Internal Report 2012–18 August 2012

Universiteit Leiden

Opleiding Informatica

Authorship Attribution using Compression Distances

Ramon de Graaff

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Authorship Attribution using Compression Distances

Ramon de Graaff

Authorship Attribution using Compression Distances

THESIS

to obtain the degree of Bachelor of Science

on Friday August 24, 2012

Leiden Institute of Advanced Computer Science

Leiden University

by

Ramon de Graaff

born on April 27, 1990

in Delft, The Netherlands

This thesis has been approved by:

Dr. W.A. Kosters, LIACS, Leiden University, The Netherlands; kosters@liacs.nl
Dr. C.J. Veenman, DT&B, Netherlands Forensic Institute; c.veenman@nfi.minvenj.nl

Abstract

Authorship attribution has been a field of interest for researchers in the past, especially for forensic purposes. In this thesis, to obtain the degree of Bachelor of Science from the Leiden University, we investigate character n-grams and so-called compression distances to prototypes on several datasets, i.e., the datasets provided by PAN Labs (a benchmarking activity on uncovering plagiarism, authorship and social software misuse). This research was done in cooperation with the Netherlands Forensic Institute (NFI), who supervised this research as well. Compression distances to randomly selected prototypes from the training-corpus documents are used as feature representation, which has shown to be an elegant and powerful method in authorship attribution on large numbers of training documents. The standard compression distances to prototypes is not applicable to a dataset with a small number of samples and authors. We propose *Bootstrapped Authorship Attribution in Compression Space* (BAACS), an adaption of the standard compression distances to prototypes, to attribute authorship for datasets with a small number of samples and authors. BAACS is based on bootstrapping, a resampling method, to enrich the distribution of the dataset with more samples trying to generalize error estimation. We draw prototypes without replacement and in order to enrich the dataset with more samples, we draw samples with replacement from the small number of source documents per author. For the open class recognition tasks, we additionally set a threshold for the minimum posterior probability for any of the known classes.

Contents

1 Introduction **1**
1.1 Problem definition . . . 1
1.2 Tools . . . 2
1.3 Definitions . . . 2
1.4 Evaluation metrics . . . 3

2 Related Work **5**

3 Methodology **7**
3.1 Character n-grams . . . 7
3.2 Compression . . . 8
3.3 Compression Distances to Prototypes . . . 11
3.4 Bootstrapped Authorship Attribution in Compression Space . . . 12
3.5 Classification algorithms . . . 14

4 Experiments **17**
4.1 Dataset 2011 . . . 17
4.2 Dataset 2012 . . . 25

5 Conclusion and Future Work **31**

References **33**

Chapter 1

Introduction

This chapter gives an introduction to the problem, as well as the definition for several terms. Furthermore metrics are defined to measure the performance in the experiments.

1.1 Problem definition

The goal of *authorship attribution* is attributing unknown text documents to the correct author after analyzing source documents from all possible authors. In this paper we use the datasets provided by PAN11 [28] and PAN12 [29]. These datasets differ quite a lot. The PAN11 datasets contain multiple short messages per author written by 26 or 72 authors, while the PAN12 datasets contain only two large messages per author written by 3, 8 or 14 authors. The method proposed in [24], *Compression Distances to Prototypes* (CDP), is designed for datasets like that of PAN11 and uses compression distances to prototypes as the feature representation for the classification algorithm. Compression distances show how similar two documents are. A small compression distance indicates a similar document, while a large compression distance indicates a dissimilar document. Prototypes, in [24], are source documents in the dataset to which the compression distance is calculated for every source document in the dataset. Since CDP is designed to classify large datasets, it is not possible to use it for datasets with a small number of documents. We propose an adaption of CDP, *Bootstrapped Authorship Attribution in Compression Space* (BAACS), to classify datasets with a small number of documents per author. Its base is the same as CDP, the compression distances to prototypes are here also used as a feature representation. Moreover, it uses the same idea underlying bootstrapping, which is a resampling method for generalization error estimation. We create more samples by drawing samples from its source document. The prototypes are drawn samples as well.

In this paper we will apply the CDP and BAACS to the datasets provided by PAN11 and PAN12, respectively. However the CDP showed a good performance on its internal validation, which is a similar dataset as the dataset of PAN11, its performance on international datasets is yet unknown. Since the performance and methods used by other contestants is already published, we are able to find out how well the CDP performs compared to other methods applied to the same datasets. After the application of CDP to PAN11, we will apply our BAACS to the datasets of PAN12, where a threshold is proposed for BAACS to deal with open class problems.

1.2 Tools

The experiments for this thesis are all done by using Matlab in combination with a Matlab Toolbox for Pattern Recognition, PRTools 4.1 [11]. We make use of the implementations of several classification algorithms, which has several standard functions to compute the posterior probabilities and create labels for the test documents.

1.3 Definitions

In this section, definitions are defined which will be used throughout this paper. The alphabet used in the train- and test documents is:

$$\Sigma = \{X \mid X \text{ is a character from the English alphabet}\}$$

This English alphabet consists of characters that are either symbols, numbers or characters (with or without punctuation marks). Let A_i^k be the k^{th} document of author i. The set of documents of author i is defined as:

$$D_i = \{A_i^k \mid k = 1, 2, \ldots, n_i\}$$

where n_i is the number of documents of author i. So we have $|D_i| = n_i$.

1.3.1 Distance measures

In this paper, we often use compression distances to compute a distance between two text files. There are many variants of compression distances. However we will only use the *Compression Dissimilarity Measure* (CDM) and the *Normalized Compression Distance* (NCD), proposed in [19] and [5], respectively.

For documents x and y, the Compression Dissimilarity Measure is defined as:

$$CDM(x,y) = \frac{C(xy)}{C(x) + C(y)}$$

where $C(x)$ is the size of the compressed object x and xy is the concatenation of x and y.

The Normalized Compression Distance is defined as:

$$NCD(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}$$

where again $C(x)$ is the size of the compressed object x. The definition of $\min(x,y)$ is the minimum of x and y and $\max(x,y)$ is defined as the maximum of x and y. If $x < y$, $\min(x,y)$ returns x and $\max(x,y)$ returns y.

1.4 Evaluation metrics

In order to evaluate the performance, we use the standard information retrieval and pattern recognition metrics of *precision*, *recall* and F_1-measure. Precision P_A, for author A, is defined as the fraction of retrieved documents (retrieved-documents(A)) that are relevant (correct(A)):

$$P_A = \frac{\text{correct}(A)}{\text{retrieved} - \text{documents}(A)} \equiv \frac{TP_A}{TP_A + FP_A}$$

where TP_A (True Positive) is the number of documents that are correctly attributed to author A and FP_A (False Positive) is the number of documents that are incorrectly attributed to author A. Recall R_A, for author A, is defined as the fraction of relevant documents (relevant-documents(A)) that are retrieved (correct(A)):

$$R_A = \frac{\text{correct}(A)}{\text{relevant} - \text{documents}(A)} \equiv \frac{TP_A}{TP_A + FN_A}$$

where FN_A (False Negative) is the number of missed attributions for author A. In 1979, Rijsbergen introduced the F_1-measure [31], which is defined as the harmonic mean of recall and precision:

$$F_1 = 2 \cdot \frac{P_A \cdot R_A}{P_A + R_A}$$

Note that F_1 depends on author A. In order to aggregate these measures over all different authors, Yang introduced two measures, micro- and macro-averaging [39]. Micro-average is averaging over the number of authors, while micro-average is averaging over the number of documents. Given a metric M (*precision*, *recall* or F_1), for a set of n authors, these measures are defined as:

$$\textit{macro-average}_M = \frac{1}{n}\sum_{i=1}^{n} M_{A_i}$$

$$\textit{micro-average}_M = \frac{1}{k}\sum_{i=1}^{n} |D_{A_i}| \cdot M_{A_i}$$

where k is the total number of test documents and M_A the metric (*precision*, *recall* or *F1*) of author A. Micro-averaging weights the performance of authors on the basis of their number of test documents. Macro-averaging gives the same weight to every author, no matter how many test documents they wrote.

Chapter 2

Related Work

The goal is to attribute texts to their corresponding authors. In order to do that, you will need a documents from each author to train a classifier. There are many ways to collect features which describes the characteristics of the texts or authors.

The basic principles in authorship attribution and an overview of the state-of-the-art techniques are described in [18, 37]. However, in this paper we only use the character n-grams because of their simplicity and high accuracy when using them as feature vectors in classification algorithms and the method *Compression Distances to Prototypes* (CDP), proposed in [24]. The use of character n-grams for text attribution was proposed in [20]. Some papers use only the 26 characters of the West-European alphabet [22], while others use punctuation in their methods [7]. Applying character n-grams has shown to be easy, but highly accurate in combination with classifiers such as Support Vector Machines [16]. While this is still a quite simple approach, it competes with a lot of other (more complex) approaches. Other classification algorithms have been applied as well, as reported in [8, 10, 23]. Since every character n-gram is valuable for text classification, feature selection does not seem a good idea. However, the more features are extracted, the more sparse the data could be. Most of the classifiers are not able to handle a large number of dimensions. Moreover, if they can, the computation time is quite large and overfitting becomes a risk. A possible way of reducing the number of features is to select the n most distinctive features. However, this is hard to compute and it is much easier to select the character n-grams by their frequency in the dataset. After ordering the features (character n-grams) on absolute frequency [16, 38] or on relative frequency [38, 40], the n most frequent character n-grams could be selected and used as feature representation in a classification algorithm.

