consisting of N items from a given string of characters. For example, a 2-gram, also known as a bi-gram, of the word "city" is "ci","it" and "ty". Kjell [22] was one of the first to use character n-grams to determine the authorship of the "Federalist Papers". It is worth noting that by using character n-grams the dimensionality of the feature vector will drastically increase, and it is important that the classification algorithm can handle high-dimensional data. If the algorithm cannot handle high dimensionality data, consider leaving this feature out.

From what is done in previous work, as seen in Table 4.2, we can see that some researchers have chosen not to include any character specific features at all. While other researchers such as de Vel et al., [10], Zheng et al. [43], and Corney et al. [8] have used several. Also we can see that some features have been more common to use compared to others. For example the feature considering the total number of alphabetic characters are used by four out of the nine different studies while the ratio of digits to characters is only used by Nizamani & Memon [30]. In the tables below "# of" refers to "the total number of".

	Description	Reference
Character specific features		
# of characters		[30] [43]
# of alphabetic characters		[8] [10] [30] [43]
# of upper-case characters		[8] [10] [43]
# of digit characters		[8] [10] [43]
# of white space characters		[8] [10] [43]
# tab spaces		[8] [10] [43]
Frequency of special char.	~, @, #, $, %, ^, &, ∗, –, _, =, +, <, >, |, [,], {, }, \, /	[1] [30] [43]
Frequency of letters	From A to Z	[43]
Ratio of digit to char.		[30]
Ratio of space to word length		[30]
# of spaces		[8]
# of spaces/# of white spaces		[8]
# of tabs/ # white spaces		[8]
Character level		[1]
Character n-gram		[8] [42]

Table 4.2: Character specific features used in previous work.

4.4.2 Word specific features

The word specific features parses the document or text as a sequence of tokens. Important features include finding the total number of short words used by an author, the total number of words, the average word length and the average sentence length. Stamatatos [36] considers these features to be the token-based features. Vocabulary richness is also considered quite important, and several of these vocabulary richness measures exists today, for instance, Honore's R measure [17]. A study of different richness measures was done by Tweedie & Bayeen [40] in 1998. They go into the different vocabulary richness measures in detail, and they also showed that these types of vocabulary richness measures should not be used as a single source of information. Besides vocabulary richness another word specific feature is the word length frequency distribution, this feature counts the number of words with varying length, usually between one and 30 characters. As with character n-grams, word n-grams can also be used as features for authorship identification. The same problem applies here as with character n-grams, which is the drastic increase in dimensionality. The misspelling of words can also be used as a feature, this can include features such as missing hyphens or using the wrong vowel. Features that originate from misspelling are known as idiosyncrasies and has been used in previous studies such as a study performed by Koppel & Schler [23]

Previous studies show that word specific features are amongst the most popular. Nine out of nine papers proposed in this thesis has used some form of word-based features. Most common are the average sentence length in terms of word-length that are present in five out of nine studies alongside vocabulary richness and the average word length. For different languages, other features are used, for example in Abbasi & Chen [1] they use elongation, which is a feature specifically used for Arabic texts.

	Description	Reference
Word specific features		
# of words		[39] [43]
# of short words	Words less than 3 characters	[8] [10] [30] [43]
# of characters in words		[8] [10] [43]
Average word length		[10] [18] [30] [39] [43]
Average sentence length	Character-length	[8] [43]
Average sentence length	Word-length	[8] [10] [18] [30] [43]
Total different words		[39] [43]
Hapax Legomena	Words occurring only once	[8] [10] [43]
Hapax Dislegomena	Words occurring only twice	[8] [43]
Vocabulary richness		[1] [8] [10] [39] [43]
Word length frequency distribution	Frequency of words in different length	[1] [8] [10] [43]
Average words per document per author		[30] [39]
Total number of sentences		[39]
Average number of short words per author		[39]
Total number of function words		[8] [10]
Word Level		[1]
Elongation		[1]
Pronouns		[42]
Word n-grams		[6]

Table 4.3: Word specific features used in previous work.

4.4.3 Syntactic features

Syntactic features represent an author's writing style better than the lexical features does. The syntactic features takes into account how someone uses specific word classes, also known as part of speech, such as nouns and adjectives. These word classes are often used unconsciously, and pattern for a specific author can emerge given enough documents.

A common use of part of speech as a syntactic feature is to count the frequency of a word class, i.e., the total number of nouns. Part of speech n-grams can also be used as a syntactic feature. To create part of speech n-grams the entire corpus needs to be tagged with part of speech first using a tagger. The Stanford's part of speech tagger [2] is an example of a tagger that can be used for this purpose. When the tagging of the corpus is complete, n-grams can be made with the part of speech tagged corpus. The use of punctuation are also used as a syntactic feature, how often an author writes a period (.) or comma (,) can say much about an author's writing style. Another important syntactic feature is the use of function words. Function words are usually among the most common words, such as prepositions, pronouns and articles.The function words comes in a predefined list of words, each word represents a new feature, and this feature represents the frequency of that word. Zheng et al. [43] stated that there currently

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

are no general good go-to set of function words. The function words are however proven to be some of the best features to distinguish between different authors [4].

As we can see in Table 4.4 the use of function words are heavily favored between the studies as well as the use of punctuation. Both these features are used in six out of nine studies while part of speech are only used in three out of nine. Same as for the word and character specific features Abbasi & Chen [1] use language-specific features such as word roots.

	Description	Reference
Syntactic features		
Punctuation	",", ".", "?", "!", ":", ";", "", """	[1] [8] [10] [30] [39] [43]
Function words	List of function word provided by researcher	[1] [8] [10] [30] [42] [43]
Part of speech	nouns, verbs, adjectives etc.	[18] [39] [42]
Word Roots	For Arabic language	[1]
Phrases		[42]
Part of speech tag n-grams		[6]

Table 4.4: Syntactic features used in previous work.

4.4.4 Structural features

The structural features shows how an author systematically arranges the layout of a document. Some of these features are: how many lines, sentences or paragraphs a document has, the average size of a paragraph or check if a new paragraph start with an indentation or not. Even the font that was used, or the color of that font can be considered structural features. In previous studies, if the corpus consists of emails a feature used is whether or not the email starts with a greeting [43] or if the sender left his or hers signature [10].

From the previous study we see in Table 4.5 that most of the structural features depends on the format of the corpus. Zheng et al. [43], de Vel et al. [10] and [30] all use emails as the corpus and uses3 some very email specific features. However, some of these features can apply to most documents such as the total number of lines, sentences and paragraphs per document, and the total number of sentences, characters and words per paragraph.

4.4.5 Content-specific features

The content-specific features are features that mean something in a particular area or context. These features are among the harder ones to extract, particularly when working with a corpus that spans more than one specific area [36]. Zheng

	Description	Reference
Structural features		
# of lines		[8] [10] [43]
# of sentences		[43]
# of paragraphs		[30] [43]
# of sentences per paragraph		[43]
# of characters per paragraph		[30] [43]
# of words per paragraph		[43]
Has a greeting		[10] [30] [43]
Has separators between paragraphs		[43]
Has quoted content	Cite original message as part of new message	[43]
Position of quoted content	Quoted content above or beyond new message	[10] [43]
Email as signature		[43]
Telephone as signature		[43]
Url as signature		[43]
Tab or space indentation		[30]
# of blank lines		[8] [10]
Has farewell acknowledgment		[10]
Has signature		[10]
Number of attachments (email)		[10]
HTML tag frequency distribution		[10]
# of HTML tag		[10]
Message level		[1]
Paragraph level		[1]
Constant information		[1]
Font color		[1]
Font size		[1]
Embedded images		[1]
Hyperlinks		[1]

Table 4.5: Structural features used in previous work.

et al. [43] used content-specific words such as "obo" which means "or best offer" and "wtb" which means "want to buy". These examples were used for selling or buying items and were manually identified by reading through different "for sale" texts. Another set of features that is considered content-specific is to identify the most frequently used words and count the occurrences of these. Hurtado et al. [18] and Zhang et al. [42] the most frequent words as content-specific features in their work. Other content-specific features used in Abbasi & Chen [1] were an authors race and nationality, and whether or not a forum message proposed the use of violence.

4.5 Model generation

Chapter two introduced five different classification algorithms; these are the artificial neural networks, support vector machines, k-nearest neighbor, decision

	Description	Reference
Content-specific features		
Frequency of content specific words	Words that mean something in a specific corpus	Zheng [43]
Most frequently used words		Hurtado, Zhang [18] [42]
Race/nationality		Abbasi [1]
Violence		Abbasi [1]

Table 4.6: Content specific features used in previous work.

trees, and random forests. This section will go through each of these algorithms and how they have performed in previous work.

4.5.1 Building the model in authorship identification

Before we begin examining each algorithm individually we first have to look at how a model is created in authorship identification. From Chapter 2 we know that a classification algorithm needs a training set to build the model and a test set to evaluate the performance. For authorship identification the class in question consists of a single author, and as seen above the attributes for each text are the stylometric feature of an author's writing style. In essence, what a classification algorithm does in authorship identification is to output a probable author from an anonymous text based on the model generated by the training set, verified by the test set. Table 4.7 shows which algorithm was used in each study and Table 4.8 shows the respective best-case results achieved.

4.5.2 Classification and results

Support vector machine is the accepted go-to algorithm in the field of authorship identification. It was introduced to the field in 2000 by Diedrich et al. [11] and has shown an incredible performance since the beginning. One of the reasons this algorithm fits good in authorship identification is the ability to handle a large number of features [19]. Because support vector machines can handle a large number of features it does not suffer from the curse of dimensionality problem. As seen in Table 4.7 nine out of nine papers examined here has used support vector machines. Also, as seen from the results, in the best case Baraka et al. [6] achieved an accuracy of 100% in the best case while others float around 80-90% accuracy/f-measure.

Only two out of nine papers used artificial neural networks in their study, that does not mean the algorithm performs any worse than support vector machines in the best case. In Zheng et al. [43] artificial neural networks achieved an accuracy of 95%, only two percent behind support vector machines. Hurtado et al. [18] achieved a score of 0.97 area under the ROC-curve [38], where a score of 0.5 is

random performance, and a score of 1 is considered not random. However, this was only on two different authors in only one particular field. When used on ten different authors in two different fields the neural network algorithm achieved a score of 0.94 area under the ROC-curve.

In Zhang et al. [42] the k-NN algorithm had the worst performance and only achieved 60% accuracy in the best case while support vector machines were around 90%. For Hurtado et al. [18] they achieved a score of 0.92 on the area under the ROC-curve with two authors in the same field. The performance went down to a score of 0.87 on the area under the ROC-curve when the number of authors was increased to five.

Decision trees also did considerably well with a 91% accuracy in the best case for Zheng et al. [43] when using 30 documents and five different authors, and 90% accuracy for Abbasi & Chen [1]. For Zheng et al. [43] the performance decreased to 80% accuracy when the number of authors increased from five to 20, and as low as 69% accuracy when the number of messages were reduced to ten. It is worth noting that both studies used the popular C4.5 algorithm for constructing the decision tree.

Random forests were only used in Hurtado et al. [18], and the results shown were also limited. They achieved a score of 0.95 on the area under the ROC-curve in the best case when the number of authors were limited to two, and a score of 0.9 when increased to five authors.

From Nizamani et al. [30], Zheng et al. [43] and Hurtado et al. [18] we can see that the accuracy are related to the number of authors. We can see that when the number of authors increases, the accuracy will decrease. Both Zheng et al. [43] and [1] shows that when more feature groups are added the accuracy increases. Only lexical features performs worse than lexical + syntactical and, lexical + syntactical performs worse than lexical + syntactical + structural. Zheng et al. [43] and Nizamani et al. [30] also show that the number of documents per author matters, and that the accuracy increases when the number of documents per author increases.

