



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

*Distributed
Computing*



Style Change Detection

Bachelor Thesis

Peter Müller

`pemuelle@student.ethz.ch`

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:

Roland Schmid, Gino Brunner
Prof. Dr. Roger Wattenhofer

March 3, 2019

Acknowledgements

I would like to thank my supervisor Roland Schmid for his valuable advice during our weekly meetings, and Gino Brunner for sharing his knowledge with me on the deep learning part of the thesis.

Abstract

Style Change Detection evolves around the task of segmenting and attributing multi-authored documents. In this thesis, we look at the previously unsolved subtask of determining the exact number of authors for a given document. We present an overview of different clustering and deep learning methods and evaluate them on a shared task in the field of text forensics.

Contents

Acknowledgements	i
Abstract	ii
1 Introduction	1
1.1 Motivation	1
1.2 Related Work	1
2 Background	3
2.1 Stylometry	3
2.2 PAN Shared Task	3
2.2.1 Past PAN Style Change Detection Tasks	3
2.2.2 PAN19	4
2.2.3 Corpus	5
2.3 Intrinsic Plagiarism Detection	6
3 Clustering	7
3.1 Preprocessing	7
3.1.1 Text Segmentation	7
3.1.2 Feature Extraction	7
3.1.3 Feature (Subset) Selection	8
3.2 Clustering methods	9
3.2.1 k-means	9
3.2.2 Hierarchical Agglomerative Clustering	9
3.2.3 Determining the optimal number of clusters	9
3.3 TF-IDF	10
3.4 Clustering Temporal Data	11

4	Deep Learning	12
4.1	Background	12
4.1.1	Word Embeddings	12
4.1.2	Autoencoder	12
4.1.3	CNN	12
4.1.4	RNN/LSTM	13
4.2	Models	13
4.3	Ordinal Regression	14
4.4	Data augmentation	15
5	Results	16
5.1	Evaluation Metrics	16
5.2	Clustering	16
5.3	Deep Learning	17
5.4	Ensembling	18
5.5	PAN19 Task 1	18
5.6	Additional Results	19
6	Discussion & Future Work	20
6.1	Discussion	20
6.2	Future Work	20
	Bibliography	22
.1	Stylometric features	-1

Introduction

1.1 Motivation

Imagine you are faced with a document written by an unknown number of authors. While reading you notice that something seems off; the writing style seems to be inconsistent. However, you can not say what those inconsistencies exactly are. You might want to find out how many authors contributed to this document and who wrote which part of the document.

This task closely relates to the text forensic research field of intrinsic plagiarism detection and has potential application in multiple fields. i.E. verifying the contributions of multiple authors to a group assignment in an educational setting. Other interest in the task might be drawn from the analysis of historically relevant documents.

In this thesis, we try to solve the problem using various statistical and machine learning methods and evaluate our approaches on a shared task in the field of text forensics. Further, we try to answer the following questions: How can we quantify the writing style of an author, and is it consistent even for short segments of text? Do the writing styles of multiple authors inside a single document differ sufficiently to distinguish their contributions?

1.2 Related Work

There have been several works [1, 2] in the closely related field of intrinsic plagiarism detection, that is, the task to identify whether a document is written by a single author or contains plagiarised sections. Even though the approaches for solving this task are similar to the ones used in this thesis, their fundamental preliminary assumptions largely deviate regarding the unknown text, i.E. that there exists precisely one *main author* who wrote at least 70% of the considered text document.[3]

Other scholars [4, 5] have researched the task of clustering multiple docu-

ments, each written by a single author. However, this differs from our task in that we want to find and cluster segments within a single document, where the segmentation by author is unknown.

Akiva and Koppel have done some research on multi-authored documents. [6, 7] They mainly focus on attributing text segments to a known number of authors and leave the task of determining this number for future research.

Background

2.1 Stylometry

Stylometry is the statistical analysis of literary style. At its core lies the assumption that authors have an unconscious writing style. This writing style supposedly consists of features which are quantifiable and can ideally be used to define clear separations between different styles. Stylometry evolves around finding these quantifiable features and solving various tasks with the help of them, mainly focused on attributing documents to a known author (so called authorship attribution).

2.2 PAN Shared Task

PAN¹ is a shared task evaluation held as a part of the yearly CLEF² conference. It covers tasks on different subjects in digital text forensics and stylometry.

2.2.1 Past PAN Style Change Detection Tasks

A lot of research has already been done on the topic of authorship attribution, i.e. assigning documents to candidate authors. However, the existing approaches neglect the fact that the documents could have been written by multiple authors. The PAN Style Change Detection tasks focus on this topic of multi-authored documents. Here are some previous PAN tasks in the domain of style change detection and their results:

¹<https://pan.webis.de>

²Conference and Labs of the Evaluation Forum: <http://www.clef-initiative.eu>

PAN16 The 2016 task consists of 3 subtasks.