Compression based authorship attribution is a quite new technique compared to the character n-grams. However, a lot of research has been done in this field. The main idea behind attributing texts to authors using compression algorithms, is to compress all unseen texts and compare them to all the training texts. A high compression rate indicates the same writing style. Many methods are proposed to use compression methods in order to attribute texts to their corresponding author [1, 21, 24, 27]. Some use the compression rate between documents, while others use compression distances or other approaches to attribute a text. Attractive in compression based authorship attribution is that it requires no preprocessing of the input documents. Multiple compression methods have been used to attribute and categorize texts, e.g., LZ76 [24], LZ77 [32], LZW [27, 32], RAR [27], gzip [27], PPM [3, 15, 30, 32]. Several of these compression algorithms have been combined with a k-Nearest Neighbour, Support Vector Machine and other classification algorithms. The method proposed in [24], shows excellent results with LZ76, while other methods prefer PPM over Lempel-Ziv variants [32].

Chapter 3

Methodology

As seen in Chapter 2, there are many ways to attribute authorship. This chapter will describe the methods used in this paper.

3.1 Character n-grams

Character n-grams are substrings of a document from length n. In this paper we only use character 2-grams and character 3-grams, because they have shown to be good features for English texts [37]. Bigrams (character 2-grams) are simply couples of two characters. Representing the word 'author' in bigrams, would result in {'au', 'ut', 'th', 'ho', 'or'}. Features can be created by using the relative frequency of occurrence within a text document. Trigrams (character 3-grams) are three characters long strings. Representing the word 'author' in trigrams, would result in {'aut', 'uth', 'tho', 'hor'}. As proposed in [41], we compute the relative frequency of the character n-grams per document, which means if the trigram 'the' occurs k times in a document where in total m trigrams are found, its relative frequency is $\frac{k}{m}$. We order the character n-grams in descending order by summing up all the relative frequencies. That means that the most frequent character n-gram over all trainingsdocuments is the first feature and so on. The amount of features is selected by selecting the m relative most frequently used n-grams, where m is less than or equal to the total amount of features extracted from the dataset. By intuition one could think the more features we have, the better accuracy we obtain. However, the complexity of the problem as well as overfitting could reduce the performance when increasing the number of features (dimensions).

3.1.1 Punctuation

Punctuation, in this paper, indicates all characters that are not in the normal 26-letters {'a', 'b',..., 'z'}-alphabet augmented with the space character, e.g., characters with punctuation marks, symbols and numbers. Capital letters are evaluated as lower-case-letters, because this approach has shown a better performance than treating them as separate features. The word 'The' would result in {'th','he'} and not in {'Th','he'}. The use of punctuation could lead to a better author-profile, because some authors do and other don't use them in their messages.

After selecting a number of features, we can train a classifier to attribute texts to their corresponding author. In this paper we use *Fisher's Linear Classifier* and a *Support Vector Machine* (SVM), because they have been used in the field of authorship attribution before. SVM, are used a lot in combination with character n-grams, for example in [36], as they have shown a great performance. Fisher's Linear Classifier, in turn, has not been used a lot in combination with character n-grams. However, Fisher's Linear Classifier has been used in compression distances to prototypes [24], to attribute texts. Since we are going to use both classifiers a lot, we made abbreviations for the several methods. They are stated in Table 3.1. We will use these terms throughout the paper.

Abbreviation	Description
SVM2	SVM with bigrams without punctuation
SVM2P	SVM with bigrams with punctuation
SVM3	SVM with trigrams without punctuation
SVM3P	SVM with trigrams with punctuation
FLC2	Fisher's linear classifier with bigrams without punctuation
FLC2P	Fisher's linear classifier with bigrams with punctuation
FLC3	Fisher's linear classifier with trigrams without punctuation
FLC3P	Fisher's linear classifier with trigrams with punctuation

Table 3.1: Explanation on the abbreviations

3.2 Compression

As described in [26], compression is about finding the shortest sequence of bits needed to represent a text. There are two ways of compressing data, lossless data compression and lossy data compression. Most likely, for text compression we want lossless data compression, because the original text can be reconstructed from the encoded file. All data compression algorithms consist of two parts, a model which estimates the probability distribution (which characters/words are more common than others) and a coder which assigns the shortest codes to the most likely character. There are several efficient coding algorithms known. In 1948, Shannon proved that the best coding for a symbol, which can be decoded again, with probability p is to assign a code of length $\log_2 \frac{1}{p}$ [33]. The best compression possible for string x is to find a program that outputs x. However, Kolmogorov proved that there is no general procedure that finds the shortest program to regenerate the string we want to compress. Models can be static and adaptive: static models compute the probability distribution after processing the input, while adaptive models try to predict a probability distribution for the next symbol on the basis of already processed input.

3.2.1 Compressors

As described in the previous section, a compression algorithm is a combination of a model and a coder. In the next paragraphs, the methods used by the compressors used throughout this paper are briefly described.

LZ76 The Lempel-Ziv algorithms adaptively build a dictionary, based on the text seen previously. In this paper we use the LZ76, proposed in [25]. LZ76 is based on the idea of randomness, where it is only interested in coding. It is an adaptive dictionary coder, which efficiently stores every substring that has been used in the past, in the dictionary. It searches for new phrases, where they are stored in the dictionary in chronological order. If the processing symbol is already in the dictionary, it adds the next symbol in the sequence of the to be encoded string and searches again in the dictionary. This is continued until a new sequence of symbols has been found and that sequence will be added in the dictionary. For example, when we would encode the string 'abbacbabcabbcba', the dictionary items would be D = {'a','b','ba','c','bab','ca','bb','cb'}. In Table 3.2, processing the string 'abbacbabcabbcba' is illustrated.

PPMd Prediction with Partial Matching (PPM) is the best-known context modeling based algorithm, first proposed in [6]. The main idea behind this algorithm is to use contexts to determine the probability of the symbol being encoded. Instead of computing probabilities for all possible combinations of contexts, we only compute the probabilities for the context which has already been seen before. The symbols which have never been seen before, in the largest context, trigger the escape sequence. The algorithm then attempts to use a smaller context, until the symbol was found in a context or it was never seen in any context. In this last case, a probability of $\frac{1}{M}$ is assigned to the probability of this symbol, where M is the size of the alphabet. For example, when we would like to encode the letter 'o' in the string 'authorship', we would attempt to look if 'o' has previously occurred in the context of 'auth'. If this fails, we would encode an escape and attempt to search for 'o' in the smaller context of 'uth'. This is continued until the symbol is found or it has not been found in any context. If this is the case, the model of order -1 is used, where all letters in the alphabet of the to be encoded file

Position	Symbol	Add to dictionary	Index	Recognized as
1	a	a	1	null, a
2	b	b	2	null, b
3	b			
4	a	ba	3	2,a
5	c	c	4	null, c
6	b			
7	a			
8	b	bab	5	3, b
9	c			
10	a	ca	6	4, a
11	b			
12	b	bb	7	2, b
13	c			
14	b	cb	8	4, b
15	a			

Table 3.2: LZ76 example of processing the string 'abbacbabcabbcba'

have the same probability. When encoding an escape, the probability for this is hard to compute since it was never used before. This is called the zero-frequency problem. There are multiple variants which give the escape symbol a count of one, where others assign the number of symbols in that context to the escape count. Another approach, similar to the number of symbols as an escape count, is reducing every symbol count by one and assigning the number of symbols in that context to the escape count. There is also a variant, "The exclusion principle", in which the number of symbols in order-$(n-1)$ can be reduced because the substring did already occur in order-n, where n is the highest order. For example, if the symbols 'i' and 'o' occur in the context 'ab' of order-n, but the model has escaped from that order, the smaller order with symbols 'i' and 'o' in context 'b' of order-$(n-1)$ can be eliminated, because they would have been captured by order-n.

We give an example of processing the string 'abbacbabcabbcba', with a order-1 PPM algorithm. The context orders are shown in Table 3.3 where the escape sequences are counted by the number of symbols in that context. The order number is the number of context characters. In the order-1 context in Table 3.3, the context is the first character in the Table followed by a dash and hopefully the to be encoded character. If not, the escape sequence is triggered.
If this string was followed by the letter 'a', encoding 'a' would get the probability of $\frac{1}{3} \cdot \frac{5}{18} = 0.0926$. With the exclusion principle, encoding 'a' would get the probability of $\frac{1}{3} \cdot \frac{5}{8} = 0.2083$, because the symbols in order-0 are discarded.