4.6 Summary

This chapter has given insight into the previous work done in the field of authorship identification. We have also looked at the document collection- and feature extraction phase in detail. The next chapter will go through how this thesis implemented the framework described in this chapter and which decisions had to

Previous research	ANN	k-NN	SVM	Decision trees	Random forests
Zhang et al. 2014 [42]		x	x		
Baraka et al. 2014 [6]			x		
Zheng et al. 2006 [43]	x		x	x	
de Vel et al. 2001 [10]			x		
Corney et al. 2001 [8]			x		
Nizamani & Memon, 2013 [30]			x		
Abbasi & Chen, 2005 [1]			x	x	
Hurtado et al. 2014 [18]	x	x	x		x
Tufan Tas & Abdul Kadir Görür, 2007 [39]			x		

Table 4.7: Algorithms used in previous work.

	ANN	KNN	SVM	DT	RF	Measure
Results						
Zhang et al. 2014 [42]		60%	90%			Accuracy
Baraka et al. 2014 [6]			100%			Accuracy
Zheng et al. 2006 [43]	95%		97%	91%		Accuracy
de Vel et al. 2001 [10]			90%			F-measure
Corney et al. 2001 [8]			82%			F-measure
Nizamani & Memon, 2013 [30]			94%			Accuracy
Abbasi & Chen, 2005 [1]			97%	90%		Accuracy
Hurtadi et al. 2014 [18]	0.97	0.92	0.92		0.95	Area under ROC
Tufan Tas & Abdul Kadir Görür, 2007 [39]			58%			Accuracy

Table 4.8: Best performance of the classifiers.

be made and why.

Chapter 5

Implementing the authorship identification system

This chapter shows how the authorship identification framework presented in Chapter 4 were used in the architecture of the implemented system for this thesis. Including data collection, feature extraction, and model generation, Figure 5.1 illustrates how we intended to implement the system, and we will now go into detailed about how we performed our implementation. This chapter will also talk about some of the alternative options that were considered and explain the process behind decisions that were made.

5.1 Data collection

Performing authorship identification requires a corpus. As previously seen in Section 4.3 a corpus can be everything from a collection of emails to source code and in this thesis the corpus consists of research papers. However the term "research paper" is very broad and to be more concrete our corpus exists of research papers in computer science under the subject of information retrieval. Two different data sets were made based on the corpus, the first one includes only research papers from information retrieval proceedings, such as "SIGIR", "ECIR" and "CIKM". The other dataset consists of several research papers from authors within information retrieval. The reason for having two separate data set was because we wanted to see how the classification algorithms performed when given different amount of data for training and testing. In the first data set, we have a small

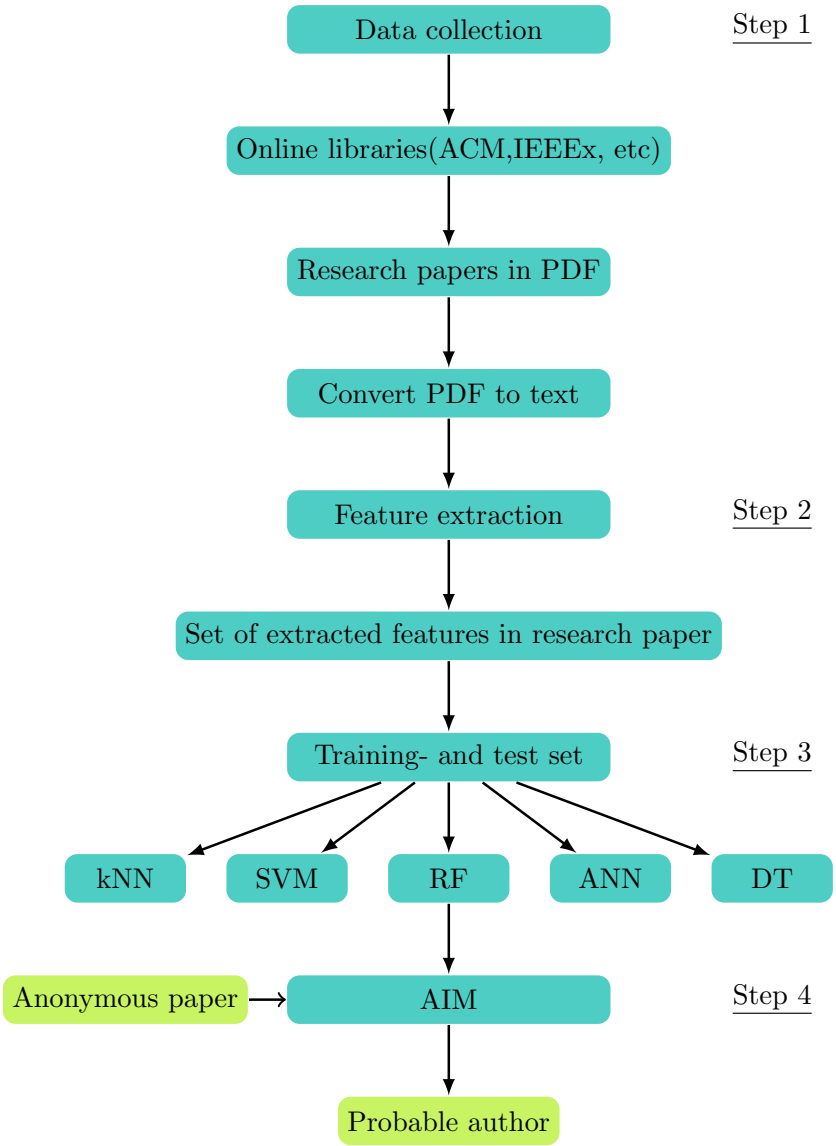

Figure 5.1: Author Identification process.

number of authors with a good amount of research papers per author. In the second data set, we have a large number of authors with a limited number of research papers per author.

For the first dataset, the dataset with a limited number of authors, the research papers were saved in separate folders. Each folder were named after the candidate author, and because we could do a lookup on the folder name, no automatic author name extraction were necessary. However, for the second data set the research papers were not structured this way. Therefore, for this dataset we needed a way to automatically extract the author name from the research paper. How we performed the author name extraction is explained in Section 5.1.2

5.1.1 Preprocessing

The next step in the process after establishing the corpus was to extract usable text from a research paper. All of the research papers used in the corpus were originally in PDF format, and automatic text extraction had to be performed to be able to extract the stylometric features. For the preprocessing of the PDFs, two different approaches were considered. The first option was to use a Java library to extract text from the PDFs directly, and the second option was to use OCR, also known as optical character recognition [14]. For regular PDF text extraction, Apaches PDFBox[1] and a layout aware text extractor for scientific articles, known as LAPDF [34] were the two candidates reviewed for the task. A major drawback with PDFBox was that extracting text from research papers created noise in the output text when extracting content from tables, pseudocode, and figures. Besides creating noise in the output, the text extractor performed very well overall.
Compared to PDFBox, the LAPDF library did not suffer to the same degree when it came to the frequency of noise in the output text. The reason for that is because the LAPDF-library intelligently ignores authors, article headlines, figures, tables, and pseudocode. However LAPDF came with a drawback as well, being optimized to work on two-column scientific articles it did not work on single-column articles.
For the second option, optical character recognition, ABBYY[2] was tested out. By being both slow, providing the same amount of noise in the data set as PDFBox and had to be purchased to access the full version, OCR was decided to not be used in the implementation.
We ended up with two possible solutions both offering two major drawbacks. On one side by settling with PDFBox we would end up with a noisy dataset and by choosing LAPDF, single-column articles had to be ignored, such as research

[1] https://pdfbox.apache.org/
[2] http://www.abbyy.com/

papers from the "ECIR" conference. We ended up with a compromise and used LAPDF whenever possible and PDFBox for the single column articles, with this solution we ended up with less noise in the dataset and able to handle more research papers.

5.1.2 Finding the author

Besides automatically extract text from the PDF for the second dataset with a large number of authors and limited training and test data we also needed to automatically extract the name of the authors of the research paper. The initial assumption was that the authors of a given research paper could be identified using the metadata of the PDF. However, it turned out that most of these PDFs only contained auto-generated metadata from the compiler and that the authors rarely bothered to update the metadata with the correct information. Therefore, the idea that the metadata could be used as a reliable source to find out who wrote the research paper was scrapped. The second solution proposed was to use the title of the research paper and perform a lookup in a database containing information about research papers and their respective authors. The database was created using a MYSQL-database created from the XML-file DBLP[3] provides free of charge which contains the entire DBLP-database. The next problem to be solved was to automatically extract the title from the research paper. Since the filename of the PDFs were mostly obscure and unusable, we could not use the filename of the PDF directly to identify the authors. Fortunately, a Java library named "Docear's PDF Inspector" [4] could do the job of extracting the title. How the title extraction works is that the library extracts the largest text from the top third of the first page of a PDF. For a research paper that is in most cases the title. After the title had been extracted, it was then used to query the database to identify the candidate authors. Figure 5.2 illustrates how the automatic author extraction process works.

5.2 Automatic feature extraction

In the implementation of the automatic feature extraction, only two out of the four feature groups were implemented. The structural and content-specific features were left out because of a limitation in the pre-processed text files generated when using PDFBox or LAPDF. The limitation was that a text-file created by extracting the content from a PDF did not maintain the structure the text had in the PDF. Therefore, we could not extract these features accurately enough to consider them reliable features. However we did come up with some possible

[3] http://dblp.uni-trier.de/
[4] https://www.docear.org/software/add-ons/docears-pdf-inspector/

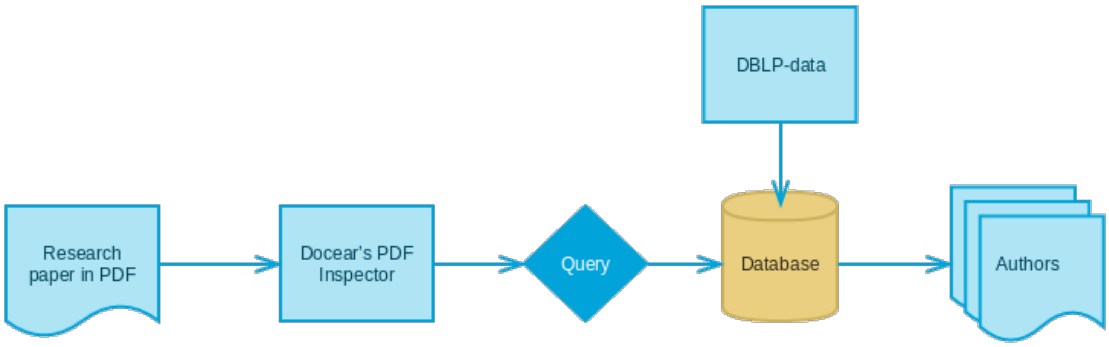

Figure 5.2: Author extraction process.

structural- and content-specific features that were not discussed in Chapter 4. For the structural features these were: *"Related work early or late in the research paper"*, *"Are the references in sorted order according to the reference list"* and *"Reference style"*. For the possible content-specific features, *"Total number of references"*, *"Usage of footnotes"* and *"Total number of footnotes"* were proposed.

5.2.1 The lexical features

The lexical features used in the implementation are influenced by the survey done in Chapter 4. Most of the features used by more than one of the previous studies has been used, but some have been chosen not to implement such as character and word n-grams. The reasoning for not including n-grams in the implementation was to avoid the curse of dimensionality problem, and thereby have an increased amount of running time for some of the algorithms. The entire list of lexical features can be seen in Table 5.1.

5.2.2 The syntactic features

The syntactic features are also inspired by the previous survey, and the list of function words used are the same list of function words used by Zheng et al. in [43]. The reasoning behind using their list of function words was because they stated that there are no good rules for deciding which function words to include. Therefore, we came to the conclusion that we would not come up with a better list of function words our self and picked the list they used for their research. The part of speech features were extracted by using Stanford's natural language processor[5] and each part of speech tag are mentioned in Table 5.2.