1. **Traditional intrinsic plagiarism detection:** Here the assumption is given that there is one main author who wrote at least 70% of the document. The task is to find the intrusive segments in the document.
2. **Diarization with a given number (n) of authors:** Segment the given document and cluster the segments using exactly n clusters containing the contributions of the n authors.
3. **Diarization with an unknown number of authors:** this task is similar to the previous task, but the number of authors n is not known.

The submissions for task 2 and 3 performed not much better than the random baseline. [8]

PAN17 1. **Style Breach Detection** Given a document, determine whether it is multi-authored and if yes, find the borders where authors switch.

The task of finding the exact character positions of where the style changes happen shows to be quite hard. None of the submission performed better than a slightly enhanced random baseline. [9]

PAN18 Since the previous task could not be solved accurately the committee relaxed the problem for the 2018 edition.

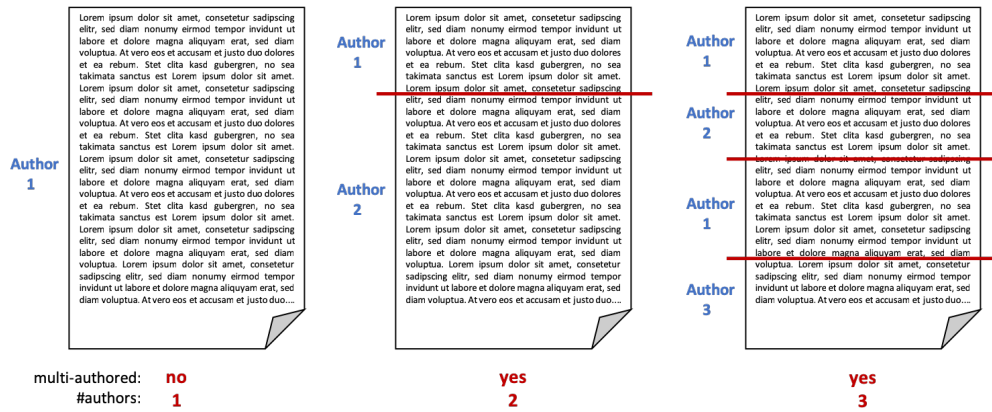
1. **Style Change Detection** Given a document, decide if the document is written by one or more authors, i.e., if there exist style changes.

This task was solved with high accuracy of 0.89 using an ensemble machine learning approach. [10]

2.2.2 PAN19

The 2019 version builds upon on the success of the previous year and is split into two connected subtasks.

1. Is a document written by one or more authors, i.e. do style changes exist or not?
2. If that document is multi-authored, how many original authors have collaborated?

Figure 2.1: PAN19 Style Change Detection Example ³

Note that the first task is the one from the 2018 PAN competition and was already solved with high accuracy.[11] The second task is a simplification of the PAN16 task 3, which only requires to find the number of authors but does not require locating the segments in the text. In this thesis, we mainly focus on solving the second task of the 2019 edition.

2.2.3 Corpus

The Corpus for this years' Style Change Detection task is based on user posts from various sites of the Stack Exchange Network. The documents contain between 300 and 2000 words and are covering different topics.

The corpus consists of a training and validation set, containing 2500 respectively 1250 document. For each problem, a ground truth with the following information is provided.

```
{
  "authors": 4,
  "structure": ["A1", "A2", "A4", "A2", "A4", "A2", "A3"],
  "switches": [805, 1552, 2827, 3584, 4340, 5489]
}
```

The keywords *authors*, *structure* and *switches* denote the number of contributing authors, a mapping of each text segment to its author and the precise character positions from the text start where the segments end respectively.

The number of authors ranges from one to five, and the class distribution for documents with (1, 2, 3, 4, 5) authors is (50%, 12.5%, 12.5%, 12.5%, 12.5%).

³<https://pan.webis.de/clef19/pan19-web/style-change-detection.html>

Note that this distribution leads to a balanced data-set for Task 1 (1 or more authors, 50-50) and Task 2 (2,3,4 or 5 authors, 25-25-25-25)

2.3 Intrinsic Plagiarism Detection

Recall the definition of the traditional intrinsic plagiarism problem as given in Section 1.2. In contrast to the extrinsic setting, where the original source document of the plagiarised text segment is available, there is no external reference collection given for this task. The plagiarism detection thus has to be performed without comparing a suspicious document to the potential sources (External Plagiarism Detection). The primary goal is therefore to find the best textual features that can distinguish the writing style of different authors in one document.

