Bauhaus-Universität Weimar
Faculty of Media
Degree Program Medieninformatik

Authorship Verification and Obfuscation Using Distributional Features

Bachelor's Thesis

Janek Bevendorff

1. Referee: Prof. Dr. Benno Stein
2. Referee: PD Dr. Andreas Jakoby

Submission date: September 15, 2016

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, September 15, 2016

. . . . . .
Janek Bevendorff

Abstract

Authorship verification is used to determine whether two texts were written by the same author. Using obfuscation techniques, it is possible to degrade the classification performance of a given verification system. In this thesis, we develop an effective obfuscation method against a simple yet competitive reference classifier using distributional text features.

Contents

Chapter 1

Introduction

The *authorship verification* problem is an active research field with the goal to determine whether a given text was written by the same author as another sample text. Related to the verification problem is *authorship attribution* where we are given a set of example texts by certain known authors and a set of texts with unknown authorship and the task is to match each unknown text with its correct author. Verification and attribution are the two specific cases of the *authorship identification* problem.

Both verification and attribution provide useful tools for literary historians or even criminal forensics and prosecution. Determining authorship of an unknown text can uncover literary mysteries, solve criminal cases by examining and attributing written pieces of evidence to criminal offenders or de-anonymize postings published on the Internet.

It's important to carefully evaluate the quality and robustness of any practical verification or attribution system. Any such system has to be robust against misjudgment or deliberate impersonation. This is especially true if automatic authorship verification or attribution is used to support critical tasks such as evidence evaluation in a criminal case. But also the case of authorship determination of a historic publication can be critical because any further literary research may be based on a false assumption.

Many written works are published anonymously or under pseudonyms. Uncovering the real author behind such a text is not a trivial task, since the authorship identification system has to be robust against any deliberate change in writing style. An author who doesn't want to be identified by his known publications will most likely try to disguise his writing style. Deliberate text or style modification to avoid authorship detection (or even impersonate another author by the use of his particular writing style) is known as *authorship obfuscation*.

There are many good reasons for obfuscating writings. Publishing political

statements in oppressive regimes may be life-threatening, so obfuscation is a necessary tool for personal safety. In a less dramatic scenario, a malicious insurance company may try to identify customers in Internet health forums to increase their insurance fees in case they have any undisclosed diseases.

When talking about obfuscation, it is also useful to differentiate between manual and automatic obfuscation of text material. Manual obfuscation is usually targeted at a human audience while machine learning-assisted approaches might still be able to identify the original author. To obfuscate against automatic verification systems, it may be necessary to obfuscate different features than a manual (human) obfuscator would address. An ideal obfuscation is robust against both human and automatic authorship verification while still maintaining an *unsuspicious* appearance. Whether a text has been obfuscated or not may or may not be apparent depending on the obfuscation approach and quality. This is comparable to black mail letters written using newspaper letters glued onto paper to conceal the author's handwriting or wiping fingerprints. In both cases it is very obvious that an obfuscation was performed, but reconstructing who was originally behind it, isn't necessarily possible. Concealing the fact that obfuscation was performed at all can be an important quality factor, though, depending on the obfuscation requirements. Replacement of a text with a pre-defined constant or random character sequence is an effective and *safe* obfuscation approach, but additionally, preservation of content (*soundness*) and text flow (*sensibleness*) are also desired [24].

1.1 One-class Classification Problem

Generally, authorship verification is considered a harder problem to solve than attribution, since we are dealing with a one-class classification problem [17]. Unlike in the attribution case, we don't have a well-defined set of classes (the authors) but only a set of examples of the same author and possibly an additional set of examples of different authors. While the *same author* class can be exhaustive and fully defined (i.e., we know every text an author has ever written before), we don't have enough evidence to completely define the *different author* class. This also means that verification can help solving the attribution problem, but the reverse is not true.

Since verification is the harder and more general problem, this thesis will exclusively focus on authorship verification and obfuscation for interfering with successful verification. Attribution was mentioned for the sake of completeness.