[5]http://nlp.stanford.edu/software/tagger.shtml

Lexical	Description
L1	# of characters
L2	# of alphabetic characters
L3	# of uppercase characters
L4	# of digits
L5	# of white space characters
L6	# of tabs
L7 - L32	Frequency of alphabetic characters, case insensitive
L33 - L54	Frequency of special characters (˜, @, #, $, %, ˆ, &, ∗, –, _, =, +, <, >, |, [,], {, }, \, /)
L55	# of words
L56	# of short words
L57	# of charcters in words
L58	Average word length
L59	Average sentence length in characters
L60	Average sentence length in words
L61	# of different words
L62	Hapax legomena (words occuring only once)
L63	Hapax dislegomena (words occuring twice)
L64	Vocabulary richness by Yule's measure
L64	Vocabulary richness by Simpson's D measure
L65	Vocabulary richness by Sichels S measure
L66	Vocabulary richness by Brunet's W measure
L67	Vocabulary richness by Honores measure
L68-L88	Word length frequency distribution (1 - 20 characters per word)

Table 5.1: List of lexical features used.

Feature	Description
S1 - S8	**Frequency of punctuation** (",", ".", "?", "!", ":", ";", "", """)
S9-S156	**Frequency of function words** (about,above,after,all,although,am,among , an,and,another,any,anybody,anyone,anything ,are,as,at,be,because,before,behind,below, beside,between,both,but,by,can,cos,do,down, each,either,enough,every,everybody,everyone ,everything,few,following,for,from,have,he ,her,him,if,in,including,inside,into,is,it ,its,latter,less,like,little,lots,many,me ,more,most,much,must,my,near,need,neither ,no,nobody,none,not,nothing,of,off,on,once ,one,onto,opposite,or,our,outside,over,own ,past,per,plenty,plus,regarding,same,several ,she,should,since,so,some,somebody,someone, something,such,than,that,the,their,them,these ,they,this,those,though,through,till,to,toward ,towards,under,unless,unlike,until,up,upon,us, used,via,we,what,whatever,when,where,whether, which,while,who,whoever,whom,whose,will,with, within,without,worth,would,yes,you,your)
S157-S193	**Frequency of part of speech** (Coordinating conjunction, Cardinal number , Determiner, Existential there, Foreign word , Preposition or subordination conjunction , Adjective, Adjective(comparative) , Adjective(superlative), List item marker , Modeal, Noun(singular or mass), Noun(plural) , Proper noun(singluar), Proper noun(plural) , Predeterminer, Possesive ending, Peronal pronoun , Possessive pronoun, Adverb, Adverb(comparative) ,Adverb(superlative),Particle, Symbol, to , Interjection, Verb(base form), Verb(past tense) , Verb(gerund or present participle) , Verb(part participle), Verb(non-third person singular present) , Verb(third person singular present) , Whdeterminer, Whpronoun, Possessive whpronoun , Whadverb

Table 5.2: List of syntactic features used.

5.2.3 Output

In total 193 lexical and syntactic features has been used in the implementation, these are represented as numeric values in a file format known as ARFF[6]. The ARFF-file, also known as attribute-relation file format, consists of a header field and a data field. In the header field, the name of the relation, and a list of all the features used are present, and in the data field each instance to be classified is represented as a string of values. These values can be a combination of strings, dates, numerics or nomials. An example of an ARFF-file can be seen in Appendix A.
In the implementation, each text-file is individually processed through a feature extraction object written in Java and the extracted features are written to the end of the ARFF-file as instances. Figure 5.3 illustrates the entire feature extraction process, and the integration with WEKA, a machine learning library is explained in Section 5.3.

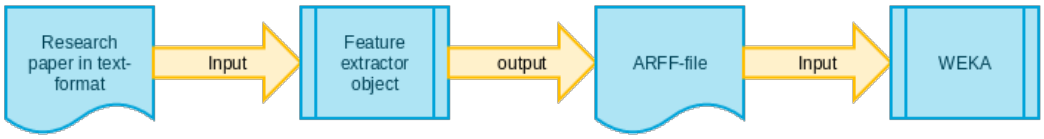

Figure 5.3: Feature extraction process.

5.3 Approach

Five different classification techniques were chosen to be used for the authorship identification, these were: support vector machines, k-nearest neighbor, decision trees, random forests and artificial neural networks. In the implementation, we used WEKA [16], a machine learning library, to generate a classification model during the training phase, and test the model out during the test phase. For each classification algorithm WEKA has one or several different implementations, and the implementation used in this thesis where: Platt's sequential minimal optimization algorithm [31] for support vector machines, instance based k [2] for the k-nearest neighbor algorithm, Quinlan's C4.5 [33] for the decision trees, Breiman's random forests algorithm [7] and multilayered perceptrons [13] for the artificial neural network. We have implemented two different ways of performing the classification, one for each of the datasets mentioned in Section 5.1.

[6]http://www.cs.waikato.ac.nz/ml/weka/arff.html

5.3.1 Implementation 1: limited authors

The first implementation were created for examples where we could perform authorship identification on a limited amount of authors where the author of each paper were known beforehand. What we did was to implement a 10-fold cross-validation to train and test the data and systematically increase the number of authors in the data set, from 2 to 20.

5.3.2 Implementation 2: many authors

For the second implementation, we intially decided that training should be performed on the lead author of the paper, and the testing should be performed on every author that contributed to the paper. However due to performance issues discussed in Chapter 6 we went through several iterations on the design and decided to train our model on a prioritized list of authors. For the second implementation we had a separate test set, and we had to decide what a correct classification should look like. The obvious choice would have been only to use the lead author of the papers in the test set. This would have been easy to implement, but we assumed that the results for the many author problem would be unsatisfactory. In the end we decided that a classification should be marked as correct if the following criteria was satisfied: A classification is marked as correct if the model can predict one of the contributing authors of the research paper. This meant that we would have to perform a lookup through our database to find every author that contributed to a research paper and hold on to that information until the model could make a prediction. Since WEKA does not support the ability to credit a classification as correctly from a list of candidate classes (authors) this functionality was implemented to support the WEKA framework and make the classification process possible. To explain the process in its entirety the classifier is first trained on every paper prior to 2014 on the author that matches the first author in the prioritized list. This means that only one author is considered during training. When the training is complete the generated model can begin to accept instances from the test set, where the test set includes research papers published after and during 2014. For each instance in the test set, if the model can predict one of the authors, we classify that instance as correctly classified. Another reasoning behind using every author that contributed to the paper for each test instance is that for a real world problem we would be satisfied if we managed to predict at least one of the authors that wrote the paper. An illustration of how the second implementation works can be seen in Figure 5.4.

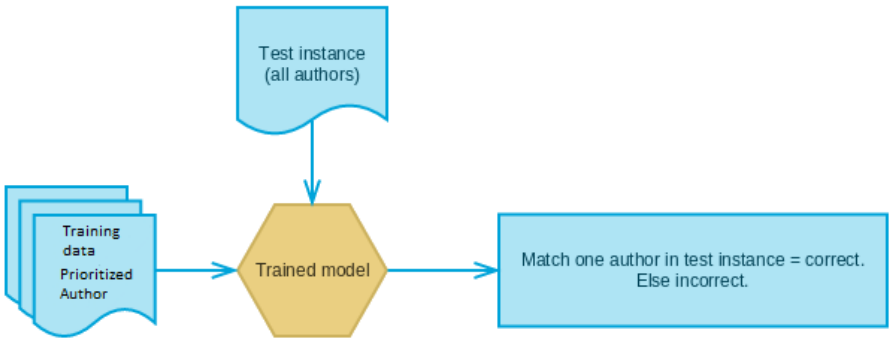

Figure 5.4: Author identification process with limited data.

5.4 Summary

This chapter has shown how we implemented the different steps in the authorship identification process. This included how we preprocessed the research papers, which features we decided to include in the implementation and how we performed the classification.

Chapter 6

Experiments and results

In this chapter, we go through how we performed the experiments and present their respective results. The chapter starts with the experimental plan where we discuss the different datasets, metrics, and the perspective of the results. Followed up by the results and a brief discussion about the results we observed during the experiments.

6.1 Experimental plan

This section presents our perspective of the experiments, the datasets used in these experiments and which metrics we decided to use for each experiment.

6.1.1 Perspective

There are two parts the implemented system is trying to solve. The first part, known as Experiment 1, considers a dataset with a limited number of authors, where each author is represented with enough training and test data. The second part, known as Experiment 2, considers a dataset with a large number of authors, and limited training- and test data for each author. The reasoning behind having these two datasets is as mentioned in Section 5.1 to see how the system handles different cases of authorship identification. Another reason for having Experiment 1 in the mix was to get results for a smaller set of authors, for example the two-author problem where only two different authors are considered. Good results on such small cases may intend that even though the system fails for a large number of authors it is still usable under the right conditions.

6.1.2 Datasets

Experiment 1

The dataset used consisted of 20 different authors with a maximum of 20 published research papers each. This dataset was further divided into three new datasets with different heuristics. In the first dataset, the heuristic was set to only consider research paper where the candidate author is also the lead author of the paper. Since we found a limited number of authors that were the lead author in twenty or more research papers this dataset was set to only contain ten published papers each. In the second dataset the heuristic was set to consider research papers where the candidate author is among the top 3 contributors of the research paper. For this case, we managed to gather the maximum of which was twenty research papers per author. In the last dataset the heuristic was set to include research papers where the candidate author has at least contributed something to the research paper, this dataset also had the maximum of twenty research papers per author. The reasoning behind having three different datasets each with their own heuristic was to observe how important the amount of work contributed to a research paper had to say for the result.

To summarize the dataset for the first experiment:

Dataset 1
Heuristics: Author is the lead author
Number of research papers: 10 for each author, 200 in total

Dataset 2
Heuristics: Author is among the top 3 contributors
Number of research papers: 20 for each author, 400 in total

Dataset 3
Heuristics: Author has contributed something to the paper
Number of research papers: 20 for each author, 400 in total

Experiment 2

For Experiment 2 the research papers in the dataset were taken from different information retrieval conferences. The conferences included in the training- and test set can be seen in the list below.

Training set
CIKM 2005, CIKM 2011, CIKM 2012, CIKM 2013, SIGIR 2005, SIGIR 2011, SIGIR 2012, SIGIR 2013, ECIR 2012, ECIR 2013 and RIAO 2007.

Test set
SIGIR 2014, CIKM 2014, ECIR 2014 and ECIR 2015.

As mentioned in Section 5.3.2 this dataset went through several iterations before the final dataset was established. After attempting to train the classifier on the lead author heuristic we realized that the training process was very slow. To clarify how slow the training process was, the only algorithms that finished training on this dataset were the decision tree algorithm and k-nearest neighbor. The support vector machines, random forests, and artificial neural networks never finished training. The reasoning that the training process was so slow for three of the algorithms were that the dataset generated was very sparse, with several thousand candidate authors, and most of them with only a single paper they had contributed to.

Normalization

Because of this the initial design had to be changed, and the idea that we came up with was to normalize the dataset. The normalization for the second iteration of the design was to identify the authors that had contributed to the most papers in the dataset and list them up in a prioritized manner, from most contributions to least contributions. This list was used to assign an author to the paper in the training set. By getting every author for the paper and cross check these authors to the prioritized list of authors we could assign the first match in this list as the author of the paper.

Third iteration

Normalization of the dataset was a step in the right direction, the number of candidate authors was reduced, and the highest number of papers per author went from a maximum of six up to thirty. To our surprise, the training process was still running very slow for the support vector machines, random forests, and neural network algorithms. We still had about 1200 different candidate authors in our training set, and we decided that we needed to modify the dataset even further. A decision was made to remove every author that only had contributed to one research paper. With the third iteration of the design, the classification algorithms were able to perform training within a reasonable time.