The traditional intrinsic plagiarism setting contains an essential condition, that there exists precisely one *main author* who wrote at least 70% of the considered text document. The PAN tasks on Style Change Detection drop this condition by having arbitrarily many *main authors* per document. With this addition, the task of finding and assigning text segments to their original authors gets much more difficult as we do now have to differentiate between more than just two writing styles per document and do not know the number or distribution of the authors beforehand.

Clustering

We assume that the style of an author is relatively constant over some length of text. Our goal is therefore to convert smaller text segment of a document into a vectorised form representing the writing style of that segment. These ‘style vectors’ can be clustered were we hope that each cluster corerespond to one author. Ideally, the number of clusters equals the number of authors. Similar methods have been applied by Sittar et al. [12] on the PAN16 Task¹.

3.1 Preprocessing

To cluster textual data we first need to perform the task of segmenting the documents into smaller chunks and compute a vector representation that expresses the writing style of each chunk.

3.1.1 Text Segmentation

In the first step, the document is split into sentences. We then use a sliding window approach to divide the document into smaller text chunks. By choosing our step-size smaller than the window-size, we get overlapping text chunks. This follows the intuition that the author switch happens at a sentence level. However, to accurately quantify the writing style, we need chunks which contain more than a single sentence.

3.1.2 Feature Extraction

We extract a set of well-known stylometric features (see Table .1) form each text segmennt, which have been proposed and summarised in previous work by Zheng [13] and Stein [14].

¹<https://pan.webis.de/clef16/pan16-web/author-identification.html>

Word Level Frequencies A list of 150 function-words (i.E. the, a, if) proposed by Zheng[13] is used to calculate the frequency for each function-word normalized over the length of the segment in words. The same is done for the most-common words in the document.

Stop-word Ratio The number of stop-words in the text segment divided by the number of words of the segment. We use the stop-word list of the NLTK Package.

Average Word Length The average word length in characters [15]

Word Length Distribution The frequencies of words with length 1-10 characters, normalised over the number of words in the segment.

Average Word Frequency Class This feature measures the likelihood of an author using rare used words as proposed by Eissen.[2] We use the Google Books most common word list to construct the class of each word [16]

Readability Metrics Various readability metrics. i.E. Flesh reading ease [17]

Punctuation/Special Character Frequencies The frequencies of various punctuations and special characters, normalised over the number of words in the segment.

Character Level n-gram Frequencies We first extract the most common character n-grams from the entire document. For a segment, we then calculate the relative frequencies of these most common n-grams.

Word Level n-gram Frequencies We first extract the most common word n-grams from the entire document. For a segment, we then calculate the relative frequencies of these most common n-grams.

POS Tag n-gram Frequencies We first extract the most common POS tag n-grams from the entire document. For a segment, we then calculate the relative frequencies of these most common n-grams.

3.1.3 Feature (Subset) Selection

To further improve our clustering approach we perform a feature subset selection.

Forward & Backward Selection A forward & backward elimination procedure is used to select the best features as described by Dash et al. [18] Initially, we start with the set of all features. We then remove the least discriminant feature in each round, meaning the feature which yields the biggest increase in the $f1$ accuracy score when removed from the subset and evaluated on the PAN19 task. This process is repeated until our $f1$ score no longer improves.

Forward selection works in the same manner by starting with an empty set and adding one feature after another.

3.2 Clustering methods

We evaluated many different clustering methods on the task, including DBSCAN [19], HDBSCAN [20] and Correlation Clustering [21]. K-means and Hierarchical Clustering yielded the most promising results and will thus be explained in more detail below.

3.2.1 k-means

The k-means method is defined as follows: Given an integer k and a set of n data points $X \subset \mathbb{R}^d$, the goal is to choose k center points P such that ϕ , the sum of the squared distances between each point and its closest center is minimised.

The k-means algorithm then operates as follows.

1. Choose k center points $P = \{p_1, p_2, \dots, p_k\}$ arbitrarily.
2. For each $i \in \{1, \dots, k\}$, set the cluster C_i to be the set of points in X that are closer to p_i than they are to p_j for all $j \neq i$.
3. For each $i \in \{1, \dots, k\}$, set p_i to be the center of mass of all points in C_i :

$$p_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$
4. Repeat 2 and 3 until C no longer changes.

We use k-means++ [22], which additionally optimizes the initial center seeding.

3.2.2 Hierarchical Agglomerative Clustering

Hierarchical Agglomerative Clustering [23] is a tree-based clustering algorithm. It initially assigns each data point to an individual cluster and then repeatedly selects and merges pairs of clusters based on the minimum distance between any clusters center of mass, until the desired number of clusters is reached.