Order 1			Order 0			Order -1		
Prediction	C	P	Prediction	C	P	Prediction	C	P
a-b	3	$\frac{1}{2}$	a	5	$\frac{5}{18}$	a	1	$\frac{1}{3}$
a-c	1	$\frac{1}{6}$	b	7	$\frac{7}{18}$	b	1	$\frac{1}{3}$
esc	2	$\frac{1}{3}$	c	3	$\frac{3}{18}$	c	1	$\frac{1}{3}$
b-a	3	$\frac{3}{10}$	esc	3	$\frac{3}{18}$			
b-b	2	$\frac{1}{5}$						
b-c	2	$\frac{1}{5}$						
esc	3	$\frac{3}{10}$						
c-a	1	$\frac{1}{5}$						
c-b	2	$\frac{2}{5}$						
esc	2	$\frac{2}{5}$						

Table 3.3: PPM example with one order processing the string 'abbacbabcabbcba', where the escapes are counted to the number of symbols in that context. The column C shows the count and the token P gives the probability

In this paper, we use the PPMd implemented by Shkarin [35], which is based on the basic PPM [17]. However, it uses a complex secondary escape estimation (SEE) model and considers three cases: binary context, nm-context and m-context. After modeling, PPMd uses arithmentic coding to compress the files.

3.3 Compression Distances to Prototypes (CDP)

Lambers and Veenman proposed a method called *Compression Distances to Prototypes* (CDP) [24]. This method creates a feature vector from the compression distances between the training texts. These feature vectors are the characteristics of the samples and could become as long as the number of objects in the training set. Lambers and Veenman propose to choose a subset of the training texts to compute the compression distances to, from now on referred to as prototypes. These prototypes are chosen randomly, as experiments have shown it does not significantly matter which of the training texts are chosen as a prototype. The number of prototypes, however, is an important parameter which needs tuning. The distance measure proposed in [24], is the "Compression Dissimilarity Measure". For practical reasons, the compression method LZ76 was chosen in [24]. Lambers and Veenman selected the Fisher's Linear Classifier for all methods and experiments, because its performance is competitive, while it has also computational attractive properties.

There are several parameters to be optimized:
- Number of prototypes
- Compression method
- Compression distance measure

- Classifier

3.4 Bootstrapped Authorship Attribution in Compression Space (BAACS)

Bootstrapping is a resampling method which tries to enrich the distribution of the dataset with more samples and tries to generalize error estimation [12]. The approach we propose to attribute authorship for a dataset with a small number of samples is based on this idea in combination with compression distances to randomly chosen prototypes. One prototype is drawn per corresponding source document, where this prototype is a part from its original. For practical reasons we draw the prototype from the first x percent from its document, where x is a percentage between 0 and 100. From the remaining $(100 - x)\%$ of the document, we draw samples randomly and with replacement. The number of samples is a parameter which needs optimization. Depending on the length of the drawn samples, the overlap and dependence between the samples increases. In case a sample would read over the end of the document, it would continue reading at the starting point of the remaining part of the source document until the required length was obtained. For all these samples, a compression distance was computed to all the prototypes. After we gained these feature vectors, we can learn a classifier in this compression distance space. If the prototypes are not drawn from every document in the dataset, the remaining documents are used for sampling and the compression distances are computed to all the other prototypes.

There are several parameters to be optimized:
- Number of samples per source document
- Percentage of prototypes
- Percentage of samples
- Number of prototypes
- Compression method
- Compression distance measure
- Classifier

3.4.1 Closed class

Closed class authorship attribution assumes that the author of the unknown text should be one of the candidate authors. In a balanced dataset, a n-fold cross validation can be used for internal validation, where n is the number of source documents per author. The source documents are $n - 1$ times used as a train document and one time as a test document. The test documents can be divided into multiple parts, to enhance more differentiated performances between the different settings of parameters.

3.4.2 Open class

Open class authorship attribution assumes that the author of the unknown text is either one of the candidate authors or someone else, i.e., the author of the unknown text is possibly not included in the set of candidate authors. The approach for the OPEN SET-problem is the same, but the labeling of the test documents is slightly different. We came up with a simple idea to make use of the model's estimated posterior probabilities. We set a threshold and if the posterior probability is lower than the threshold, it will be marked as "Unknown".

Suppose there are n authors and $2n$ documents in the datasets, where all authors have two documents in the train set. We train on all first documents of $n-1$ authors and leave all documents of the n^{th} author out of the train set. So the classifier is trained without the n^{th} author having any documents in the train set. In the validation set, we label the second document of the n^{th} author in the ground truth as "Unknown" and all the second documents of the other authors to their corresponding label. Every author is once marked as "Unknown" and every document is once left out as "Unknown". We determine all posterior probabilities for every test document we offered to the trained models, those values will be the thresholds to be tested on. After determining the possible thresholds, we need to come up with the threshold that performs "the best". Now we do the same experiment with offering every document of every author once as "Unknown", but we measure the performance for every threshold. We measure the performance in two ways, one where the precision, recall and *F1*-score of the unknown author are treated as any other author (the performance computed this way is referred to as *PN*) and one where the precision, recall and *F1*-score of the unknown author weights the same as the precision, recall and *F1*-score averaged over all the other authors (the performance computed this way is referred to as *P50*). The micro- and macro-average performance measures in Chapter 1 need adaption to compute the *P50*.

Given a metric M (*precision*, *recall* or *F1*) with the n^{th} author marked as "Unknown", for a set of n authors, these new adapted measures are defined as:

$$\textit{macro-average-weigthed}_M = \frac{1}{2n-2} \cdot \sum_{i=1}^{n-1} M_{A_i} + \frac{1}{2} \cdot M_{A_n}$$

$$\textit{micro-average-weighted}_M = \frac{1}{2k} \cdot \sum_{i=1}^{n-1} (|D_{A_i}| \cdot M_{A_i}) + \frac{1}{2} \cdot M_{A_n}$$

where k is the total number of test documents and M_A the metric (*precision*, *recall* or *F1*) of author A. The "known" classes will be treated according to the regular macro- and micro averages. Every document which is offered to the model as "Unknown" in combination with a threshold will provide a *PN* and a *P50*. Since there will be $2n$ documents are offered as "Unknown", there will be $2n$ times a *PN* and *P50* per threshold. These n performances are averaged over the number of documents, which will define the mean *macro-average-precision*, mean *macro-average-recall*, mean *macro-average-F_1*, mean *micro-average-precision*, mean *micro-average-recall* and mean *micro-average-F_1* for a specific threshold. The average taken over these metrics will give one value per threshold, referred to as the mean performance threshold. There will be two best mean performing thresholds, one for *PN* and one for *P50* are from now on referred to as *PN-T* and *P50-T*, respectively.

3.5 Classification algorithms

Throughout this paper, we use several classification algorithms. In the next subsections we describe the basic ideas of the two most frequently used classifiers in this paper. Further details can be found in the given references.

3.5.1 Support Vector Machine

Support Vector Machines (SVM) are proposed in [4], where the main idea is to find the optimal hyperplane that separates two or more classes. In a 2-dimensional space, let each class have a set of vectors which defines their position in the space. The set of vectors is said to be optimally separated if the distance of the closest vector to the hyperplane is maximal and it is separated without error. The margin of a SVM is the distance of the separating hyperplane to the hyperplanes of the closest vectors to that hyperplanes on both sides. However, this will only hold for linear separable data. The SVM for linear unseparable data needs adaption, where data points on the incorrect side of the margin boundary receive a distance dependent penalty. Maximizing the margin is dependent of the C-parameter which controls the trade-off between the factor to the penalty and the size of the margin. This parameter needs parameter tuning, because of its dependence on the dataset. More details can be found in [9].

3.5.2 Fisher's Linear Classifier

Fisher's Linear Discriminant was first proposed in [14], where *Fisher's Linear Classifier* (FLC) was later adapted from. The main idea of this linear classifier is, finding a projection to a line in a way that samples from different classes are well separated. This is done by measuring the separation between projections of different classes. Let $v \cdot x$ be the projection of sample x onto a line in direction v. Suppose we have a 2-class problem and take the mean of the projections per class, say M_1 and M_2. The biggest difference ($|M_2 - M_2|$) between these values, would seem a good option to be used for classification: the bigger the difference the better it would classify. However, it might be inaccurate when the variance is large and the classes still overlap. This difference measure ($|M_1 - M_2|$) needs to be normalized by a factor which is proportional to variance. This factor is defined by the scatter of the class, the spread of data around the mean. By computing the scatter and mean of the classes, the optimal projection line can be computed. After this, a threshold is set to determine the class the sample will be attributed to. More details can be found in [2].

Chapter 4

Experiments

In this section all experiments will be discussed, i.e., the results on how different feature sets would have performed on the datasets of PAN11 ass well as the internally validated and actual performance on the datasets for the different tasks of PAN12.

4.1 Dataset 2011

PAN11 [28] had different subtasks within the Author Identification task, therefore different datasets have been provided. These sets consist of English emails of the Enron-database, made public by the Federal Energy Regulatory Commission [13]. The datasets with the names "Small" (SmallTrain, SmallValid and SmallTest) and "Large" (LargeTrain, LargeValid and LargeTest) are in this paper used for experiments. The author sets are disjoint. The number of authors, number of documents and sizes of the datasets used in the PAN11-Lab are shown in Table 4.1.

Some parts of the texts are not written in English. Personal names and email addresses in the corpus have been replaced by `<NAME/>` and `<EMAIL/>` tags, respectively. However these replacements are not perfect and some texts may still contain personal names or email addresses. The authorship was determined by the email addresses, the assumption we make here is that a unique email address belongs to a specific person. However, email addresses can be used by multiple persons, so that assumption could be wrong in some cases. On the other hand, multiple email addresses can be used by one person, so there could be two of the same authors in the dataset. The name or email address of the author are replaced by an author ID, that means the dataset is labeled by unique ID's. An example of a message is shown in Figure 4.1.