Final dataset

The final numbers of the training set were as follows: 1683 different instances, 408 unique authors, the highest amount of papers per author were 30 and the lowest number of papers per author were 2. The test set consisted of 707 different instances. It is also worth noting that research papers in the dataset that were

unreadable by our PDF-parser, and research papers that we were unable to find in our database was not included in either the training set, nor the test set.

6.1.3 Metrics

For the first experiment which we expected to perform reasonably well we used three different metrics, these were: *success rate*, which is the number of instances that is classified correctly. The second metric we used was the *F1-measure* which is as mentioned in Chapter 2 is the harmonic mean between precision and recall. The last metric we consider is *Cohen's Kappa* which measures the internal agreement of the classifier.
For the second experiment we only considered the *success rate* because we had already made the assumption that the results from this experiment was going to be poor.

6.2 Experimental setup

The experiments are performed by gathering data from different proceedings and Google Scholar. For the first experiment approximately 500 research papers were manually downloaded and about 450 of these were approved for use in the different datasets mentioned above. We had to download more than 450 research papers because there were some of the papers our system were unable to parse. For the second experiment we used research papers published in different information retrieval proceedings, these papers were provided by Kjetil Nørvåg, the supervisor for this thesis. We have attached our ARFF files to make it easy to replicate the results from Experiment 1 by using the WEKA explorer. However for Experiment 2, due to the nature of how it was implemented the experiment has to be executed directly from the Java application. Instructions on how to set up the database and run the implemented system please refer to Appendix7.4.

6.3 Experiment 1

In the first experiment we considered the three datasets mentioned in Section 6.1.2. We ran each dataset with a different number of papers per author ranging from five to twenty. The reasoning for doing this was to not only compare the different heuristics but also how the classification on each dataset performs with a different size of training- and test data.

6.3.1 Dataset 1 - lead author

In the first dataset we considered the heuristic where the author of the paper also has to be the lead author of that paper. Two iterations of the classifiers were executed on this dataset, the first iteration only used five research papers per author as training- and test data, while the second iteration used all ten research papers. The results from this dataset can be seen in Figure 6.1 and Figure 6.2

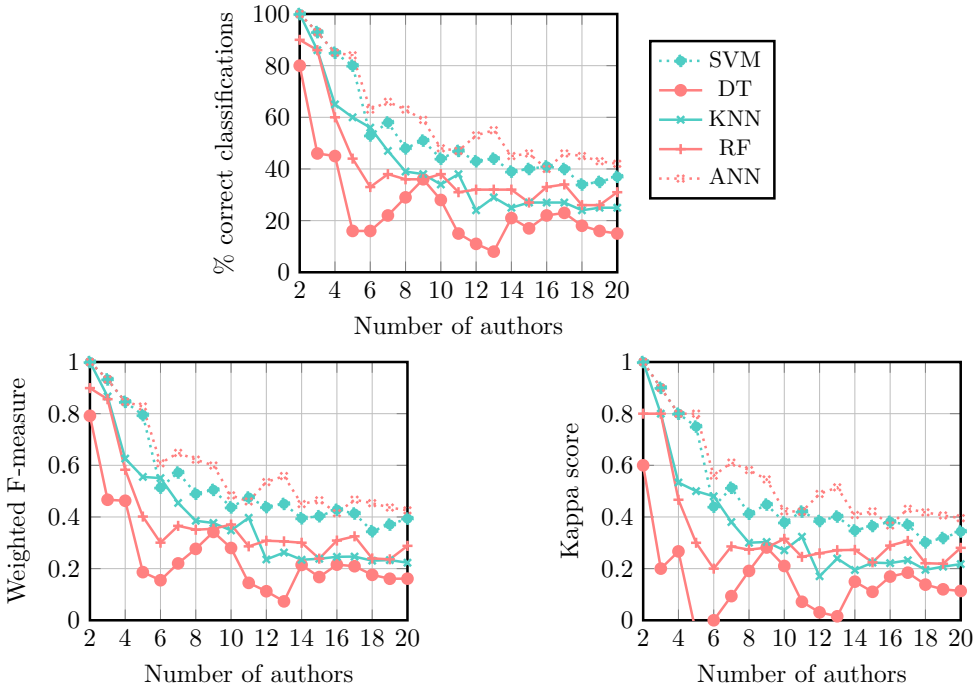

Figure 6.1: Lead author with 5 papers per author.

6.3.2 Dataset 2 - top 3

In the second dataset, we considered the heuristics where the author of the paper had to be among the top three authors that contributed to the paper. Also in this dataset two different iterations were executed. The first iteration had fifteen papers per author and the second iteration had twenty papers per author. The reasoning behind not testing with ten and five research papers per author was because we had already done this with what we believe is a better heuristic in the first dataset. The two iterations in the second dataset could, therefore, be compared directly with the first dataset as well as internally with a different number of training- and test data. The results from this dataset can be seen in Figure 6.3 and Figure 6.4

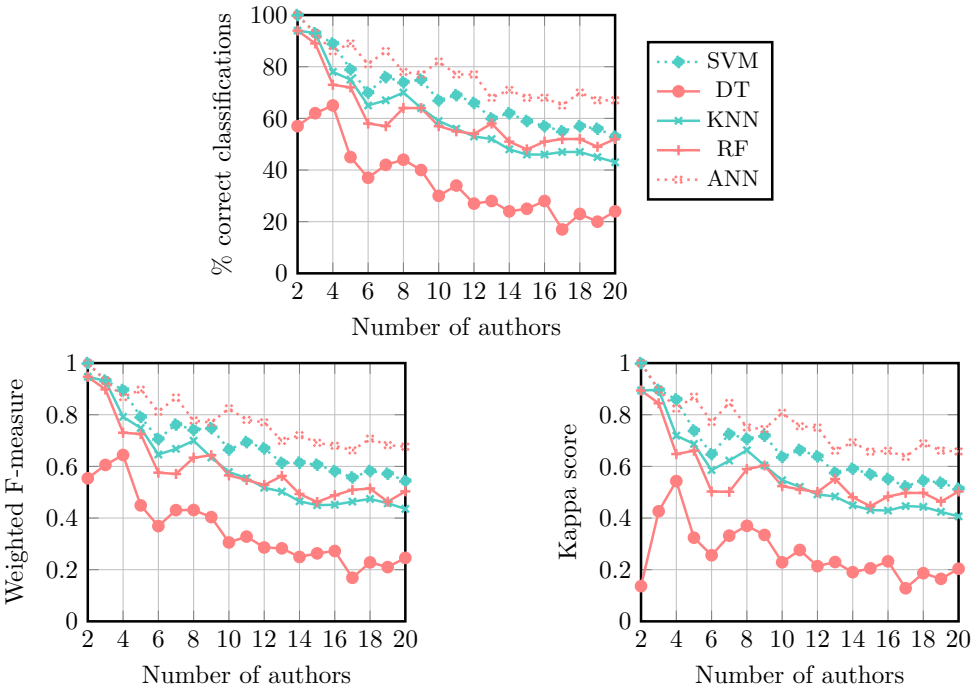

Figure 6.2: Lead author with 10 papers per author.

6.3.3 Dataset 3 - random

For the last dataset we considered the heuristics that the author of the paper could be in any order, i.e., the author could be mentioned last in the list of contributors. We only ran one iteration of this dataset. The number of research paper used in this iteration was twenty and with the same reasoning as above this can be directly compared to the second and first dataset. The result from this dataset can be seen in Figure 6.5

6.3.4 Observations

Algorithm performance

From the results presented above we can see that the algorithm that performed best overall was the artificial neural network with multilayered perceptron and not far behind we have support vector machines. The random forests performed worse than artificial neural networks and support vector machines but overall better than both the k nearest neighbor and decision trees. The most surprising result was that the decision tree algorithm performed so horribly. In previous work as seen in Chapter 2 decision trees showed decent performance and the idea

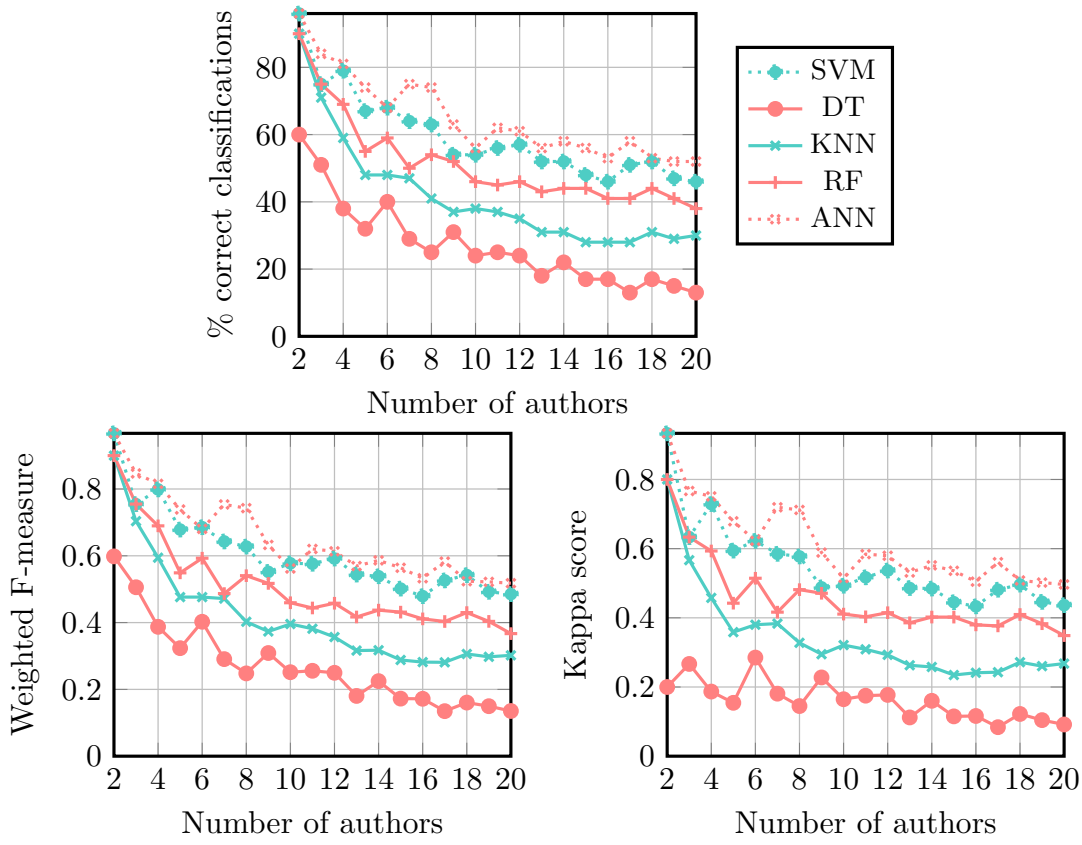

Figure 6.3: Top 3 author with 15 papers per author.