3.2.3 Determining the optimal number of clusters

Most clustering algorithms require the number of clusters k as an input. A common approach for finding the optimum k is to execute a clustering algorithm for all k in a certain range and select the best result based on some clustering metric. In our case, we select the k Clustering with the highest Silhouette Index.

Silhouette Index To measure the performance of a clustering we use the Silhouette Index [24]. The Silhouette Index measures how well points are matched to their own cluster and how good they are separated from their neighbouring cluster. The Silhouette Index $SIL(C)$ of a clustering C is defined as follows.

Let us denote a partition of a data set X by $C = \{C_1, C_2, \dots, C_K\}$, where C_k indicates the k^{th} cluster in the data set with $k = 1, \dots, K$. The Silhouette of point x is defined as,

$$S(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}} \quad (3.1)$$

where $a(x)$ is the mean distance of x to all other points in the same cluster as x , and $b(x)$ being the smallest average distance of x to all points in different clusters than x .

The Silhouette Index of a clustering C is then defined as the average of the silhouettes of all points in C .

$$SIL(C) = \frac{1}{|C|} \sum_{x \in C} S(x) \quad (3.2)$$

3.3 TF-IDF

The *term frequency-inverse document frequency* (tf-idf) is a metric often used when dealing with textual data in Information Retrieval. The tf-idf reflects the importance of a single word within a document concerning an entire collection of documents.

Tf-idf is used successfully for the task of document clustering by Zhao et al. [23] We modify their original approach in the following way.

Let D be a document which is segmented into n chunks d_i using a sliding window approach. V_D is the vocabulary of t distinct words in the document D . Each chunk gets represented in the tf-idf weight model as follows:

$$d_i \rightarrow \{tf_1^i idf_1, tf_2^i idf_2, \dots, tf_t^i idf_t\} \quad (3.3)$$

where tf_1^i is the term frequency of the 1^{st} word of the vocabulary in the chunk d_i and idf_1 is the inverse document frequency of the 1^{st} word. (Note that in our case we treat the text chunks d_i as the ‘documents’ and the document D as the ‘corpus’ when calculating the tf-idf). As in the original paper, these vectors get clustered using Hierarchical Agglomerative Clustering.

We further experimented with the same method using character n-grams instead of full words.

3.4 Clustering Temporal Data

The temporal relation of textual data is an additional aspect we had to look at. Since most segments by a single author are longer than one sentence, the probability that adjacent sentences should be in the same cluster is higher than the probability that they are in different ones. In other words, there are only a few switches and longer segments. By using overlapping windows, we guide the clustering in the right direction since adjacent windows differ only by the step size and therefore share many features.

We further explored a two-step approach by first treating the sentences of a document as a graph where each sentence is a node connected to its previous and following sentence and then apply correlation clustering [21]. In a second step, we use k-means on the resulting segments to account for the case where authors have written multiple segments scattered across the document. This approach could not surpass the random baseline and was therefore not further explored.

Deep Learning

In a second approach to solving the problem, we take a look at some deep learning methods. We look at the task as a supervised text classification problem and try out some well-known deep learning models.

4.1 Background

4.1.1 Word Embeddings

A word embedding is a function $W : word \rightarrow \mathbb{R}^d$ mapping words to high dimensional vectors. There are a lot of pre-trained embeddings available which are trained on the semantics of words. This means that words which have similar semantic are close in the mapped vector space. For the reason that certain synonyms could potentially determine the style of an author, we decided to train the embeddings from scratch as part of our deep learning model.

4.1.2 Autoencoder

An autoencoder is a sequence-to-sequence model. It utilises a deep learning model to convert a sequence to a fixed size vector (encoder) and a second one to reconstruct a sequence from this vector (decoder). In our case the sequences are sentences, and we use two multi-layered LSTMs for our autoencoder model as proposed by Sutskever et al. [25]. We train the autoencoder on all sentences of the train corpus and then use the encoder to bring the sentences into vector representation for our deep learning models.

4.1.3 CNN

CNNs (Convolutional Neural Networks) are extremely succesful used for many computer vision task. They use layers with convolving filters that are applied to local features. When dealing with text, the convolutional filters are moved over

the temporal dimension and are therefore able to capture the temporal relation of the textual data. CNNs have been successfully applied to a multitude of Natural Language Processing tasks in the past [26]. One problem arises with the use of convolutional layers. Unlike fully connected layers which have the whole input as the receptive field, convolutional layers only have a small part (filter size) as their receptive field of the input. To increase the receptive field, we increase the depth of the network and use a pooling layer after each convolutional layer.

4.1.4 RNN/LSTM

A Recurrent Neural Network is a sequence of neural network blocks that are linked to each other like a chain. Each block gets one data-point of a sequence and a message from its successor block as an input. This type of neural network is especially well-suited when dealing with time series data, i.e. text. However, they are not able to forward information when the sequence gets too long.