1.2 Thesis Objective

The main objective of this thesis is to develop a reference authorship verification classifier using simple distributional text features and develop and evaluate an obfuscation technique to effectively degrade classification performance of this reference classifier. Performance degradation is achieved by systematically modifying the n-gram distribution of a text. It is shown that distributional features can be obfuscated easily by systematically modifying a text's n-gram distribution which in turn significantly lowers the accuracy of the whole authorship verification classifier.

Further testing is performed on how the obfuscated corpus affects the winning authorship verification system from the 2015 PAN authorship identification competition [28].

Additionally to the experiments on the PAN corpus, a new corpus is developed which does not inhere certain issues of the PAN corpus. Experiments are repeated on this corpus, confirming the aforementioned results.

Chapter 2

Related Work

Authorship attribution and verification are usually done using so-called stylometric features. These are certain (mostly linguistic) characteristics of a text suited for identifying its author. We briefly review known identification, obfuscation and de-obfuscation approaches in the following sections.

2.1 Attribution and Verification Approaches

Identifying authors is a problem scientists have been researching for centuries [4] and many different approaches have been developed over the years [26]. While early approaches consist of manually comparing wording and structure of paragraphs, Zheng et al. [31] propose a rich stylometric feature set with the writeprints technique and a general framework for automatic computer-driven authorship attribution. The writeprints feature set consists of *lexical* features (used words and characters, general vocabulary richness), *syntactic* features (mostly function words and POS n-grams), *structural* features (text organization and layout) and *content-specific* features (keywords and phrases). The framework consists of four consecutive steps they call *message collection*, *feature extraction*, *model generation* and *author identification*. The model training and identification steps are done using support vector machines (SVM), backpropagation neural networks and decision trees.

Abbasi and Chen [1] further develop the writeprints technique as a Karhunen-Loeve transforms technique. Using a sliding window, they build an author profile describing the variances of writeprint features. Features not used by a certain author are regarded as *disruptors* and the appearance of such disruptors in a text reduce the similarity to an author's profile. Additionally to the features suggested by Zheng et al., they also use idiosyncratic features (e.g. misspellings and grammatical mistakes). Abbasi and Chen show that the improved writeprints technique outperforms the previously suggested SVM

technique on texts from the Internet consisting of emails, forum posts, chat logs and ebay comments.

A slightly different approach to attribution is presented by Clark and Hannon [10] who identify an author by analyzing their use of synonyms for a word.

Zhao et al. [30] use POS tags and function words and measure the difference between their distributions using the Kullback-Leibler divergence for finding the author of a text. Using this technique, they achieve an attribution accuracy beyond 96 % on a selection of books from Project Gutenberg[1].

Since word-based approaches are often topic-dependent, Lipka and Stein [19] suggest the use of co-stems to identify an author. A co-stem is the part of a word that is removed by a stemming algorithm and can be used to identify an author without carrying any information about a text's topic.

For authorship verification, Koppel and Schler [17] come up with the unmasking approach. Instead of measuring only certain features that make two (sufficiently long) texts similar, they iteratively extract the features which allow for discrimination between the two texts. The extracted features are ordered by importance and the most important feature are removed in each iteration. While classification performance slowly degrades for texts by different authors, it degrades much faster for texts by the same author. This method relies on the assumption that texts written by one author only differ in very few superficial features.

Several other verification approaches, mostly for shorter texts, exist and were submitted as part of recent PAN[2] authorship identification competitions. The winning authorship verification method of the 2015 contest was submitted by Bagnall [3]. On the PAN 2015 corpus, their classification accuracy lies above 75 % using a multi-headed recurrent neural network. However, this result comes at a huge computational cost for training the model and a runtime between 13 and 24 hours on a typical PC or virtual machine.

In the 2014 competition, two evaluation corpora were provided, one with novels and one with essays. With 72 % accuracy on the novels corpus, the winning approach by Modaresi and Gross [22] compares several semantic and syntactic text features which are average sentence length, punctuation usage and spaces before and after commas. However, using spaces before and after commas as a discriminating feature appears to be a bit random and is likely an editorial artifact of the corpus.