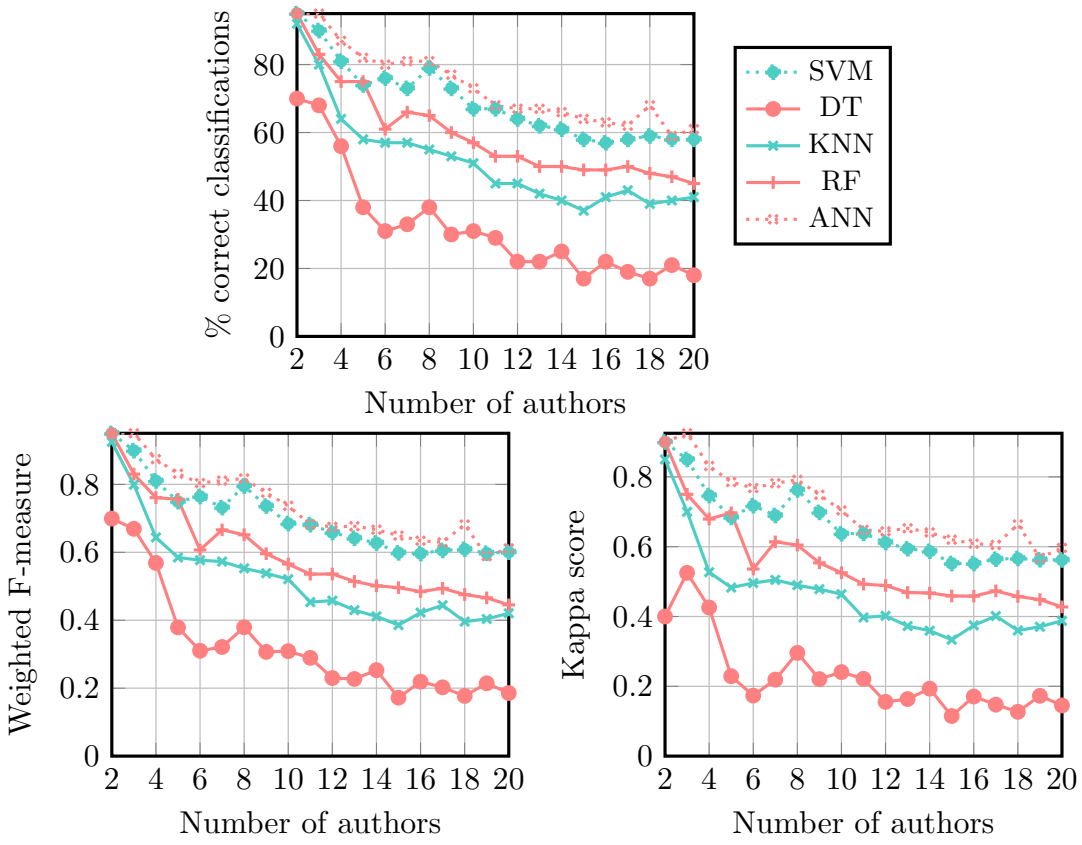

Figure 6.4: Top 3 author with 20 papers per author.

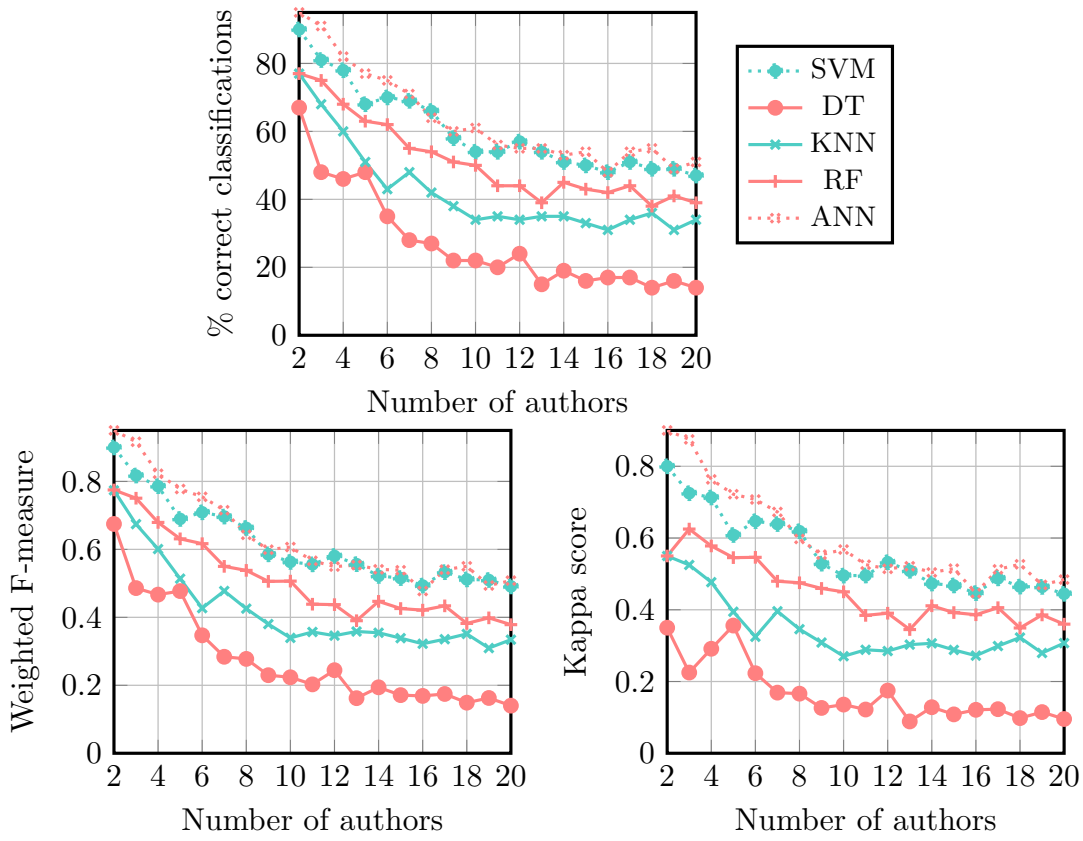

Figure 6.5: Random position of author with 20 papers per author.

was that decision trees would at least perform on par with k nearest neighbor. Table 6.1 shows the standings of the different algorithms. The classification algorithms were run a total of between nine and nineteen times depending on the dataset where we for every iteration added a new target author, from two authors up to a maximum of twenty. Every time we added a new target author the problem at hand got a little harder to solve. For the two-author problem, where we only consider assigning a paper to one of the two target authors our classifier reach a maximum accuracy of 100% with a kappa score and F1-measure of 1. As seen in the result graphs above for the twenty-author problem we reach a maximum accuracy of 67% with a kappa score of 0.66 and an F1-measure of 0.68. In the two subsections below we discuss how we interpreted the results for the twenty-author problem. The main reason we chose to take a close look at the twenty-author problem was because we believe that it was the hardest problem for our classification algorithms to solve and, therefore, provides the most important result. For the exact numbers in the results graphs, please refer to Appendix 7.4.

Performance	Algorithm
1	ANN
2	SVM
3	RF
4	KNN
5	DT

Table 6.1: Overall performance of the algorithms.

Comparing the number of training and test instances

Previous work done on authorship identification shows that increasing the amount of training data to the classifier often makes the classifier perform better. In our experiment we did tests to see how our classifier and dataset holds up to that statement. We performed these tests on the lead author dataset and top 3 author dataset. For the first dataset we started to test with five research papers per author for our ten-fold cross validation and increased the size of the dataset to ten research papers per author. When considering all twenty authors in the dataset we saw a clear increase in performance that can be seen in Table 6.2. For the accuracy, the number of correctly classified instances the performance improved by 61%, for F1-Measure the performance increased by 64% and for the Kappa score the performance increased by 72%. For the dataset considering the top 3 authors the overall performance of the classifier also increased, however by a smaller amount. The performance in accuracy increased by 27%, the F1-measure

increased by 27% and the Kappa score increased by 36%. Overall this confirms what has been done in previous research and shows that the performance of the classifiers increase when the number of training- and test instances increases.

Comparing heuristics

One of the goals we had with our thesis was to see how much of a difference it made that the author was presented as lead author of the paper versus being among the top contributors or even one of the authors that contributed least to a paper. Initially we assumed that being lead author of a research paper would give the author permission to structure the paper how he or she wanted and most likely that author has contributed most to what is written in the paper. Therefore our hypothesis was that it would be easier to classify papers where each author also was the lead author of the paper. The hypothesis also assumed that being among the top three contributors to the research paper would perform better than the heuristic that the author of a research paper could be any of the authors. As seen in Table 6.2 comparing the lead author dataset with ten papers to the top 3 author dataset with fifteen papers we can see an improvement in performance by between 39 to 53% when applying the classification algorithms to all twenty target authors. By increasing the number of papers per author from fifteen to twenty for the top 3 dataset the performance of the lead author heuristics still performed between 9 to 11% better. This is, however, a considerably smaller increase in performance compared to the dataset with fifteen papers per author. What we can conclude from this is that the lead author heuristics is indeed better than the top 3 contributor heuristic considering the performance is better and the dataset is at most half the size of the top 3 contributors dataset. For the last comparison that we are going to do we look at the difference in performance between the top 3 contributor heuristic with twenty papers per author and compare it to the heuristic where the author can be any of the authors that contributed to the paper. Also in this case we can see that our hypothesis is correct because the top 3 contributor heuristics performs between 22 to 30% better than the random heuristic. We believe that this is the case because if you are listed as the author that contributed least to the paper your writing style, or stylometry, doesn't appear nearly as often in the paper as it would have done if you were among the authors that contributed the most.

6.4 Experiment 2

The second experiment we consider a single dataset that has been created based on research papers from several information retrieval conferences mentioned in Section 6.1.2. This experiment is what we consider to be a problem that we

Comparisons	Kappa	F-measure	Accuracy
Lead author (10 vs 5 papers)	72%	64%	61%
Top 3 author (20 vs 15 papers)	36%	27%	27%
Lead author 10 papers vs top 3 15 papers	53%	39%	42%
Lead author 10 papers vs Top 3 20 papers	11%	9%	11%
Top 3 20 papers vs Random 20 papers	30%	25%	22%

Table 6.2: Percent increase in performance for different datasets for the twenty-author problem.

can perform authorship identification in the real world, and we consider this experiment the real test of this thesis.

6.4.1 Results

Table 6.3 shows the result from the test set. The main difference between this experiment and Experiment 1 is that we decided to use a dedicated training set, and a dedicated test set. The reasoning behind this choice is that we wanted to attempt to use authorship identification on a real world problem to see how our system would perform when put on a difficult test.

Algorithm	Correctly classified	Accuracy
SVM	70 out of 707 correctly classified	9.9%
Neural Networks	66 out of 707 correctly classified	9.3%
KNN	48 out of 707 correctly classified	6.8%
Random Forests	40 out of 707 correctly classified	5.7%
Decision trees	9 out of 707 correctly classified	1.3%

Table 6.3: Accuracy and number of instances correctly classified for Experiment 2.

6.4.2 Observations

Initially, the assumption was that the classification results from this experiment would be very bad. At one stage of the process we would consider us lucky if we were able to perform the experiment at all with all the different design changes on the training set to be able to perform the training. From the results, we can see that the SVM algorithm comes out ahead with an accuracy of 9.9%. There were some surprises by comparing the results from this experiment to the previous experiment. Previously the neural network algorithm performed overall better than the support vector machines algorithm and in this experiment the

support vector machines algorithm outperforms the neural network algorithm with 0.6% accuracy. Also, the KNN algorithm outperformed the random forests algorithm by 1.1% in accuracy which also differs from what we saw in our previous experiment. Moreover, as it always has the decision tree algorithm came in last with an accuracy of 1.3%.

6.5 Summary

In this chapter, we have looked at results for both our experiments and discussed their respective results. We have also proved our hypothesis by showing through our results by determining which heuristic was best and performed an experiment on a real world problem. In the next chapter we will whether or not we feel we answered our research questions for this thesis, which problems we encountered when we implemented the system, and further work.

Chapter 7

Conclusion

In this chapter we conclude our work, we discuss and answer the research question presented in the introduction, what this thesis contributes to the research field and recommendations for further work.

7.1 Discussion

In the introduction to this thesis, three different research questions were asked. In this section, these questions are answered by evaluating our work and the results presented in Chapter 6.

RQ1: What are the different features and techniques used in authorship identification field when looked at as a classification problem?

Before we began implementing our system to perform authorship identification we wanted to know what studies had been conducted before in this area and which features and techniques were considered the state of the art. In Chapter 2, 3 and 4 we have looked at the theory behind classification, how to measure the performance of classification algorithms, and talked about the algorithms used in our work. We have also looked at other areas that relate to authorship identification and several different studies in the research field. But have we been able to answer the research question? We believe that the work we have done in this thesis is sufficient for the reader to get an understanding of the problem in question, but that there may be previous important studies that possibly have been overlooked.