Dataset	No. authors	No. documents	Total size (kB)
SmallTrain	26	3001	1365
SmallValid	23	518	191
SmallTest	23	495	193
LargeTrain	72	9337	3924
LargeValid	66	1298	1862
LargeTest	64	1300	529

Table 4.1: Dataset 2011 The number of authors and number of documents per dataset

```
Hi

Thanks for the invitation − the day sounds great. <NAME/> and <NAME/>
    will be there with bells and whistles (I have a friend coming in for
    the weekend so I won't be able to make it but thrilled that <NAME/>
    and <NAME/> are in the same class again). Talk to you soon.
```

Figure 4.1: Document 1 of author ID 255029

Two types of feature sets have been tested on the datasets of PAN11: character n-grams and *Compression Distances to Prototypes* (CDP)[24]. The mean performance is the average of the *macro-average-precision*, *macro-average-recall*, *macro-average-F_1*, *micro-average-precision*, *micro-average-recall* and *micro-average-F_1*.

4.1.1 Small dataset

The "Small"-dataset contains 3001 documents written by 26 different authors, 518 documents by 23 authors and 495 documents by 23 authors, used for training, validation and testing, respectively, so not all authors have documents in SmallValid and SmallTest. The distribution of the training-, validation- and test-corpus over the different authors is shown in Figure 4.2. Some of the authors have a lot of messages in the dataset, while others only have a few.

4.1.1.1 Character n-grams

As described in Chapter 2, character n-grams have been often used in the past in the field of authorship attribution. To determine a baseline for our experiments, we

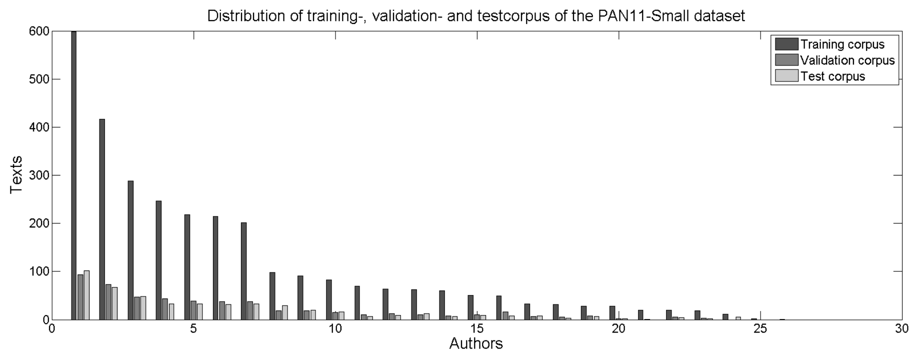

Figure 4.2: PAN11 Distribution of training-, validation- and test-corpus of the PAN11-Small dataset

experiment with these features first. As described in Section 3.1, we order the character n-grams using their relative frequency, in descending order.

Bigrams As described in Chapter 3, we take pairs of two characters from the English alphabet augmented with the space character. That will give us at most $27^2 = 729$ features. However, not all bigrams are in the dataset, that means 654 bigrams are extracted from the "Small"-dataset as features which are used by FLC2 and SVM2. The performance of FLC2 and SVM2 on the "Small"-dataset is shown in Figure 4.3. When punctuation, described in Section 3.1.1, is taken into account 2124 bigrams are extracted and are used by FLC2P and SVM2P. The use of punctuation for bigrams in the "Small"-dataset has shown to be effective as FLC2P and SVM2P outperformed FLC2 and SVM2, respectively, as clearly shown in Figure 4.3. The SVM2P outperforms FLC2P averaged over the number of features by 2.60 percentage points.

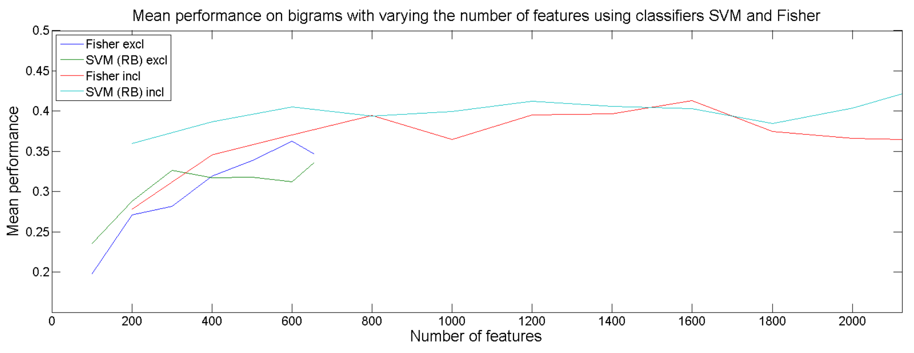

Figure 4.3: Punctuation on bigrams A comparison of the mean performance with varying the number of bigrams, trained on SmallTrain and validated on SmallValid

Trigrams When extracting the trigrams without punctuation we get at most $27^3 = 19,683$ features, in this dataset 6964 trigrams are discovered. When we extract trigrams with punctuation, $16,296$ trigrams are provided. We took a closer look at trigrams in combination with classifiers in this dataset. Based on the results in Figure 4.4, trigrams with punctuation are chosen for the comparison, because FLC3P and SVM3P outperform FLC3 and SVM3, respectively. In Figure 4.5, the mean performance for five classification algorithms is shown using trigrams with punctuation as features. The SVM outperforms the rest of the classification algorithms, although it competes with Fisher's Linear Classifier. The use of SVM3P, averaging over the number of features, improves the mean performance of SVM3 by 6.00 percentage points.

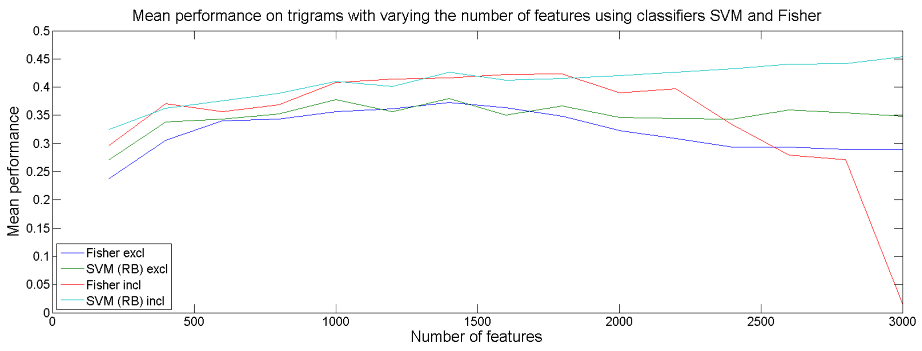

Figure 4.4: Punctuation on trigrams A comparison of the mean performance with varying the number of trigrams, trained on SmallTrain and validated on SmallValid

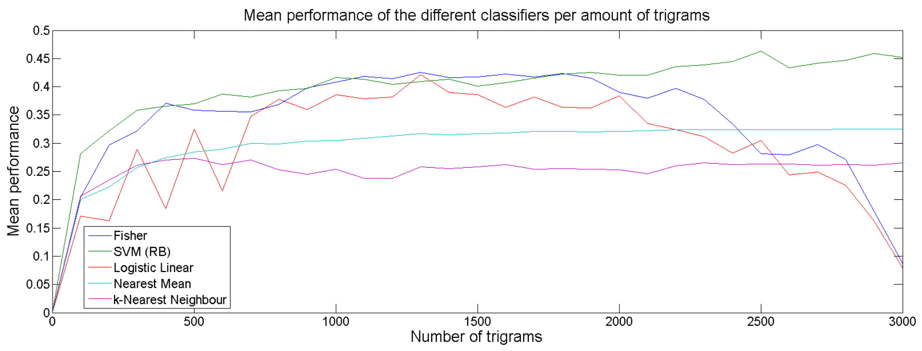

Figure 4.5: Performance on the Small Dataset Mean performance of different classifiers with varying the number of punctuated trigrams, trained on SmallTrain and validated on SmallValid

4.1.1.2 Compression Distances to Prototypes

In the method proposed in [24], there are three parameters that need to be set, i.e., the number of prototypes used to train the classifier, the distance measure and the compression method. The classifier that is used is the Fisher's Linear Classifier, as proposed in [24] in combination with the compression method LZ76 [25], however PPM proposed in [17] shows a better performance than LZ76 in [32]. We used the PPM implementation by Shkarin proposed in [35], available at [34]. Both compression methods and the two distance measures, *CDM* and *NCD*, are used to test the performance. By increasing the number of prototypes with 100 prototypes every step and measuring the mean performance of all the four combinations of LZ76 and PPMd with *CDM* and *NCD* at every step, we get Figure 4.6. Because the prototypes have a random permutation, these experiments have been repeated ten times with different orders of prototypes to stabilize the results. Experiments with a smaller stepsize, on which Figure 4.6 is based, show a maximum mean performance at 2520 prototypes, *NCD* as the distance measure and PPMd as the compression method. Although *NCD* outperforms *CDM* for both compression methods, they compete to each other. Remarkable is that *NCD* with PPMd constantly increases performance, however it is slightly worse until 2400 features. After that, the performance of *CDM* begins to drop, while *NCD* is still increasing and begins to drop after 2700 features. The same applies to compression method LZ76, where both compression distances compete and *NCD* is slightly worse until 1000 features. The validation and test results are shown in Table 4.2. These performances would, with a ranking number of 17, result in a shared 2th place in the PAN11 contest, performing on the closed "Small"-dataset in the Authorship Identification task.