LSTMs (Long short-term memory) are a special type of recurrent neural networks that try to solve the problem of RNNs by allowing information to be remembered for much longer [27].

4.2 Models

In this section, we shortly explain the general structure of our best-performing deep learning architectures.

Text & Stylometric Features - CNN We extend on the Deep Learning ideas mentioned in the paper from last years winner (Zlatkove et al. ,section 3.4 [11]).

We tokenize the document and zero-pad it to a length of 7000 tokens. Additionally, we use a sliding window approach and extract the features for each document as described in section 3.1. We use the tokens and features as the input for our neural network. For the token/text input, we use an embedding layer with embedding dimension 300, followed by three convolutional layers with 64 filters each and filter sizes 3, 7 and 11. Each convolutional layer is followed by a squeeze and excitation block and a global max-pool layer [28]. For the feature input, we use 3 convolutional layers with 64 filters each and filter sizes 2,4 and 6. Again followed by a squeeze and excitation block and a global max-pool layer. We concatenate all six max-pool layers and add a fully connected layer with 256 hidden nodes.

Time Distributed CNN - LSTM We tokenize the document, zero-pad it to a length of 7000 token and split it into segments with 200 tokens each. A time distributed block is added. It consists of an embedding layer with 50 dimensions, three convolutional layers with 64 filters each and filter sizes 1, 2 and 3. Each convolutional layer is followed by a squeeze and excitation block and a global max-pool layer. The output of this time-distributed block is fed into an LSTM with 256 units, followed by a fully connected layer.

Time Distributed CNN - CNN This model is the same as the previous one. However, instead of using an LSTM at the end, we use three stacked blocks, each consisting of two convolutional layers with 64 filters and filter size three, followed by a pooling layer with pool size three. In the end, we add a fully connected layer.

Stylometric Features - LSTM We use a sliding window approach and extract all stylometric features from the chunks (see section 3.1). We use the resulting vectors as the input for our classifier, which consists of just one LSTM layer with 256 units, followed by a fully connected layer.

Autoencoder - Deep CNN We use the trained autoencoder (see section 4.1.2) to convert each sentence of a document to a 200-dimensional vector. These vectorised sentences are fed into a CNN with the following structure: eight blocks, with each block consisting of two convolutional layers with 128 filters of size three followed by a pooling layer with pool-size three. In the end, we add a fully connected layer.

We further tried using a deep character based model called VDCNN [29]. However, we were not able to achieve results surpassing a random baseline and will therefore not detail it any further.

4.3 Ordinal Regression

Ordinal regression is the task of assigning data points to a set of ordered categories. The task of predicting the number of authors falls into this category, since we want the error of the model to be smaller if we get closer to the correct number of authors. Ordinal regression shares properties of both classification and regression. As most Deep Learning models cannot deal with ordinal regression by default, we use a method proposed by J. Cheng [30]. With this method, instead of just one-hot encoding the labels, we also set the index of categories smaller than the true label to one (see example in table 4.1). To decode the class probabilities given by the network to integer labels, we use the index of the first probability which is smaller than 0.5. With this method, the network learns the ordinal relation between the data.

Integer Label	One-hot encoding	Ordinal encoding
1	[1 0 0 0]	[0 0 0 0]
2	[0 1 0 0]	[1 0 0 0]
3	[0 0 1 0]	[1 1 0 0]
4	[0 0 0 1]	[1 1 1 0]

Table 4.1: Example of Ordinal Label Encoding

4.4 Data augmentation

Data augmentation is the process of generating artificial training data and is used to prevent a model from overfitting when there is not enough real data available.

Since our train corpus is relatively small, with only 300 documents per class, our models tend to overfit. For this reason, we implement the following method to generate additional train samples:

1. Split all train documents into segments grouped by author
 $D_i = [A1_1, A2_1, A3_1, A2_2, A3_2] \rightarrow [A1_1], [A2_1, A2_2], [A3_1, A3_2]$
2. Select x random samples from the set of grouped segments
 $x = 3 \rightarrow [A1_1], [A4_1, A4_2, A4_3], [A5_1]$
3. Shuffle and combine the segments from the selected samples.
 $[A1_1], [A4_1, A4_2, A4_3], [A5_1] \rightarrow D_A = [A4_2, A5_1, A4_1, A1_1, A4_3]$

Results

5.1 Evaluation Metrics

In addition to accuracy and macro f1 score, we evaluate our classifiers using the Ordinal Classification Index (OCI) [31], as proposed by the PAN committee. The Ordinal Classification Index measures how much the result diverges from the ideal prediction and how inconsistent the classifier is concerning the relative order of the classes. It ranges from 0 to 1 with 0 being the best score.