RQ2: Is it possible to perform authorship identification on research papers?

Throughout the period of writing this thesis, a system was implemented with the sole purpose of using classification to identify authors of research papers. To determine whether or not authorship identification of research papers we built the system to be very similar to what has been done before. Each feature was stripped down to a set of features to determine the authors writing style and use these features as input to a classification algorithm. From the results presented in Chapter 6 authorship identification of research papers using classification is indeed possible but according to our results only to some extent. For a very limited number of authors, authorship identification can be applied with a very high accuracy reaching an accuracy of up to 100% in the simplest case the two-author problem. However when we look at the results from our second experiment where we have over 400 different authors for our algorithms to choose from the accuracy plummets drastically compared to the first experiment. There have also been some limitations that we have had to handle that has to be taken into account when evaluating our results. The first limitation we want to talk about and that most of our system relies on is the ability to automatically parse and convert a PDF-file to plain text. Even though we tried to find the PDF-text extraction libraries that suited our problem best there were still some limitations. These limitations hindered us in implementing a whole category of features that could have helped us increase the accuracy of the classifiers even further. Creating a library with the sole purpose of extracting text and maintaining the structure of the research papers is something we feel fitting as further work for this research area. Another limitation that came up was during generation of the dataset for the first experiment. As most websites that offer research papers, such as ACM and IEEE, prevents the user of automatically downloading research papers using a crawler (according to their Robots.txt file). Because of this, the 450 different research papers used in the first experiment were downloaded manually. Downloading all these research papers manually has taken a lot of time and, therefore, the number of candidate authors tested as well as the number of research papers per author had to be limited. The last issue we want to talk about regarding limitations in our work was that we had to tweak the dataset in Experiment 2 more than initially anticipated. Some of the classification algorithms couldn't handle a large set of candidate authors and instances, and we did not get a chance to legitimately test the system in an optimal setting. Given these limitations and the fact that authorship identification on a large set of candidate authors showed mediocre results we advice that even though we have shown that authorship identification is possible you should think twice before applying and relying on authorship identification before, or after a double-blind review, or for an author to use authorship identification to acquire an unbiased review of a research paper.

RQ3: If authorship identification of research papers is possible, does the amount of work contributed to a research paper by an author affect the degree of authors that can be identified?

Above we have concluded that authorship identification of research papers is possible. However, how much does the amount of work contributed to a research paper have to say about whether or not we can identify the author in question? To answer this question, we hypothesized three different heuristics that we would conduct experiments on. The first heuristic would only take into account papers where the candidate author were the lead author of the paper. The second heuristic would take into account papers where the candidate author where among the top three contributors of the paper. The last heuristic would consider papers where the candidate author contributed to the paper, it did not matter whether or not the candidate author was the author that had contributed the least to the paper. In Chapter 6, we were able to determine that the amount of work contributed by the candidate author mattered as seen in the results. Our initial hypothesis where right and the lead author heuristic outperformed the top three author heuristic that again outperformed the heuristic where the author could be any of the above, or have contributed less. One experiment that we wished we could have made would have been to be able to compare the lead author heuristic with twenty papers per author versus the top three heuristic with twenty paper per author. Unfortunately due to the fact that we would have had to find twenty different research paper for each candidate author where the author would have had to be the lead author this task was not feasible.

7.2 Contributions

The contributions this thesis makes to the research area of authorship identification is a study of how authorship identification can be used to determine the authorship of research papers. It also shows techniques that needs to be improved to make authorship identification even easier, such as a better tool to extract plain text from research papers that maintains the structure of the paper. We have also seen what is feasible when choosing the size of the dataset to be classified. Furthermore, we have shown how much of a difference it makes when the candidate author has contributed a lot versus contributed a little to a research paper. We also think that we have laid a foundation of what can be given as new masters theses in the future described in the further work section.

7.3 Further work

There are several directions that can be looked at when thinking about the future of authorship identification for research papers. Below we have listed up everything we believe is appropriate recommendations for further work.

Authorship identification as a service
Being able to use authorship identification as a web service would be something that can be seen as a business opportunity. The concept is to let a customer bring their own set of training- and test data and use the service to perform authorship identification either for free or for predetermined fee. Keep in mind that for this to be realizable the authorship process would have to be overhauled and improved by a large margin.

Identify a group of researchers
As a masters thesis, the ability to determine a group of researchers rather than a single individual would be suitable. The idea is to determine a group of researchers or a research department, for example, the "author" could be the computer science department at NTNU.

Probability distributed authorship identification
In this thesis we can output a single author as the candidate author for our classification problem, this is due to a limitation in WEKA. However what could have been interesting and useful would be the ability to produce a list of possible candidates with a probability assigned to each candidate. We think that this task as well would be suitable for a new master's thesis.

PDF-extraction library that maintains structure
As mentioned in the discussion a limitation with the system implemented in this thesis was the ability to extract text from PDFs and still maintain the structure of the research paper. Even though this is not about authorship identification per se, the possibility to use a library like this would help improve the accuracy of the classifications.

7.4 Summary

Throughout this thesis we have worked against answering the research questions listed in Section 1.3:

RQ1 What are the different features and techniques used in authorship identification field when looked at as a classification problem?

RQ2 Is it possible to perform authorship identification on research papers?

RQ3 If authorship identification of research papers is possible, does the amount of work contributed to a research paper by an author affect the degree of authors that can be identified?

We have in this report presented theory, related fields, and previous studies in the research area of authorship identification. We have also shown that authorship identification of research papers has been proven possible, and we have implemented a system that can be used for this purpose. We have also presented our findings and our recommendations for further work in this area.

Appendix A

ARFF example

Below is an artificial example of how the ARFF-file format may look like.

```
@RELATION Authorship

@ATTRIBUTE totalNumberOfWords  NUMERIC
@ATTRIBUTE totalNumberOfDigits   NUMERIC
@ATTRIBUTE averageSentenceLength  NUMERIC
@ATTRIBUTE averageWordLength   NUMERIC
@ATTRIBUTE class        {BillGates,SteveJobs}

@DATA
3987,40,17.2,5.6,BillGates
4001,21,16.1,4.7,BillGates
4012,22,19.3,5.1,BillGates
2081,17,4.5,6.2,SteveJobs
1873,20,5.4,5.2,SteveJobs
2530,29,4.7,5.4,SteveJobs
```

Appendix B

Data from result graphs

The tables below show exact values of the graphs presented in Chapter 6.

	SVM	DT	KNN	RF	ANN
2 Authors	100.0	80.0	100.0	90.0	100.0
3 Authors	93.0	46.0	86.0	86.0	93.0
4 Authors	85.0	45.0	65.0	60.0	85.0
5 Authors	80.0	16.0	60.0	44.0	84.0
6 Authors	53.0	16.0	56.0	33.0	63.0
7 Authors	58.0	22.0	47.0	38.0	66.0
8 Authors	48.0	29.0	39.0	36.0	63.0
9 Authors	51.0	36.0	38.0	36.0	59.0
10 Authors	44.0	28.0	34.0	38.0	48.0
11 Authors	47.0	15.0	38.0	31.0	47.0
12 Authors	43.0	11.0	24.0	32.0	53.0
13 Authors	44.0	8.0	29.0	32.0	55.0
14 Authors	39.0	21.0	25.0	32.0	45.0
15 Authors	40.0	17.0	27.0	27.0	46.0
16 Authors	41.0	22.0	27.0	33.0	40.0
17 Authors	40.0	23.0	27.0	34.0	46.0
18 Authors	34.0	18.0	24.0	26.0	45.0
19 Authors	35.0	16.0	25.0	26.0	43.0
20 Authors	37.0	15.0	25.0	31.0	42.0

Table B.1: Accuracy of lead author 5 papers per author.

	SVM	DT	KNN	RF	ANN
2 Authors	1.0	0.79	1.0	0.9	1.0
3 Authors	0.93	0.47	0.87	0.86	0.93
4 Authors	0.85	0.46	0.63	0.58	0.85
5 Authors	0.79	0.19	0.56	0.4	0.83
6 Authors	0.51	0.16	0.55	0.3	0.61
7 Authors	0.57	0.22	0.45	0.37	0.65
8 Authors	0.49	0.28	0.39	0.35	0.62
9 Authors	0.5	0.34	0.38	0.35	0.6
10 Authors	0.44	0.28	0.35	0.37	0.48
11 Authors	0.48	0.14	0.4	0.29	0.46
12 Authors	0.44	0.11	0.24	0.31	0.54
13 Authors	0.45	0.07	0.26	0.31	0.56
14 Authors	0.4	0.21	0.23	0.3	0.45
15 Authors	0.4	0.17	0.24	0.24	0.46
16 Authors	0.43	0.22	0.25	0.31	0.41
17 Authors	0.41	0.21	0.25	0.33	0.47
18 Authors	0.35	0.18	0.23	0.24	0.45
19 Authors	0.37	0.16	0.23	0.24	0.44
20 Authors	0.39	0.16	0.22	0.29	0.43

Table B.2: F-measure of lead author 5 papers per author.

	SVM	DT	KNN	RF	ANN
2 Authors	1.0	0.6	1.0	0.8	1.0
3 Authors	0.9	0.2	0.8	0.8	0.9
4 Authors	0.8	0.27	0.53	0.47	0.8
5 Authors	0.75	-0.05	0.5	0.3	0.8
6 Authors	0.44	0.0	0.48	0.2	0.56
7 Authors	0.51	0.09	0.38	0.29	0.61
8 Authors	0.41	0.19	0.3	0.27	0.58
9 Authors	0.45	0.28	0.3	0.28	0.55
10 Authors	0.38	0.21	0.27	0.32	0.42
11 Authors	0.42	0.07	0.32	0.25	0.42
12 Authors	0.38	0.03	0.17	0.26	0.49
13 Authors	0.4	0.02	0.24	0.27	0.51
14 Authors	0.35	0.15	0.19	0.27	0.41
15 Authors	0.37	0.11	0.22	0.22	0.42
16 Authors	0.38	0.17	0.22	0.29	0.37
17 Authors	0.37	0.18	0.23	0.31	0.43
18 Authors	0.3	0.14	0.2	0.22	0.42
19 Authors	0.32	0.12	0.21	0.22	0.41
20 Authors	0.34	0.11	0.22	0.28	0.4

Table B.3: Kappa score of lead author 5 papers per author.

	SVM	DT	KNN	RF	ANN
2 Authors	100.0	57.0	94.0	94.0	100.0
3 Authors	93.0	62.0	93.0	89.0	93.0
4 Authors	89.0	65.0	78.0	73.0	86.0
5 Authors	79.0	45.0	75.0	72.0	89.0
6 Authors	70.0	37.0	65.0	58.0	81.0
7 Authors	76.0	42.0	67.0	57.0	86.0
8 Authors	74.0	44.0	70.0	64.0	78.0
9 Authors	75.0	40.0	64.0	64.0	77.0
10 Authors	67.0	30.0	59.0	57.0	82.0
11 Authors	69.0	34.0	56.0	55.0	77.0
12 Authors	66.0	27.0	53.0	54.0	77.0
13 Authors	60.0	28.0	52.0	58.0	68.0
14 Authors	62.0	24.0	48.0	51.0	71.0
15 Authors	59.0	25.0	46.0	48.0	68.0
16 Authors	57.0	28.0	46.0	51.0	68.0
17 Authors	55.0	17.0	47.0	52.0	65.0
18 Authors	57.0	23.0	47.0	52.0	70.0
19 Authors	56.0	20.0	45.0	49.0	67.0
20 Authors	53.0	24.0	43.0	52.0	67.0

Table B.4: Accuracy of lead author 10 papers per author.