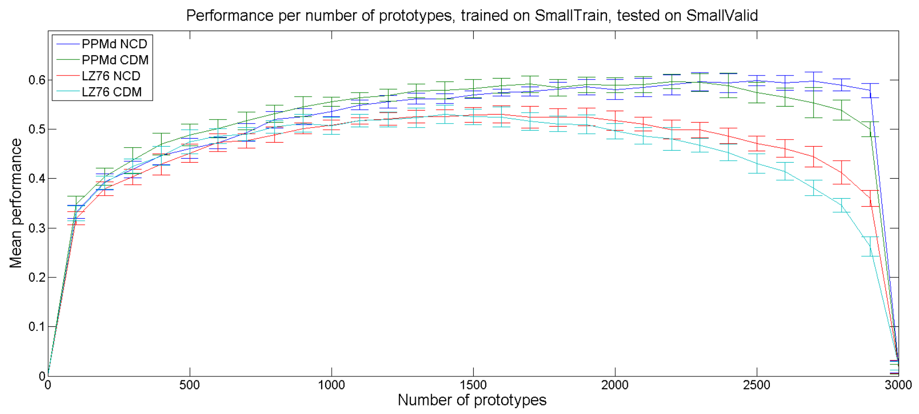

Figure 4.6: Prototypes on the Small Dataset Mean performance and standard deviation for a 10-repeat experiment with varying number of prototypes, trained on SmallTrain using Fisher's Linear Classifier and validated on SmallValid

		Macro-average			Micro-average		
Dataset	No. features	Precision	Recall	F_1-measure	Precision	Recall	F_1-measure
SmallValid	2520	0.6089	0.4530	0.4742	0.7185	0.7297	0.6934
SmallTest	2520	0.5303	0.4308	0.4477	0.6928	0.7152	0.6861

Table 4.2: Performance on Small Dataset The performance on SmallValid and SmallTest using Fisher's Linear Classifier

4.1.2 Large dataset

The "Large"-dataset contains 9337 documents written by 72 different authors, 1298 documents by 66 authors and 1300 documents by 64 authors, used for training, validation and testing, respectively. For more details, see Section 4.1. The distribution of the training-, validation- and test-corpus is shown in Figure 4.7.

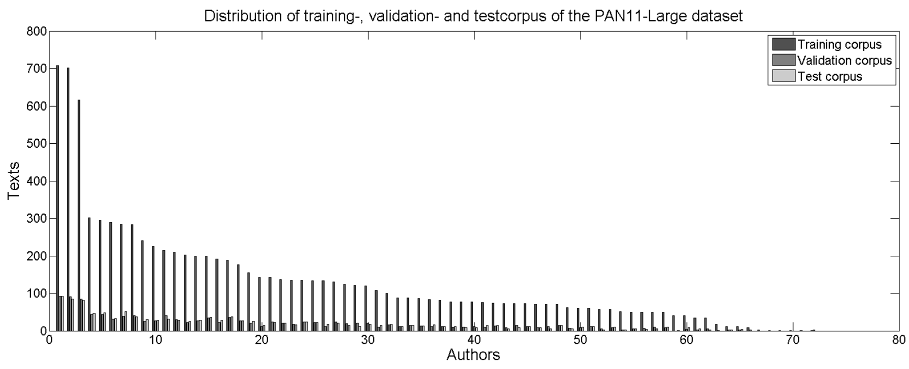

Figure 4.7: PAN11 Distribution of training-, validation- and test-corpus of the PAN11-Large dataset

4.1.2.1 Character n-grams

We determine a baseline for the experiments with compression distances to randomly chosen prototypes by using character n-grams ordered using their relative frequency first.

Bigrams Punctuation, as described in Section 3.1.1, has shown in Section 4.1.1.1 that using it for bigrams in the "Small"-dataset of PAN11 improves the performance. Without punctuation, 700 bigrams can be extracted from the "Large"-dataset, while using punctuation provides 2783 bigrams. In this dataset the performance with punctuation is again better than without punctuation. The performance for these features is also shown in Figure 4.8. Again, SVM2P performs on averaging over the number of features better than FLC2P, now with 3.55 percentage points.

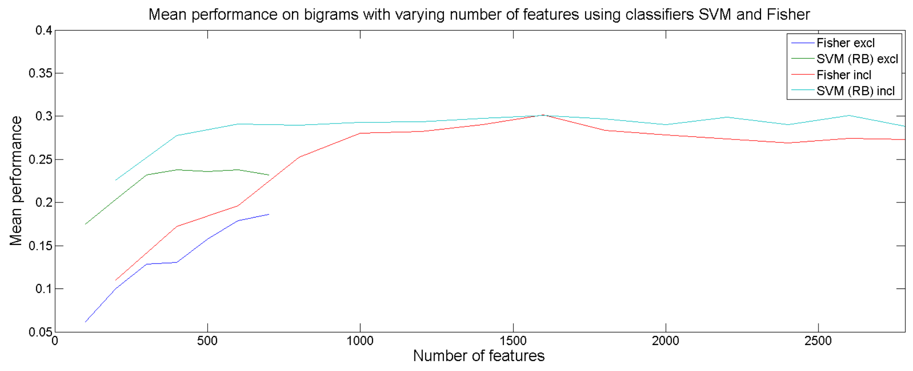

Figure 4.8: Punctuation on bigrams A comparison of the mean performance with varying the number of bigrams, trained on LargeTrain vand validated on LargeValid

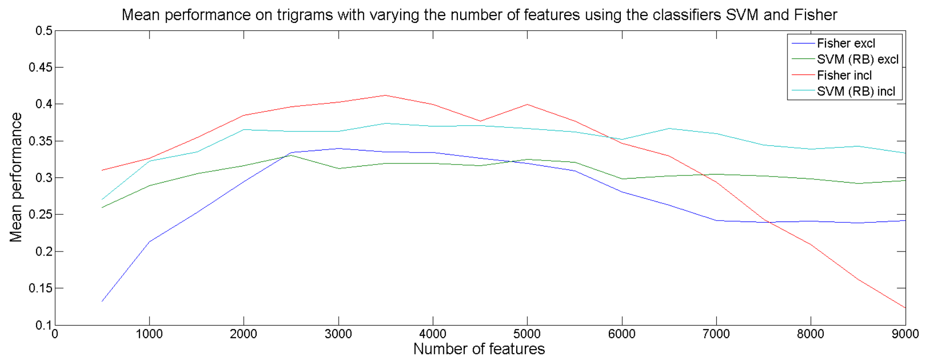

Figure 4.9: Punctuation on Trigrams A comparison of the mean performance with varying the number of trigrams, trained on LargeTrain and validated on LargeValid

Trigrams Experiments have shown that the use of punctuation for bigrams in the dataset of PAN11 improved the performance. The same applies for using trigrams with punctuation on the "Large"-dataset, this is shown in Figure 4.9. When extracting the trigrams without punctuation we get at most $27^3 = 19,683$ features, in this dataset 9026 trigrams are provided. When we extract trigrams with punctuation from the "Large"-dataset, $24,025$ trigrams are discovered. Remarkable is that for the first time FLC3P clearly outperforms SVM3P on a large range of features. The maximum score is reached by FLC3P at 3500 features.

4.1.2.2 Compression Distances to Prototypes

We used both compression methods (PPMd and LZ76) and the two distance measures (*CDM* and *NCD*) in combination with Fisher's Linear Classifier to test the performance on the "Large"-dataset. By increasing the number of prototypes with 50 prototypes at every step until 5000 prototypes and measuring the mean performance for all the four combinations of LZ76 and PPMd with *CDM* and *NCD* at every step, we get Figure 4.10. Unfortunately, for computational time reasons, the results are not averaged with different permutations of prototypes and for the same reason a step size of 400 is chosen after 5000 prototypes. The results of the mean performance in Figure 4.10 show a maximum mean performance at 4900 prototypes, *NCD* as the distance measure and PPMd as the compression method. Although *NCD* outperforms *CDM* for both compression methods, like in Section 4.1.1.2, they compete to each other. Both compression distances with compression method LZ76 compete all the time, however above 2850 features *NCD* almost always outperforms *CDM*. A maximum mean performance on LargeValid is reached at 4900 prototypes as the number of prototypes, *NCD* as the distance measure and PPMd as the compression method. The validation and testing results are shown in Table 4.3. These performances would, with a ranking number of 24, result in a 4th place in the PAN11 contest, performing on the closed "Large"-dataset in the Authorship Identification task.