Since no previous solutions to this task are available we compare our model to a random baseline which randomly guesses any of the 4 categories.

5.2 Clustering

We evaluated the different clustering methods on the PAN19 corpus. We performed the grid search for the best window size and feature subset selection on the train set. The resulting hyper-parameters were used to evaluate the model on the validation set.

Clustering Algorithm	Feature Set	Scores		
		Acc	F1	OCI
KMeans	All	0.34	0.32	0.79
	Forward selection	0.36	0.33	0.79
Hierarchical Agglomerative	All	0.30	0.29	0.82
	Forward selection	0.32	0.30	0.80
	tf-idf (words)	0.36	0.34	0.77
	tf-idf (char. 3,4 & 5-grams)	0.36	0.35	0.77
Random Baseline		0.25	0.25	0.86

Table 5.1: Clustering Results on PAN19 Validation Set (Task 2)

5.3 Deep Learning

We train all classifiers on the PAN19 train set using 5-fold cross-validation. The PAN19 validation set is only used in the end to calculate the scores.

Data Augmentation We could improve our results by a few percent on average by generating an additional 5000 augmented train samples using the PAN19 and PAN18 style change corpus.

Model	Scores		
	Acc	F1	OCI
Time Distributed CNN - LSTM	0.36	0.36	0.77
Time Distributed CNN - CNN	0.37	0.37	0.75
Autoencoder - CNN	0.30	0.27	0.79
CNN Text & Stylometric Features	0.37	0.34	0.73
Stylometric Features - LSTM	0.33	0.33	0.76
Random Baseline	0.25	0.25	0.86

Table 5.2: Deep Learning Results on PAN19 Validation Set (Task 2)

5.4 Ensembling

To further improve our solution we build an Ensemble Classifier using our best performing models. We use the predictions from all five k-folds to train a Logistic Regression Classifier and evaluate again on the PAN19 validation set.

In addition we tried a second ensemble method, which simply averages the predictions of all classifiers.

Model	Scores		
	Acc	F1	OCI
Ensemble - Logistic Regression (soft voting)	0.40	0.36	0.73
Ensemble - Averaging	0.36	0.34	0.70
Random Baseline	0.25	0.25	0.86

Models used by the Ensemble Classifier:

- ↳ tf-idf (char. 3,4 & 5-grams)
- ↳ tf-idf (words)
- ↳ Kmeans Forward selection
- ↳ CNN Text & Stylometric Features
- ↳ Time Distributed CNN - CNN

Table 5.3: Ensemble Results on PAN19 Validation Set (Task 2)

5.5 PAN19 Task 1

The submissions of the PAN19 style change detection task will be ranked by a combination of the accuracy for task 1 and the Ordinal Classification Index for task 2. Recall the definition of the first task in the PAN19 competition.

1. Given a document, is the document written by one or more authors, i.e. do style changes exist or not?

As we mentioned before this is the same task as in the PAN18 competition. We therefore additionally evaluate our best model on the PAN18 test set and compare it to the winning submission by Zlatkova et al.[11]

We use the same CNN model utilising the text and stylometric features as described in section 4.2. We train our model again using 5-fold cross-validation on the PAN19 train set and evaluate on the validation set. For the PAN18 corpus the test set used to rank the submissions is available. For this reason we train

using 5-fold cross-validation on the PAN18 train and validation set and evaluate on the PAN18 test set.

Model	Corpus	Acc
CNN Text & Stylometric Features	PAN19 (Validation set)	0.91
CNN Text & Stylometric Features	PAN18 (Test set)	0.88
Zlatkova et al. [11]	PAN18 (Test set)	0.89
Random Baseline		0.50

Table 5.4: Deep Learning Results on PAN19 (Task 1) & PAN18

5.6 Additional Results

We additionally trained one model on the sub tasks to decide between 1 and 2, 2 and 3, 3 and 4, 4 and 5 authors)

Model	Task	Acc
CNN Text & Stylometric Features	1 or 2 authors?	0.88
CNN Text & Stylometric Features	2 or 3 authors?	0.77
CNN Text & Stylometric Features	3 or 4 authors?	0.63
CNN Text & Stylometric Features	4 or 5 authors?	0.53

Table 5.5: Result on multiple sub tasks

Discussion & Future Work

6.1 Discussion

We were able to show that the clustering as well as the deep learning approaches are to some degree able to determine the number of authors and surpass the random baseline, with our best performing classifier being an ensemble of both approaches. Due to the lack of previous approaches to this task we are not able to compare the performance of our solution to existing ones. However, we could show that our deep learning model performs almost as well on the first task as the winning model from the 2018 PAN competition. On the harder task of determining the exact number of authors the performance decreases.