	SVM	DT	KNN	RF	ANN
2 Authors	1.0	0.55	0.95	0.95	1.0
3 Authors	0.93	0.61	0.93	0.9	0.93
4 Authors	0.9	0.64	0.79	0.73	0.87
5 Authors	0.79	0.45	0.75	0.73	0.9
6 Authors	0.71	0.37	0.65	0.58	0.81
7 Authors	0.76	0.43	0.67	0.57	0.87
8 Authors	0.74	0.43	0.7	0.63	0.78
9 Authors	0.75	0.4	0.63	0.64	0.77
10 Authors	0.67	0.31	0.58	0.57	0.82
11 Authors	0.69	0.33	0.55	0.55	0.78
12 Authors	0.67	0.29	0.52	0.53	0.77
13 Authors	0.61	0.28	0.5	0.56	0.7
14 Authors	0.61	0.25	0.46	0.49	0.72
15 Authors	0.61	0.26	0.45	0.46	0.69
16 Authors	0.58	0.27	0.45	0.49	0.68
17 Authors	0.56	0.17	0.46	0.51	0.66
18 Authors	0.58	0.23	0.47	0.51	0.71
19 Authors	0.57	0.21	0.46	0.46	0.68
20 Authors	0.54	0.25	0.44	0.5	0.68

Table B.5: F-measure of lead author 10 papers per author.

	SVM	DT	KNN	RF	ANN
2 Authors	1.0	0.14	0.89	0.89	1.0
3 Authors	0.9	0.43	0.9	0.84	0.9
4 Authors	0.86	0.54	0.72	0.65	0.82
5 Authors	0.74	0.32	0.69	0.66	0.87
6 Authors	0.65	0.26	0.59	0.5	0.77
7 Authors	0.73	0.33	0.62	0.5	0.85
8 Authors	0.71	0.37	0.66	0.59	0.75
9 Authors	0.72	0.33	0.6	0.6	0.74
10 Authors	0.64	0.23	0.55	0.52	0.81
11 Authors	0.66	0.28	0.52	0.51	0.76
12 Authors	0.64	0.21	0.49	0.5	0.75
13 Authors	0.58	0.23	0.48	0.55	0.66
14 Authors	0.59	0.19	0.45	0.48	0.69
15 Authors	0.57	0.21	0.43	0.45	0.66
16 Authors	0.55	0.23	0.43	0.48	0.66
17 Authors	0.52	0.13	0.45	0.5	0.64
18 Authors	0.55	0.19	0.44	0.5	0.69
19 Authors	0.54	0.16	0.42	0.46	0.66
20 Authors	0.51	0.2	0.41	0.5	0.66

Table B.6: Kappa score of lead author 10 papers per author.

	SVM	DT	KNN	RF	ANN
2 Authors	96.0	60.0	90.0	90.0	96.0
3 Authors	75.0	51.0	71.0	75.0	84.0
4 Authors	79.0	38.0	59.0	69.0	81.0
5 Authors	67.0	32.0	48.0	55.0	74.0
6 Authors	68.0	40.0	48.0	59.0	68.0
7 Authors	64.0	29.0	47.0	50.0	75.0
8 Authors	63.0	25.0	41.0	54.0	74.0
9 Authors	54.0	31.0	37.0	52.0	63.0
10 Authors	54.0	24.0	38.0	46.0	56.0
11 Authors	56.0	25.0	37.0	45.0	62.0
12 Authors	57.0	24.0	35.0	46.0	61.0
13 Authors	52.0	18.0	31.0	43.0	56.0
14 Authors	52.0	22.0	31.0	44.0	58.0
15 Authors	48.0	17.0	28.0	44.0	56.0
16 Authors	46.0	17.0	28.0	41.0	53.0
17 Authors	51.0	13.0	28.0	41.0	58.0
18 Authors	52.0	17.0	31.0	44.0	53.0
19 Authors	47.0	15.0	29.0	41.0	52.0
20 Authors	46.0	13.0	30.0	38.0	52.0

Table B.7: Accuracy of top 3 authors 15 papers per author.

	SVM	DT	KNN	RF	ANN
2 Authors	0.97	0.6	0.9	0.9	0.97
3 Authors	0.75	0.51	0.7	0.75	0.85
4 Authors	0.8	0.39	0.59	0.69	0.82
5 Authors	0.68	0.32	0.48	0.55	0.74
6 Authors	0.69	0.4	0.48	0.59	0.68
7 Authors	0.64	0.29	0.47	0.49	0.75
8 Authors	0.63	0.25	0.4	0.54	0.74
9 Authors	0.55	0.31	0.37	0.52	0.63
10 Authors	0.58	0.25	0.4	0.46	0.56
11 Authors	0.58	0.26	0.38	0.44	0.62
12 Authors	0.59	0.25	0.36	0.46	0.61
13 Authors	0.54	0.18	0.32	0.42	0.57
14 Authors	0.54	0.22	0.32	0.44	0.59
15 Authors	0.5	0.17	0.29	0.43	0.56
16 Authors	0.48	0.17	0.28	0.41	0.53
17 Authors	0.53	0.13	0.28	0.4	0.58
18 Authors	0.54	0.16	0.31	0.43	0.52
19 Authors	0.49	0.15	0.3	0.4	0.52
20 Authors	0.49	0.14	0.3	0.37	0.52

Table B.8: F-measure of top 3 authors 15 papers per author.

	SVM	DT	KNN	RF	ANN
2 Authors	0.93	0.2	0.8	0.8	0.93
3 Authors	0.63	0.27	0.57	0.63	0.77
4 Authors	0.73	0.19	0.46	0.59	0.75
5 Authors	0.59	0.15	0.36	0.44	0.68
6 Authors	0.62	0.29	0.38	0.51	0.62
7 Authors	0.59	0.18	0.38	0.42	0.72
8 Authors	0.58	0.14	0.33	0.48	0.71
9 Authors	0.49	0.23	0.29	0.47	0.59
10 Authors	0.49	0.16	0.32	0.41	0.52
11 Authors	0.52	0.17	0.31	0.4	0.58
12 Authors	0.54	0.18	0.29	0.41	0.58
13 Authors	0.49	0.11	0.26	0.39	0.53
14 Authors	0.48	0.16	0.26	0.4	0.55
15 Authors	0.45	0.12	0.23	0.4	0.54
16 Authors	0.43	0.12	0.24	0.38	0.5
17 Authors	0.48	0.08	0.24	0.38	0.56
18 Authors	0.5	0.12	0.27	0.41	0.51
19 Authors	0.45	0.1	0.26	0.38	0.5
20 Authors	0.44	0.09	0.27	0.35	0.5

Table B.9: Kappa score of top 3 authors 15 papers per author.

	SVM	DT	KNN	RF	ANN
2 Authors	95.0	70.0	92.0	95.0	95.0
3 Authors	90.0	68.0	80.0	83.0	95.0
4 Authors	81.0	56.0	64.0	75.0	87.0
5 Authors	74.0	38.0	58.0	75.0	82.0
6 Authors	76.0	31.0	57.0	61.0	80.0
7 Authors	73.0	33.0	57.0	66.0	81.0
8 Authors	79.0	38.0	55.0	65.0	81.0
9 Authors	73.0	30.0	53.0	60.0	77.0
10 Authors	67.0	31.0	51.0	57.0	73.0
11 Authors	67.0	29.0	45.0	53.0	68.0
12 Authors	64.0	22.0	45.0	53.0	67.0
13 Authors	62.0	22.0	42.0	50.0	67.0
14 Authors	61.0	25.0	40.0	50.0	66.0
15 Authors	58.0	17.0	37.0	49.0	64.0
16 Authors	57.0	22.0	41.0	49.0	63.0
17 Authors	58.0	19.0	43.0	50.0	62.0
18 Authors	59.0	17.0	39.0	48.0	68.0
19 Authors	58.0	21.0	40.0	47.0	59.0
20 Authors	58.0	18.0	41.0	45.0	61.0

Table B.10: Accuracy of top 3 authors 20 papers per author.

	SVM	DT	KNN	RF	ANN
2 Authors	0.95	0.7	0.92	0.95	0.95
3 Authors	0.9	0.67	0.8	0.83	0.95
4 Authors	0.81	0.57	0.64	0.76	0.87
5 Authors	0.75	0.38	0.58	0.76	0.83
6 Authors	0.76	0.31	0.58	0.61	0.8
7 Authors	0.73	0.32	0.57	0.67	0.81
8 Authors	0.8	0.38	0.55	0.65	0.82
9 Authors	0.74	0.31	0.54	0.6	0.78
10 Authors	0.68	0.31	0.52	0.57	0.74
11 Authors	0.68	0.29	0.45	0.54	0.68
12 Authors	0.66	0.23	0.46	0.54	0.67
13 Authors	0.64	0.23	0.43	0.51	0.68
14 Authors	0.63	0.25	0.41	0.5	0.67
15 Authors	0.6	0.17	0.39	0.5	0.65
16 Authors	0.6	0.22	0.42	0.48	0.63
17 Authors	0.61	0.2	0.44	0.49	0.63
18 Authors	0.61	0.18	0.4	0.48	0.68
19 Authors	0.6	0.21	0.4	0.47	0.59
20 Authors	0.6	0.19	0.42	0.45	0.61

Table B.11: F-measure of top 3 authors 20 papers per author.

	SVM	DT	KNN	RF	ANN
2 Authors	0.9	0.4	0.85	0.9	0.9
3 Authors	0.85	0.53	0.7	0.75	0.93
4 Authors	0.75	0.43	0.53	0.68	0.83
5 Authors	0.68	0.23	0.48	0.7	0.79
6 Authors	0.72	0.17	0.5	0.54	0.77
7 Authors	0.69	0.22	0.5	0.61	0.78
8 Authors	0.76	0.3	0.49	0.6	0.79
9 Authors	0.7	0.22	0.48	0.55	0.75
10 Authors	0.64	0.24	0.46	0.53	0.7
11 Authors	0.64	0.22	0.4	0.49	0.65
12 Authors	0.61	0.16	0.4	0.49	0.64
13 Authors	0.59	0.16	0.37	0.47	0.65
14 Authors	0.59	0.19	0.36	0.47	0.64
15 Authors	0.55	0.11	0.33	0.46	0.62
16 Authors	0.55	0.17	0.37	0.46	0.61
17 Authors	0.56	0.15	0.4	0.47	0.61
18 Authors	0.57	0.13	0.36	0.46	0.66
19 Authors	0.56	0.17	0.37	0.45	0.57
20 Authors	0.56	0.15	0.39	0.43	0.6

Table B.12: Kappa score of top 3 authors 20 papers per author.

	SVM	DT	KNN	RF	ANN
2 Authors	90.0	67.0	77.0	77.0	95.0
3 Authors	81.0	48.0	68.0	75.0	91.0
4 Authors	78.0	46.0	60.0	68.0	82.0
5 Authors	68.0	48.0	51.0	63.0	77.0
6 Authors	70.0	35.0	43.0	62.0	75.0
7 Authors	69.0	28.0	48.0	55.0	71.0
8 Authors	66.0	27.0	42.0	54.0	64.0
9 Authors	58.0	22.0	38.0	51.0	60.0
10 Authors	54.0	22.0	34.0	50.0	61.0
11 Authors	54.0	20.0	35.0	44.0	56.0
12 Authors	57.0	24.0	34.0	44.0	55.0
13 Authors	54.0	15.0	35.0	39.0	55.0
14 Authors	51.0	19.0	35.0	45.0	53.0
15 Authors	50.0	16.0	33.0	43.0	54.0
16 Authors	48.0	17.0	31.0	42.0	48.0
17 Authors	51.0	17.0	34.0	44.0	54.0
18 Authors	49.0	14.0	36.0	38.0	55.0
19 Authors	49.0	16.0	31.0	41.0	49.0
20 Authors	47.0	14.0	34.0	39.0	51.0

Table B.13: Accuracy of random author.