		Macro-average			Micro-average		
Dataset	No. features	Precision	Recall	F_1-measure	Precision	Recall	F_1-measure
LargeValid	4900	0.5323	0.3887	0.4056	0.6104	0.6079	0.5764
LargeTest	4900	0.5766	0.4212	0.4479	0.6302	0.6131	0.5896

Table 4.3: The performance on LargeValid and LargeTest using Fisher's Linear Classifier

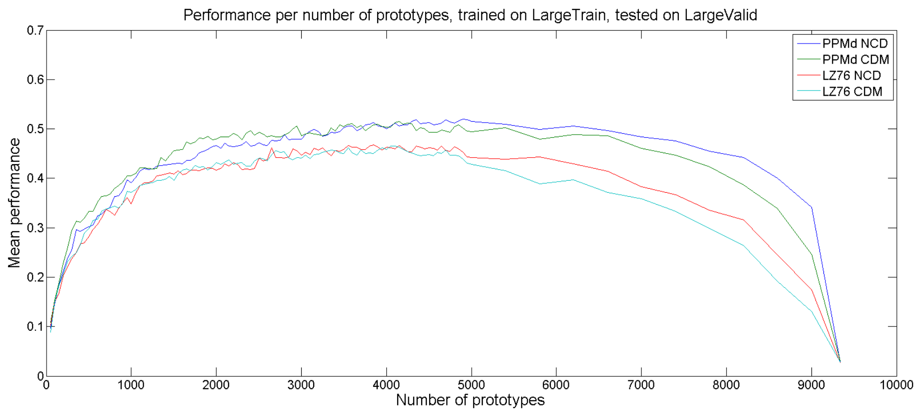

Figure 4.10: Prototypes on the Large Dataset Mean performance with varying the number of prototypes, trained on LargeTrain using Fisher's Linear Classifier and validated on LargeValid

4.2 Dataset 2012

PAN12 [29] has different subtasks within the Author Identification task. It is divided into “Traditional Authorship Attribution” and “Sexual Predator Identification”. The “Traditional Authorship Attribution”, in turn, is divided into “Traditional (closed-class/open-class)” and “Authorship clustering/intrinsic plagiarism”. The focus for this paper will be the “Traditional (closed-class/open-class)” of the “Traditional Authorship Attribution”-task.

The task description, from [29], is as follows: “Within the closed class you will be given a closed set of candidate authors and are asked to identify which one of them is the author of an anonymous text. Within the open class you have to consider also that it might be that none of the candidates is the real author of the document.”
These tasks will be performed on three different datasets (A, C and I) containing 3, 8 and 14 authors, respectively. Task A, C and I are the closed class variant, where Task B, D and J are the open class variant. Regarding the datasets, Task A, C and I have the same dataset as B, D and J, respectively. All authors in every dataset given for training have two documents.

The mean sizes and standard deviations for the datasets of PAN12 are shown in Figure 4.11. Details on the datasets in PAN12 can be found in Table 4.4. In the open class tasks, the “Unknown” author will be treated as an extra class.

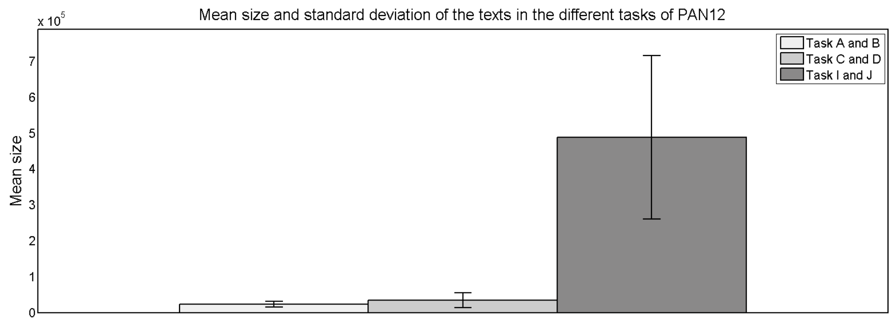

Figure 4.11: PAN12 Distribution of training- of the PAN12

These datasets required a whole different approach than the datasets of PAN11. This is because the dataset of PAN12 was very different from that of PAN11. Compression Distances to Prototypes [24] has shown a great performance in authorship attribution on large numbers of short texts, see Section 4.1, so we adapted this approach to *Bootstrapped Authorship Attribution in Compression Space* (BAASC) for the contest of PAN12. The approach is described in Section 3.4. The PAN 2012 Lab had a big challenge, because the datasets consist of only two train documents per author. No validation set is provided by PAN12, so we need to do our own validation with

Dataset	No. classes	No. documents	Size (kB)
Task A train	3	6	9-32
Task A test	3	6	5-43
Task B test	4	10	10-39
Task C train	8	16	11-72
Task C test	8	8	10-43
Task D test	9	17	10-74
Task I train	14	28	179-1023
Task I test	14	14	231-1123
Task J test	15	16	98-1271

Table 4.4: Dataset 2012 The number of authors and number of documents per dataset

those two train documents. That means we can only use one train document to train and one to validate. Fortunately, these documents are quite large as shown in Figure 4.11.

The parameters that we need to set are the distance measure, the compression method, classification algorithm, the number of prototypes, the prototype size, the number of samples and the sampling size. To narrow down these parameters, we need to make some assumptions. In previous chapters we saw that the compression method PPMd outperformed LZ76, so we take PPMd as the compression method. The distance measure *NCD* outperformed *CDM* in combination with the compression method PPMd, as seen in Sections 4.1.1.2 and 4.1.2.2. Due to time constraints and based on experiments in [32], we had to make an assumption for the distance measure far before all tests were finished. These earlier results pointed out that *CDM* was going to outperform *NCD*, so that is why *CDM* was chosen for these datasets. After these assumptions, we only have the following variables: the classification algorithm, the number of samples, the number of prototypes, the prototype size and the sampling size.

Although PAN12 said: "The performance of your authorship attribution will be judged by average precision, recall, and F1 over all authors in the given training set.", they only measured the performance by the number of correct attributions divided by all documents, corresponding to the micro-average recall. When setting the parameters, we tuned on the F_1-score, because it is the harmonic mean between precision and recall, however contributions were in the end judged on the recall.

4.2.1 Closed class

The internal validation of the closed class is done using a ten repeat 2-fold cross validation, which means using every document once in the trainset and once in the validation set is repeated ten times and that the performances are averaged over the number of repeats. To create a good estimation of the performance during the evaluation, the test documents were divided into three equal parts. The internal validation was performed on

n prototypes, where n is the number of train documents and in the internal validation equal to the number of authors as well. The number of samples is set to 30, because performance graphs, similar to Figures 4.12, 4.13 and 4.14, show a constant performance from that point on and with a larger number of samples overfitting becomes a risk.

The datasets used in these tasks consist of three, eight and fourteen authors for Task A, Task C and Task I, respectively. All authors have two documents in the dataset. The sizes of these documents differ quite a lot as shown in Table 4.4. Based on the internal validations over all combinations in the parameters, the best performance was using Fisher's Linear Classifier or *Logistic Linear Classifier* in combination with several prototype sizes and samples as shown in Table 4.5. These values are based on the graphs in Figures 4.12, 4.13 and 4.14, after choosing 30 as the number of samples. Although Fisher's Linear Classifier and "Logistic Linear Classifer" compete, we choose Fisher's Linear Classifier because it has been used before in the same field in [24].

After we tuned the parameters and estimated the performance on the internal validation, we need to make a model to release the test documents provided by PAN12 on. The only adjustable parameter now is the number of prototypes taken. There is no need for keeping documents aside for testing, so we have twice as many train documents. We can train with more train documents and create more features as well. The internal validation was done using n prototypes, where n is the number of train documents, but only half of the dataset is used because the other half is used as a test document. With twice as many train documents, we submit two models. In the first submission we use n prototypes, where n is now equal to 6, 16 and 28 for Task A, Task C and Task I, respectively. For this first submission we take a percentage of all the train documents (first and second of every author) and use them as the prototypes. From the remaining part of the train documents we draw thirty samples of a size conform the tuned parameter specified in Table 4.5. In the second submission we use 3, 8 and 14 prototypes, because that is same as used in the internal validation. The prototypes are a percentage of every first document of every author. We add samples from the documents we used as a test document in the internal validation. This creates a model with more train documents than in the internal validation, which in theory would not harm the performance. However, the performance of submission 2 is quite worse than the performance of the internal validation and submission 1. Submission 1 performs pretty well. The performances of the two submissions on the test documents provided by PAN12 are shown in Table 4.6.

			Macro-average			Micro-average		
Dataset	Prototypes	Samples	Precision	Recall	F_1-measure	Precision	Recall	F_1-measure
Task A	20%	70%	0.6356	0.7111	0.6611	0.6356	0.7111	0.6611
Task C	20%	90%	0.7718	0.8167	0.7793	0.7718	0.8167	0.7793
Task I	50%	70%	0.8125	0.8571	0.8193	0.8125	0.8571	0.8193

Table 4.5: Internal validation The internal validation and parameters on the closed-class datasets of PAN12, 10-repeat 2-fold cross validation

		Macro-average			Micro-average		
	Dataset	Precision	Recall	F_1-measure	Precision	Recall	F_1-measure
ONE	Task A	1.000	1.000	1.000	1.000	1.000	1.000
	Task C	0.8125	0.8750	0.8333	0.8125	0.8750	0.8333
	Task I	0.5952	0.7143	0.6310	0.5952	0.7143	0.6310
TWO	Task A	0.5000	0.6667	0.5556	0.5000	0.6667	0.5556
	Task C	0.5463	0.7901	0.5481	0.7598	0.4706	0.3843
	Task I	0.4167	0.5000	0.4405	0.4167	0.5000	0.4405

Table 4.6: PAN12 Lab (closed class) The performance on the closed class tasks of the PAN12 Lab for SUBMISSION 1 and SUBMISSION 2.