We make the assumption that our models are able to find some features defining the style of an author (high accuracy on task 1), however they are not able to find enough stylistic features to distinguish between multiple authors. To confirm this hypothesis we trained our model on the sub tasks explained in section 5.6. From the results we observed the accuracy dropping for each subsequent task, meaning the more authors there are in a text the harder it is for our models to extract and distinguish each individual style.

We further think that the size of the given corpus is too small to solve the second task with high accuracy. Our deep learning models tend to over-fit and fail to generalise. With our data augmentation technique (see section 4.4) we saw some improvements and our best results were achieved using the PAN18 corpus as an additional train source.

6.2 Future Work

One concept which we came across near the end of the thesis was Deep Embedded Clustering [32]. Due to the lack of remaining time, we were not able to implement it, but the fundamental idea behind it sounds quite promising since it would essentially combine both of our methods.

We saw promising results on the clustering part when using tf-idf representations, however we did not further explore tf-idf on the deep learning models. It would be interesting to see if the use of tf-idf representations instead of embeddings could also improve our deep learning models.

Bibliography

- [1] E. Stamatatos, “Intrinsic Plagiarism Detection Using Character n-gram Profiles,” 2009.
- [2] S. M. z. Eissen and B. Stein, “Intrinsic Plagiarism Detection,” in *ECIR*, 2006.
- [3] M. P. Kuznetsov, A. Motrenko, R. Kuznetsova, and V. V. Strijov, “Methods for Intrinsic Plagiarism Detection and Author Diarization,” in *CLEF*, 2016.
- [4] A. Daks and A. Clark, “Unsupervised Authorial Clustering Based on Syntactic Structure,” in *ACL*, 2016.
- [5] R. Layton, P. A. Watters, and R. Dazeley, “Automated unsupervised authorship analysis using evidence accumulation clustering,” *Natural Language Engineering*, vol. 19, pp. 95–120, 2013.
- [6] N. Akiva and M. Koppel, “Identifying Distinct Components of a Multi-author Document,” in *2012 European Intelligence and Security Informatics Conference*. Odense, Denmark: IEEE, Aug. 2012, pp. 205–209. [Online]. Available: <http://ieeexplore.ieee.org/document/6298832/>
- [7] —, “A generic unsupervised method for decomposing multi-author documents,” *Journal of the American Society for Information Science and Technology*, vol. 64, no. 11, pp. 2256–2264, 2013. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.22924>
- [8] E. Stamatatos, M. Tschuggnall, B. Verhoeven, W. Daelemans, G. Specht, B. Stein, and M. Potthast, “Clustering by Authorship Within and Across Documents,” in *CLEF*, 2016. [Online]. Available: <http://ceur-ws.org/Vol-1609/16090691.pdf>
- [9] M. Tschuggnall, E. Stamatatos, B. Verhoeven, W. Daelemans, G. Specht, B. Stein, and M. Potthast, “Overview of the Author Identification Task at PAN-2017: Style Breach Detection and Author Clustering,” in *CLEF*, 2017. [Online]. Available: http://ceur-ws.org/Vol-1866/invited_paper_3.pdf
- [10] M. Kestemont, M. Tschuggnall, E. Stamatatos, W. Daelemans, G. Specht, B. Stein, and M. Potthast, “Overview of the Author Identification Task at PAN-2018: Cross-domain Authorship Attribution and Style Change Detection,” in *CLEF*, 2018. [Online]. Available: http://ceur-ws.org/Vol-2125/invited_paper_2.pdf