Fréry et al. [11], who won the competition on the essays corpus with 71 % accuracy, use machine learning methods based on decision trees. The used features are a mean and an absolute counter value for four types of stylometric

[1] https://www.gutenberg.org/

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

characteristics: sentence lengths, variety of vocabulary, words / n-grams and punctuation marks.

Other participants such as Khonji and Iraqi [15] use similarity measures between texts based on the *general impostors* (GI) method developed by Koppel and Winter [18]. The general impostors method determines if two texts X and Y were written by the same author by comparing X to Y and a selected set of texts by different authors (impostors). If X is sufficiently more similar to Y than to any of the impostor documents, it is classified as being written by the same author as Y. Contrary to unmasking, which requires very long texts, this approach works well even on short texts.

As writeprints and synonym techniques are mainly designed for authorship attribution and are fairly complex, we do not directly employ them in our own authorship verification classifier yet. However, a distance measure between two distributions may also help determining if two texts are generally similar in style and therefore perform well in a verification scenario. Distributional features are also very easy to calculate on any set of features which can be represented as a probability distribution. We therefore use the Kullback-Leibler divergence proposed by Zhao et al. as a central feature.

We also use similar text preprocessing steps as the PAN 2015 winner Bagnall and use their classification results as a baseline to assess the competitiveness of our classifier. However, by using simple distributional features, we also achieve a much lower runtime of only a few seconds for training our model.

2.2 Authorship Obfuscation and Imitation

With the ability to attribute texts to certain authors or verify the similarity between two texts, the question arises how robust the existing approaches are against fraud and deception. Most obfuscation approaches are targeted at authorship identification and only few at verification.

Brennan and Greenstadt [6] and Brennan et al. [5] introduce the term *adversarial stylometry*. In a study they asked participants to hand in a pre-existing piece of writing of their own and then write two new texts. In one text they should try to obscure their authorship by modifying their writing style. In the other text they should imitate the style of another author. From these texts the Brennan-Greenstadt corpus was compiled. A publicly available extended version of the corpus also contains additional texts from an Amazon Mechanical Turk (AMT) task. All participants were untrained (non-professional) writers without any prior knowledge about obfuscation. The study shows that existing writeprints techniques were very susceptible to such primitive obfuscation and imitation and couldn't identify authors with certainty above random chance

anymore.

For comparison, texts were automatically translated several times using machine translation methods. However, other than manual obfuscation, automatic translation was found to be an ineffective obfuscation method compared to manual obfuscation.

Juola and Vescovi [12, 13] confirm the study results from Brennan and Greenstadt using JGAAP[3] (Java Graphical Authorship Attribution Program) for evaluating the effectiveness of obfuscations. JGAAP is a freely-available tool for automatic stylometric text analysis for authorship attribution. Although their results are in line with the prior experiments, they also find character-based obfuscation approaches to be more robust against automatic obfuscation than word-based approaches.

Khosmood and Levinson [16] propose an automatic obfuscation technique using synonym and phrase replacements. They also evaluate their approach using JGAAP and can show results comparable to manual obfuscation.

Further analyzing automatic machine translation as an obfuscation method, Caliskan and Greenstadt [7] show that translation is indeed ineffective and even allows for translator identification. They show that more translation iterations generally degrade the authorship identification performance, but certain features about an author remain. These are mostly top trigrams, words and top bigrams. Translated texts also contain characteristics about the used translation engine allowing for translator identification with over 90 % accuracy. However, the paper only describes experiments using the Google and Bing translator.

Another automatic obfuscation method is presented by McDonald et al. [20, 21] who developed Anonymouth, an automatic anonymization tool. The program requires as input one reference text, one text which is to be obfuscated and at least three texts by different authors. Using clustering methods, the most important authorship features are extracted and removed. Obfuscation with Anonymouth proves effective against authorship attribution using writeprints features.

Kacmarcik and Gamon [14] target the unmasking method by Koppel and Schler with a *deep obfuscation* technique. Unmasking appears to be very robust against superficial feature obfuscation because less obvious features remain untouched. Kacmarcik and Gamon use unmasking as a feedback function for finding further features which also need to be obfuscated. Using this deep obfuscation technique, they can break the unmasking verification method.