	SVM	DT	KNN	RF	ANN
2 Authors	0.9	0.67	0.77	0.77	0.95
3 Authors	0.82	0.49	0.67	0.75	0.92
4 Authors	0.79	0.47	0.6	0.68	0.82
5 Authors	0.69	0.48	0.51	0.63	0.78
6 Authors	0.71	0.35	0.43	0.62	0.75
7 Authors	0.7	0.28	0.48	0.55	0.72
8 Authors	0.66	0.28	0.43	0.54	0.64
9 Authors	0.59	0.23	0.38	0.51	0.6
10 Authors	0.56	0.22	0.34	0.51	0.61
11 Authors	0.56	0.2	0.36	0.44	0.57
12 Authors	0.58	0.24	0.35	0.44	0.55
13 Authors	0.56	0.16	0.36	0.39	0.55
14 Authors	0.52	0.19	0.35	0.45	0.54
15 Authors	0.52	0.17	0.34	0.43	0.54
16 Authors	0.49	0.17	0.32	0.42	0.48
17 Authors	0.53	0.17	0.34	0.43	0.54
18 Authors	0.51	0.15	0.35	0.38	0.55
19 Authors	0.51	0.16	0.31	0.4	0.49
20 Authors	0.49	0.14	0.33	0.38	0.51

Table B.14: F-measure of random author.

	SVM	DT	KNN	RF	ANN
2 Authors	0.8	0.35	0.55	0.55	0.9
3 Authors	0.73	0.23	0.53	0.63	0.87
4 Authors	0.71	0.29	0.48	0.58	0.76
5 Authors	0.61	0.36	0.39	0.55	0.72
6 Authors	0.65	0.22	0.32	0.55	0.71
7 Authors	0.64	0.17	0.4	0.48	0.67
8 Authors	0.62	0.17	0.35	0.48	0.6
9 Authors	0.53	0.13	0.31	0.46	0.55
10 Authors	0.5	0.14	0.27	0.45	0.57
11 Authors	0.5	0.12	0.29	0.38	0.53
12 Authors	0.53	0.17	0.28	0.39	0.51
13 Authors	0.51	0.09	0.3	0.34	0.52
14 Authors	0.47	0.13	0.31	0.41	0.5
15 Authors	0.47	0.11	0.29	0.39	0.51
16 Authors	0.45	0.12	0.27	0.39	0.45
17 Authors	0.49	0.12	0.3	0.41	0.51
18 Authors	0.46	0.1	0.32	0.35	0.53
19 Authors	0.46	0.11	0.28	0.39	0.47
20 Authors	0.44	0.1	0.31	0.36	0.48

Table B.15: Kappa score of random author.

Appendix C

Setup to perform the different experiments

C.1 Experiment 1

Experiment 1 is the easiest to run. The simplest form is the use the datasets in the deliveries directly with WEKA, however the implemented system can also be used.

C.1.1 Run with datasets and WEKA

The delivered datasets can be run directly by downloading and running WEKA (version 3.6.12 was used during the experiments in this thesis). Start of by choosing the Explorer and then choose Open File. Then proceed to the Classify tab and choose the desired algorithm. In this thesis we used SMO, J48, IBk, RandomForest and MultilayerPerceptron.

C.1.2 Run experiment 1 with the implemented system

This system is built using Gradle [1]. In order to get the necessary dependencies Gradle 2.x has to be installed. After Gradle has been successfully installed proceed to build the system to download the necessary dependencies. The build.gradle file might have to be modified in order to install the local dependencies included in the deliveries. After installing the dependencies proceed to open the MainExperiment1.java file. Now some of the variables has to be

[1] https://gradle.org/

edited in order to run the experiment. The "numberOfAuthors" variable has to be set according to the number of authors in the dataset. For example if you are performing authorship identification on a two-author problem the variable has to be set to 2. When the "numberOfAuthors" variable has been set, the "author" variable is up next. The "author" variable should be the path to your datasets. The structure of the folders with the PDFs has to be structured in a specific way in order to run the program. Inside the main folder we will have a folder named "output" and folders containing the PDFs for each authors. The name of each folder directly in the main folder has to be unique in order to distinguish between the different authors. Inside each author folder two new folders has to be created, the first one named "single", and the second folder named "double". In the "single" folder single-column PDFs are stored, and in the "double" folder double-column PDFs are stored. When everything is set the MainExperiment1.java class can be executed. In the deliveries a folder named "Experiment1testfolder" illustrates the file-structure. Please note that the multilayered perceptrons algorithm can take quite a while to finish running.

C.2 Experiment 2

Experiment 2 can run any number of PDFs but can be harder to set up compared to the first experiment.

C.2.1 Setting up the database

For the second experiment a database with all the recordings from DBLP has to be set up. Instruction on how to do this can be seen in the DatabaseReadme.txt file in the AuthorshipIdentification project. The guide on how to set up the database was taken from `https://code.google.com/p/dblp-parser/`. When the database is set up, the project can be run.

C.2.2 Run experiment 2 with the implemented system

The first thing that has to be done is set up Gradle. Please see Section C.1.2. After the Gradle step is finished, the PDFs has to be validated as readable. By running the PDFChecker.java class, PDFs with less than 3 pages or has less than 1000 readable characters will be deleted. Change the variable "s" to the destination of the PDFs. Then run the class. When the PDFs has been validated, the next step is to make sure that they exist in the database, that is done by running the PDFCheckerDatabase.java file. In this file, change the variable "path" to the path of the PDFs. If the author cannot be found in the database, the file is deleted. When the PDFCheckerDatabase.java class has

finished running, the training process can begin. To train the data proceed to run the MainExperiment2.java file, change the variable "author" to the path of the trainingdata folder. After being run the program will create a training set in the "output" folder inside the trainingdata folder. When the training set is completed, it is time to create and run the test set. First verify the PDFs with the PDFChecker.java program, then open the CreateAndClassifyTestSet.java, and change the variable "mainPath" to the path to the folder containing the training- and test data. Finally run the CreateAndClassifyTestSet.java and wait for the results, this might take a while.

Bibliography

[1] Abbasi, A. and Chen, H. (2005). Applying authorship analysis to extremist-group web forum messages. *IEEE Intelligent Systems*, 20(5):67–75.

[2] Aha, D. W., Kibler, D. F., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6:37–66.

[3] Argamon, S., Koppel, M., Pennebaker, J. W., and Schler, J. (2009). Automatically profiling the author of an anonymous text. *Commun. ACM*, 52(2):119–123.

[4] Argamon, S. and Levitan, S. (2005). Measuring the usefulness of function words for authorship attribution. In *Proceedings of the Joint Conference of the Association for Computers and the Humanities and the Association for Literary and Linguistic Computing*.

[5] Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley.

[6] Baraka, R., Salem, S., Abu, M., Nayef, N., and Shaban, W. A. (2014). Arabic Text Author Identification Using Support Vector Machines. In *Journal of Advanced Computer Science and Technology Research*, volume 4, pages 1–11.

[7] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[8] Corney, M., Mohay, G. M., Andersona, A., and de Vel, O. Y. (2001). Identifying the Authors of Suspect Email. In *Computers and Security*.

[9] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

[10] de Vel, O. Y., Anderson, A., Corney, M., and Mohay, G. M. (2001). Mining email content for author identification forensics. *SIGMOD Record*, 30(4):55–64.

[11] Diederich, J., Kindermann, J., Leopold, E., and Paass, G. (2003). Authorship attribution with support vector machines. *Appl. Intell.*, 19(1-2):109–123.

[12] Frantzeskou, G., Stamatatos, E., Gritzalis, S., and Katsikas, S. K. (2006). Effective identification of source code authors using byte-level information. In *28th International Conference on Software Engineering*, pages 893–896.

[13] Gardner, M. and Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)— a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14–15):2627 – 2636.

[14] Govindan, V. and Shivaprasa, A. (1990). Character recognition — a review. *Pattern Recognition*, 23(7):671 – 683.

[15] Gray, A., Sallis, P., and Macdonell, S. (1997). Software forensics: Extending authorship analysis techniques to computer programs. In *Proceedings of the 3rd Biannual Conference of the International Association of Forensic Linguists (IAFL)*, pages 1–8.

[16] Hall, M. A., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18.

[17] Honoré, A. (1979). Some Simple Measures of Richness of Vocabulary. *Association for Literary and Linguistic Computing Bulletin*, 7(2):172–177.

[18] Hurtado, J., Taweewitchakreeya, N., and Zhu, X. (2014). Who wrote this paper? learning for authorship de-identification using stylometric featuress. In *Proceedings of the 15th IEEE International Conference on Information Reuse and Integration*, pages 859–862.

[19] Joachims, T. (1998). Text categorization with suport vector machines: Learning with many relevant features. In *Machine Learning: ECML-98, 10th European Conference on Machine Learning*, pages 137–142.

[20] Jockers, M. L. and Witten, D. M. (2010). A comparative study of machine learning methods for authorship attribution. *LLC*, 25(2):215–223.

[21] Kestemont, M., Luyckx, K., Daelemans, W., and Crombez, T. (2012). Cross-genre authorship verification using unmasking. *English Studies*, 93(3):340–356.

[22] Kjell, B., Woods, W. A., and Frieder, O. (1994). Discrimination of authorship using visualization. *Inf. Process. Manage.*, 30(1):141–150.

[23] Koppel, M. and Schler, J. (2003). Exploiting stylistic idiosyncrasies for authorship attribution. In *Proceedings of IJCAI'03 Workshop on Computational Approaches to Style Analysis and Synthesis*, pages 69–72.

[24] Koppel, M. and Schler, J. (2004). Authorship verification as a one-class classification problem. In *Machine Learning, Proceedings of the Twenty-first International Conference*.

[25] Koppel, M., Schler, J., and Bonchek-Dokow, E. (2007). Measuring differentiability: Unmasking pseudonymous authors. *Journal of Machine Learning Research*, 8:1261–1276.

[26] Maurer, H. A., Kappe, F., and Zaka, B. (2006). Plagiarism - A survey. *J. UCS*, 12(8):1050–1084.

[27] Mendenhall, T. C. (1887). The characteristic curves of composition. In *Science*, volume 9, pages 237–249.

[28] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc.

[29] Mosteller, F. and Wallace, D. L. (1964). *Inference and Disputed Authorship: The Federalist Papers*. Addison-Wesley.

[30] Nizamani, S. and Memon, N. (2013). CEAI: CCM based email authorship identification model. *CoRR*, abs/1312.2451.

[31] Platt, J. C. (1998). Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

[32] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

[33] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

[34] Ramakrishnan, C., Patnia, A., Hovy, E. H., and Burns, G. A. P. C. (2012). Layout-aware text extraction from full-text PDF of scientific articles. *Source Code for Biology and Medicine*, 7:7.

[35] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47.

[36] Stamatatos, E. (2009). A survey of modern authorship attribution methods. *JASIST*, 60(3):538–556.

[37] Stein, B., Lipka, N., and zu Eissen, S. M. (2008). Meta analysis within authorship verification. In *19th International Workshop on Database and Expert Systems Applications*, pages 34–39.

[38] Tan, P., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining.* Addison-Wesley.

[39] Tufan, T. and Görür, A. K. (2007). Author identification for turkish texts. In *Cankaya University Journal of Arts and Sciences*, volume 1, pages 151–161.

[40] Tweedie, F. J. and Baayen, R. H. (1998). How variable may a constant be? measures of lexical richness in perspective. *Computers and the Humanities*, 32(5):323–352.

[41] Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, second edition.

[42] Zhang, C., Wu, X., Niu, Z., and Ding, W. (2014). Authorship identification from unstructured texts. *Knowl.-Based Syst.*, 66:99–111.

[43] Zheng, R., Li, J., Chen, H., and Huang, Z. (2006). A framework for authorship identification of online messages: Writing-style features and classification techniques. *JASIST*, 57(3):378–393.