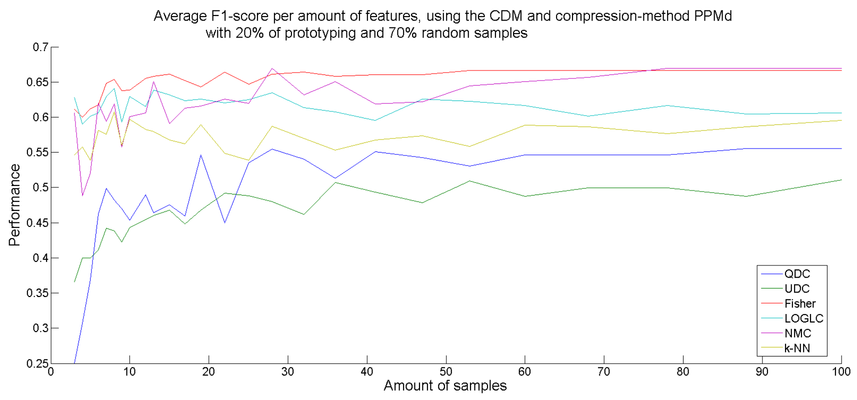

Figure 4.12: Task A and B The average F_1-score for the 10 repeat 2-fold cross validation

4.2.2 Open class

The datasets used in these tasks consist of three, eight and fourteen authors for Task B, Task D and Task J, respectively. The datasets are equal to their closed class variant. After tuning the parameters for the closed class, we need to tackle the open class problem. The internal validation on the open datasets is done using a ten repeat experiment on the dataset while measuring the performance. Per repeat, every author is k times offered as the "Unknown", where k is the number of documents per author. In Table 4.8, we see that *PN* is higher than *P50* on every dataset. This corresponds to our

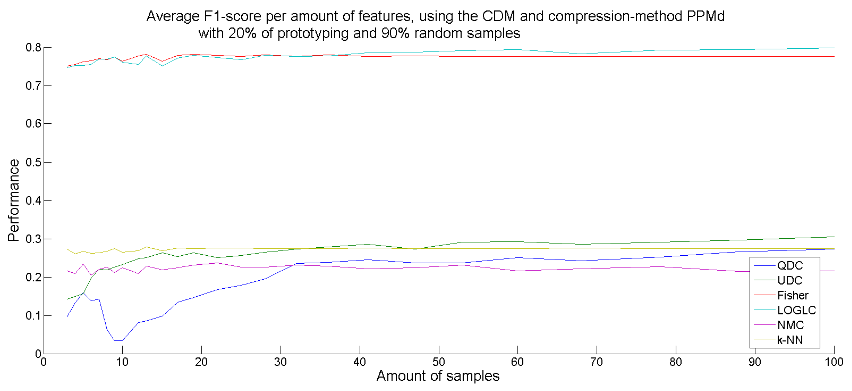

Figure 4.13: Task C and D The average F_1-score for the 10 repeat 2-fold cross validation

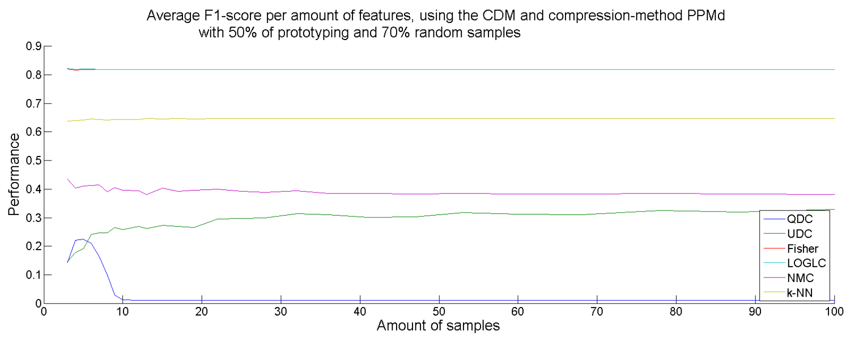

Figure 4.14: Task I and J The average F_1-score for the 10 repeat 2-fold cross validation

expectation, because here only $\frac{1}{n}$th, with n authors, is offered as "Unknown". Distinguishing the "Unknown" author is here as important as attributing the test documents to their corresponding author. We have created *P50* because we expect more "Unknown" authors in the testsets provided by PAN12 than only $\frac{1}{n}$th. With more than $\frac{1}{n}$th of "Unknown" labels for the test documents, it is getting more important to distinguish the "Unknown" author, than to attribute the documents to their corresponding author.

In Table 4.7, the number of known versus unknown authors in the testset provided by PAN12 is shown, as well as the computed thresholds. In Table 4.9 the performances are shown for the submissions on the testset provided by PAN12. After we computed the thresholds, we take the same models for SUBMISSION 1 and SUBMISSION 2 as we did for the closed class. That is, n prototypes for SUBMISSION 1 and $\frac{1}{2}n$ for SUBMISSION 2 where n is the number of train documents. As we expect more or equal documents of the "Unknown" author, we submit the models with the threshold *P50-T*. In TASK B, where the number of documents by "Unkown" authors is only four, the threshold *PN-T* would perform slightly better. Fortunately in TASK D, the number of documents by "Unknown" authors is nine, which is about half of the dataset. The threshold *P50-T* performs a lot better than threshold *PN-T*. In Task J, both thresholds came up with the same labeling for the test dataset. The models from Submission 2 came up with the same labels for both thresholds on all tasks.

Dataset	Known	Unknown	Total	*PN-T*	*P50-T*
Task B	6	4	10	0.9749	0.9812
Task D	8	9	17	$5.45 \cdot 10^{-4}$	0.0084
Task J	14	2	16	$6.90 \cdot 10^{-5}$	$4.19 \cdot 10^{-4}$

Table 4.7: PAN12 (open class) Testset Distribution of the provided test documents of the open class tasks for PAN12 Lab, including the calculated thresholds

			Macro-average			Micro-average		
Dataset	Prototypes	Samples	Precision	Recall	*F_1*-measure	Precision	Recall	*F_1*-measure
Task B *(PN-T)*	20%	70%	0.4713	0.5185	0.4840	0.4713	0.5185	0.4840
Task B *(P50-T)*	20%	70%	0.4452	0.5139	0.4681	0.4452	0.5139	0.4681
Task D *(PN-T)*	20%	90%	0.6370	0.6979	0.6398	0.6370	0.6979	0.6379
Task D *(P50-T)*	20%	90%	0.4745	0.5149	0.4763	0.4745	0.5149	0.4763
Task J *(PN-T)*	50%	70%	0.7540	0.8129	0.7651	0.7540	0.8129	0.7651
Task J *(P50-T)*	50%	70%	0.5516	0.6506	0.5854	0.5516	0.6506	0.5854

Table 4.8: Internal validation The internal validation performances on open class datasets of PAN12, 10 repeat 2-fold cross validation

		Macro-average			Micro-average		
	Dataset	Precision	Recall	*F_1*-measure	Precision	Recall	*F_1*-measure
ONE	Task B *(PN-T)*	0.7500	0.6250	0.6250	0.7000	0.6000	0.6000
	Task B *(P50-T)*	0.4750	0.5000	0.4444	0.4600	0.5000	0.4444
	Task D *(PN-T)*	0.5463	0.7901	0.5481	0.7598	0.4706	0.3843
	Task D *(P50-T)*	0.7063	0.8519	0.7500	0.7773	0.7647	0.7500
	Task J *(PN-T)*	0.7222	0.8000	0.7444	0.6771	0.7500	0.6979
	Task J *(P50-T)*	0.7222	0.8000	0.7444	0.6771	0.7500	0.6979
TWO	Task B	0.1625	0.3125	0.2054	0.1800	0.3000	0.2143
	Task D	0.1796	0.4444	0.2243	0.0951	0.2353	0.1188
	Task J	0.6167	0.7333	0.6489	0.5781	0.6875	0.6083

Table 4.9: PAN12 Lab (open class) The performances on the open class tasks of the PAN12 Lab for submission 1 and submission 2.

Chapter 5

Conclusion and Future Work

The goal of authorship attribution is attributing unknown text documents to the correct author after analyzing source documents from all possible authors. The method proposed in [24], *Compression Distances to Prototypes* (CDP), is designed for datasets with a large number of documents and authors. We adapted CDP to *Bootstrapped Authorship Attribution in Compression Space* (BAACS), where BAACS is designed for datasets with a small number of documents and authors. First, we evaluated the performance of character n-grams (with/without punctuation) in combination with several classifiers and CDP on the dataset provided by the PAN Lab 2011. After that, we optimized the parameters for BAACS using internal validation and submitted runs in the PAN Lab 2012.