The general issue all shown obfuscation methods have in common is that they target certain identification or verification systems specifically. Since none

[3] http://evllabs.com/jgaap/w/

of them has been developed to obfuscate against verification using distributional features, we develop our own obfuscation technique which targets the distributional features of our authorship verification classifier.

2.3 Obfuscation Safety

For practical obfuscation, it is important to determine how safe an obfuscation method is. Using the extended Brennan-Greenstadt corpus, Afroz et al. [2] show that it is very easy to detect if a text was obfuscated. They build a multi-class SVM classifier with the three classes *regular*, *imitation* and *obfuscation*. Using writeprints features, they can detect with 85 % accuracy whether a text is an imitation and with 89.4 % whether a text is obfuscated. As most important feature for detecting deception, they identify function words. An actual identification of the original authors, however, cannot not be achieved.

Reconstructing the real authorship behind automatically obfuscated texts is done by Thi et al. [29]. The results of their research show that de-obfuscation is very easy if the obfuscation algorithm is deterministic and the initial training set used for obfuscation is known. Therefore obfuscation shows the same implications as cryptography where one cannot rely on the security given by a hidden algorithm. For de-obfuscation, Thi et al. determine the set of most important features (the features that are obfuscated using deep obfuscation) from the obfuscated document using the initial obfuscation training set and compare each of its feature points to all authors in the set. The author with the highest distance in every feature is the original author of the obfuscated document. This way, Thi et al. can break the unmasking obfuscation by Kacmarcik and Gamon. Anonymouth by McDonald et al. partially uses randomization, but can be broken by reconstructing the k-means clusters used during obfuscation.

Our obfuscation algorithm is mostly deterministic as obfuscation safety wasn't the priority for this thesis. However, the de-obfuscation approaches above should be kept in mind and randomization may be added in the future to harden the obfuscation against reversal.

Chapter 3

Reference Classifier Implementation

In order to be able to properly evaluate the effects of obfuscation techniques on a corpus, we need a reference authorship verification classifier. Several working classifier implementations exist and are available from past PAN competitions. However, the PAN classifiers which were available to us appear to be specifically tailored to the PAN setup and also lack proper documentation and customizability. It was therefore decided to implement a new classifier which will be used as reference throughout this thesis.

3.1 General Ideas

If two texts were written by the same author, they are expected to have more features in common than texts written by different authors. Features are specific characteristics of a text, such as writeprints features (cf. Section 2.1).

The classifier should be as simple and as little a black box as possible in order to be able to analyze the influence of text manipulations on individual features. We therefore need simple yet effective features which allow us to solve the authorship verification problem.

A very simple feature is the difference between n-gram distributions of two texts. If the two texts were written by the same author, we expect the n-gram distribution to be more similar than if they were written by different authors. For measuring the difference between distributions, several methods exist which will be discussed later.

The measurement of the difference between two n-gram distributions can then be supplemented by the addition of other stylometric features to improve the overall accuracy of the reference classifier.

Table 3.1: PAN14/15 corpus statistics. Average word counts are per text.

	Cases Training	Cases Test	Avg. Words Training	Avg. Words Test
PAN15	100	500	340	510
PAN14 Novels	100	200	3090	6000
PAN14 Essays	200	200	830	820

3.2 Used Corpus

For initial experiments, the English-language *PAN 2015 Authorship Verification* corpus [28] was used. The PAN 2015 Authorship Verification corpus is split into a training part of 100 cases and a test part of 500 cases (hereafter referred to as PAN15 training and test corpus). Each case consists of one or more reference texts by a *"known"* author and one comparison text by an *"unknown"* author. Both corpus parts are balanced, i.e. half the cases contain texts by the same author while the other half of the cases contains texts written by different authors.

Texts in the training corpus have an average length of 340 words after simple white-space tokenization (about 1980 characters, including white space). Test texts have on average 510 words (2690 characters).

By providing a balanced set of cases with texts by the same and different authors, the PAN corpus basically tries to approximate the one-class classification problem (described in Section 1.1) as a two-class problem which can be solved by simple binary Bayesian classifiers.