- [11] D. Zlatkova, D. Kopev, K. Mitov, A. Atanasov, M. Hardalov, I. Koychev, and P. Nakov, “An Ensemble-Rich Multi-Aspect Approach Towards Robust Style Change Detection: Notebook for PAN at CLEF 2018,” in *CLEF*, 2018.
- [12] A. Sittar, H. R. Iqbal, and R. M. A. Nawab, “Author Diarization Using Cluster-Distance Approach,” in *CLEF*, 2016.
- [13] R. Zheng, J. Li, H. Chen, and Z. Huang, “A framework for authorship identification of online messages: Writing-style features and classification techniques,” *Journal of the American Society for Information Science and Technology*, vol. 57, no. 3, pp. 378–393, 2006. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.20316>
- [14] B. Stein, N. Lipka, and P. Prettenhofer, “Intrinsic plagiarism analysis,” *Language Resources and Evaluation*, vol. 45, no. 1, pp. 63–82, Mar. 2011. [Online]. Available: <http://link.springer.com/10.1007/s10579-010-9115-y>
- [15] D. I. Holmes, “The Evolution of Stylometry in Humanities Scholarship,” *Literary and Linguistic Computing*, vol. 13, no. 3, pp. 111–117, Sep. 1998. [Online]. Available: <https://academic.oup.com/dsh/article/13/3/111/933269>
- [16] P. Norvig, “Google Books Common Words.” [Online]. Available: <http://norvig.com/google-books-common-words.txt>
- [17] R. Flesch, “A new readability yardstick,” *Journal of Applied Psychology*, vol. 32, no. 3, pp. 221–233, 1948.
- [18] M. Dash and H. Liu, “Feature selection for classification,” *Intelligent Data Analysis*, vol. 1, no. 1, pp. 131–156, Jan. 1997. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1088467X97000085>
- [19] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise.” AAAI Press, 1996, pp. 226–231.
- [20] R. J. G. B. Campello, D. Moulavi, and J. Sander, “Density-Based Clustering Based on Hierarchical Density Estimates,” in *Advances in Knowledge Discovery and Data Mining*, ser. Lecture Notes in Computer Science, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Springer Berlin Heidelberg, 2013, pp. 160–172.
- [21] N. Bansal, A. Blum, and S. Chawla, “Correlation Clustering,” *Machine Learning*, vol. 56, no. 1, pp. 89–113, Jul. 2004. [Online]. Available: <https://doi.org/10.1023/B:MACH.0000033116.57574.95>
- [22] D. Arthur and S. Vassilvitskii, “K-means++: The Advantages of Careful Seeding,” in *Proceedings of the Eighteenth Annual ACM-SIAM*

- Symposium on Discrete Algorithms*, ser. SODA '07. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035, event-place: New Orleans, Louisiana. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1283383.1283494>
- [23] Y. Zhao, G. Karypis, and U. Fayyad, “Hierarchical Clustering Algorithms for Document Datasets,” *Data Mining and Knowledge Discovery*, vol. 10, no. 2, pp. 141–168, Mar. 2005. [Online]. Available: <https://doi.org/10.1007/s10618-005-0361-3>
- [24] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, Nov. 1987. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/0377042787901257>
- [25] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to Sequence Learning with Neural Networks,” in *Proc. NIPS*, Montreal, CA, 2014. [Online]. Available: <http://arxiv.org/abs/1409.3215>
- [26] Y. Kim, “Convolutional Neural Networks for Sentence Classification,” *arXiv:1408.5882 [cs]*, Aug. 2014, arXiv: 1408.5882. [Online]. Available: <http://arxiv.org/abs/1408.5882>
- [27] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: <http://dx.doi.org/10.1162/neco.1997.9.8.1735>
- [28] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, “Squeeze-and-Excitation Networks,” *arXiv:1709.01507 [cs]*, Sep. 2017, arXiv: 1709.01507. [Online]. Available: <http://arxiv.org/abs/1709.01507>
- [29] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, “Very Deep Convolutional Networks for Text Classification,” Jun. 2016. [Online]. Available: <https://arxiv.org/abs/1606.01781v2>
- [30] J. Cheng, “A neural network approach to ordinal regression,” *arXiv:0704.1028 [cs]*, Apr. 2007, arXiv: 0704.1028. [Online]. Available: <http://arxiv.org/abs/0704.1028>
- [31] J. S. Cardoso and R. Sousa, “Measuring the performance of ordinal classification,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 25, no. 08, pp. 1173–1195, Dec. 2011. [Online]. Available: <https://www.worldscientific.com/doi/abs/10.1142/S0218001411009093>
- [32] Y. Wang, Z. Shi, X. Guo, X. Liu, E. Zhu, and J. Yin, “Deep Embedding for Determining the Number of Clusters,” in *AAAI*, 2018.

.1 Stylometric features

Stylometric Feature	Count	F1
50 most common word 2-gram frequencies	50	0.313
50 most common character 3-gram frequencies	50	0.304
50 most common character 2-gram frequencies	50	0.294
Sichel S metric	1	0.283
Functionword frequency	150	0.275
50 most common word frequencies (excl. stop-words)	50	0.270
Linsear write formula	1	0.269
Flesch reading ease	1	0.261
Sentence length frequency distribution	7	0.257
Numeric character frequency	1	0.256
POS tag frequency	12	0.252
Average word length	1	0.248
Punctuation frequencies	8	0.243
Special character frequencies	21	0.242
20 most common POS tag trigram frequencies	20	0.240
Uppercase letter frequency	1	0.239
Average sentence length (chars)	1	0.238
Average word frequency class	1	0.236
Average sentence length (words)	1	0.234
Average syllables per word	1	0.232
Word length frequency distribution	10	0.226
Stopword ratio	1	0.218
Coleman Liau index	1	0.198
Smog index	1	0.192
Yule k metric	1	0.181

Table .1: List of stylometric features used, sorted by their f1 score on the PAN19 task