The experiments in this paper on the dataset of the PAN Lab 2011 [28] show that the use of character n-grams, with $n \in \{2, 3\}$, yields a good performance with both Support Vector Machine (SVM) and Fisher's Linear Classifier (FLC). For the dataset of the PAN Lab 2011, character n-grams with punctuation should always be included in the features, as these character n-grams with punctuation outperfomed character n-grams without punctuation in combination with several classifiers. Remarkable is that combining a SVM with character n-grams, almost always outperforms FLC, except for a feature space based on trigrams with puntuation on the Large Dataset of the PAN Lab 2011. The experiments in this paper show that the use of CDP is an elegant and powerful method in attributing authorship, where it is significantly better than using character n-grams in combination with several classifiers. For large texts by a small number of authors, BAACS shows a good performance on the test documents provided by the PAN Lab 2012 [29]. Although the method proposed in [24] needed adaption, the performance is still stable and quite good for this relative simple approach. Even the performance on the open class is quite stable, although an estimation of the distribution of the test documents is needed to select a specific threshold, that is, the number of texts by an unknown author versus the number of texts by known authors that will be offered to the model.

Future Work

Since a lot of research has been done on character n-grams, that will not be the focus of the future work. Research on compression distances to randomly chosen prototypes, however, is a better focus, since not much research in this field has been done yet. Since Support Vector Machines have shown its excellent performance in combination with character n-grams and SVMs compete with Fisher's Linear Classifier, an interesting future work would be combining SVMs with compression distances to prototypes. These prototypes are randomly chosen in this paper, but selecting prototypes could probably be done in a smarter way. Given the results on the PAN Lab 2011, interesting could the performance be with the distance measure *NCD* instead of *CDM* on PAN Lab 2012. As Bootstrapped Authorship Attribution in Compression Space performs quite well on a very small dataset, we are interested in the performance on slightly larger datasets, e.g., three or four documents per author. Since PPMd outperformed LZ76 in combination with Fisher's Linear Classifier we used the compression method PPMd in the dataset of the PAN Lab 2012. Interesting would be if LZ76, according to our expectations, indeed performs worse than PPMd in combination with the same classifier. We did not adjust the number of prototypes, because it was hard to do so with only one training document, but we are interested in doing so with slightly more samples in the dataset. Moreover, instead of creating one prototype per document, creating several prototypes per document could be interesting. Regarding the drawing of prototypes, it would be interesting to draw prototypes with replacement, i.e., draw samples from the full source document, since the information of that prototype to the other prototypes is now lost. When there are more source documents available for training, it could be interesting to draw samples by combining (parts of) these source documents.

References

[1] D. Benedetto, E. Caglioti, and V. Loreto, "Language Trees and Zipping," *Physical Review Letters*, vol. 88, p. 048702, 2002.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics).* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[3] V. Bobicev, "Text Classification Using Word-Based PPM Models," *The Computer Science Journal of Moldova*, vol. 14, no. 2, pp. 183–201, 2006.

[4] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory.* ACM Press, 1992, pp. 144–152.

[5] R. Cilibrasi and P. M. B. Vit, "Clustering by Compression," *IEEE Transactions on Information Theory*, vol. 51, no. 4, pp. 1523–1545, 2005.

[6] J. Cleary and I. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Transactions on Communications*, vol. 32, no. 4, pp. 396–402, 1984.

[7] M. Corney, A. Anderson, G. Mohay, and O. De Vel, "Identifying the authors of suspect e-mail," 2001.

[8] R. M. Coyotl-Morales, L. Villaseñor Pineda, M. Montes-y Gómez, and P. Rosso, "Authorship attribution using word sequences," in *Proceedings of the 11th Iberoamerican conference on Progress in Pattern Recognition, Image Analysis and Applications*, ser. CIARP'06; LNCS 4225. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 844–853.

[9] N. Cristianini and J. Shawe-Taylor, *An introduction to support Vector Machines: and other kernel-based learning methods.* New York, NY, USA: Cambridge University Press, 2000.

[10] M. Demuth, K. Tearle, and H. Taylor, "An algorithm for automated authorship attribution using neural networks," *Literary and Linguist Computing*, vol. 23, no. 4, pp. 425–442, 2008.

[11] R. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D. Tax, and S. Verzakov, "PR-Tools4.1, A Matlab Toolbox for Pattern Recognition," 2007. [Online]. Available: http://prtools.org/

[12] B. Efron, "Bootstrap methods: Another look at the Jack-knife," *Annals of Statistics*, vol. 7, no. 1, pp. 1–26, 1979.

[13] FERC, "Federal Energy Regulatory Commission." [Online]. Available: http://www.ferc.gov/

[14] R. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[15] E. Frank, C. Chui, and I. H. Witten, "Text Categorization Using Compression Models," in *Proceedings of the Conference on Data Compression*, ser. DCC '00. Washington, DC, USA: IEEE Computer Society, 2000, p. 555.

[16] J. Houvardas and E. Stamatatos, "N-Gram Feature Selection for Authorship Identification," in *Proceedings of the 12th International Conference on Articial Intelligence: Methodology, Systems and Applications*, ser. AIMSA'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 77–86.

[17] P. G. Howard, "The Design and Analysis of Efficient Lossless Data Compression Systems," Providence, RI, USA, Tech. Rep., 1993.

[18] P. Juola, "Authorship attribution," *Foundations and Trends in Information Retrieval*, vol. 1, no. 3, pp. 233–334, 2006.

[19] E. Keogh, S. Lonardi, and C. A. Ratanamahatana, "Towards parameter-free data mining," in *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04. New York, NY, USA: ACM, 2004, pp. 206–215.

[20] V. Kešelj, F. Peng, N. Cercone, and C. Thomas, "N-gram-based author profiles for Authorship Attribution," in *Proceedings of the Conference Pacific Association for Computational Linguistics*, ser. PACLING03. Dalhousie University, NS, CA, 2003, pp. 255–264.

[21] D. V. Khmelev and W. J. Teahan, "A repetition based measure for verification of text collections and for text categorization," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, ser. SIGIR '03. New York, NY, USA: ACM, 2003, pp. 104–110.

[22] B. Kjell, "Authorship Attribution of Text Samples using Neural Networks and Bayesian Classifiers," *IEEE International Conference on Systems, Man and Cybernetics*, pp. 1660–1664, 1994.

[23] M. Koppel, J. Schler, and S. Argamon, "Computational methods in authorship attribution," *Journal of the American Society for Information Science and Technology*, vol. 60, no. 1, pp. 9–26, Jan. 2009.

[24] M. Lambers and C. Veenman, "Forensic Authorship Attribution Using Compression Distances to Prototypes," in *Proceedings of the Third International Workshop on Computational Forensics*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 13–24.

[25] A. Lempel and J. Ziv, "On the Complexity of Finite Sequences," *IEEE Transactions on Information Theory*, vol. 22, no. 1, pp. 75–81, 1976.

[26] M. V. Mahoney, *Data Compression Explained*, 2010. [Online]. Available: http://mattmahoney.net/dc/dce.html

[27] Y. Marton, N. Wu, and L. Hellerstein, "On compression-based text classification," *In Proceedings of the European Conference on Information Retrieval*, pp. 300–314, 2005.

[28] PAN11, "PAN 2011 Lab Uncovering Plagiarism, Authorship, and Social Software Misuse," 2011. [Online]. Available: http://www.webis.de/research/events/pan-11/

[29] PAN12, "PAN 2012 Lab Uncovering Plagiarism, Authorship, and Social Software Misuse," 2012. [Online]. Available: http://www.uni-weimar.de/medien/webis/research/events/pan-12/pan12-web/

[30] F. Peng, D. Schuurmans, and S. Wang, "Language and task independent text categorization with simple language models," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, ser. NAACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 110–117.

[31] C. J. V. Rijsbergen, *Information Retrieval*, 2nd ed. Newton, MA, USA: Butterworth-Heinemann, 1979.

[32] D. Sculley and C. E. Brodley, "Compression and Machine Learning: A New Perspective on Feature Space Vectors," in *Proceedings of the Data Compression Conference*, ser. DCC '06. Washington, DC, USA: IEEE Computer Society, 2006, p. 332.

[33] C. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423 & 623–656, 1948.

[34] D. Shkarin, "All about data compression, image and video." [Online]. Available: http://www.compression.ru/ds

[35] ——, "PPM: One Step to Practicality," in *Proceedings of the Data Compression Conference*, ser. DCC '02. Washington, DC, USA: IEEE Computer Society, 2002, p. 202.

[36] E. Stamatatos, “Ensemble-based Author Identification Using Character N-grams,” in *In Proceedings of the 3rd International Workshop on Textbased Information Retrieval*, 2006, pp. 41–46.

[37] ——, “A Survey of Modern Authorship Attribution Methods,” *Journal of the American Society for Information Science and Technology*, vol. 60, no. 3, pp. 538–556, 2009.

[38] Z. Wei, D. Miao, J.-H. Chauchat, and C. Zhong, “Feature selection on Chinese text classification using character n-grams,” in *Proceedings of the 3rd international conference on Rough sets and knowledge technology*, ser. RSKT’08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 500–507.

[39] Y. Yang, “An Evaluation of Statistical Approaches to Text Categorization,” *Information Retrieval*, vol. 1, no. 1-2, pp. 69–90, 1999.

[40] A. Zečević, “N-gram based text classification according to authorship,” in *RANLP Student Research Workshop*, ser. RANLP 2011, 2011, pp. 145–149.

[41] G. K. Zipf, *Human Behaviour and the Principle of Least Effort: An Introduction to Human Ecology.* Cambridge, UK: Addison-Wesley Press, 1965.