For controlling the generalizability of results, the preceding PAN14 *Novels* and *Essays* corpora were used, which are of the same basic structure.

Table 3.1 shows an overview of all three corpora. Word counts are per text while each *case* consists of two or more texts.

3.3 Text Preprocessing

Before building language models from the corpus texts, a normalization step was performed to strip features from the text which are either clearly editorial feature or at least don't obviously belong to an author's style. Some of these normalizations are based on the normalization decisions made by Bagnall [3] for their competition-winning classification technique.

- **White space reduction:** duplicate white space characters where collapsed into one space character. Line breaks were replaced with a single space character also.
- **Number mapping:** All numbers where mapped to a common number symbol (we chose the number `0`).
- **Character de-duplication:** Runs of identical characters where reduced to a maximum length of 5. This was especially done to reduce the length of overly long hyphen sequences and affects only very few texts.
- **Lowercase conversion:** all texts were converted to lowercase characters.
- **Vocabulary reduction:** Any infrequent characters, i.e. characters appearing less than 1 in 10,000, were removed reducing the character vocabulary to

  `<space>!"'-.,;:()?_0abcdefghijklmnopqrstuvwxyz`

3.4 Language Model Generation

The normalized texts were then used to build an individual language model of character trigrams for each text which will later be used to calculate distance features between pairs of texts.

Character trigrams contain enough information to describe basic word structures, word beginnings and endings, short function words, punctuation and sentence boundaries while still being relatively common, even in short texts.

3.5 Selection of Features

For measuring the difference between language models of two texts, a small set of common distributional difference measures were selected as features for the reference classifier.

3.5.1 Kullback-Leibler Divergence

The most common difference measure between two probability distributions is the *Kullback-Leibler divergence* (also known as *KL divergence* or simply *KLD*). The KLD measures the relative entropy of a distribution P with regard to another distribution Q. Therefore we get the encoding overhead, if we encode a sample of P with an optimal code for Q.

In our case of authorship verification, we can assume our first text to represent the *true* distribution P and our second text to be an approximation of P named Q. If both texts are identical, P and Q will also be identical and therefore the KLD will be 0. If the texts differ, also P and Q are expected to diverge and therefore the KLD increases. According to the hypothesis formulated in Section 3.1, the overall difference is smaller if both texts were written by the same author.

The Kullback-Leibler divergence for discrete probability distributions P and Q is defined as follows:

$$\mathrm{KLD}(P\|Q) = \sum_i P[i] \log \frac{P[i]}{Q[i]} \ . \tag{3.1}$$

For probability distributions, it is always positive semi-definite and unbounded towards infinity. Using the base 2 logarithm, the measurement unit of the KLD is *bits*.

Another important property to point out is the KLD's non-symmetry (i.e., $\mathrm{KLD}(P\|Q) \neq \mathrm{KLD}(Q\|P)$). Due to its asymmetric nature, the KLD is not a real metric. This isn't a big issue in our scenario, but we have to keep in mind that the order in which we compare texts matters. Symmetric variations of the KLD exist and are discussed in the coming sections.

One obvious problem that arises from the above definition is that the KLD is obviously not defined for any $Q[i] = 0$. To solve this problem, we can either assume that for any $Q[i] = 0$ we also have $P[i] = 0$ or make sure that $Q[i] = 0$ never occurs. The latter can be accomplished by only working on an intersection subset of n-grams from both document vectors $\mathbf{p}$ and $\mathbf{q}$ so that $\mathrm{freq}(q_i) = 0$ implies $\mathrm{freq}(p_i) = 0$ (and vice versa) or by smoothing one (or both) probability distributions, using, e.g., Dirichlet smoothing. For the reference classifier, the use of an n-gram subset was chosen because smoothing of both distributions appeared to be too disruptive on texts as short as in the PAN15 corpus. Therefore, P and Q represent probability distributions over a common subset of $\mathbf{p}$ and $\mathbf{q}$.

As mentioned in the previous chapter, using KLD as a difference measure between writing styles of authors is not a new idea. Zhao et al. [30] already described an application in authorship attribution (although using POS tags and function words instead of n-grams). Their good results let assume that a KLD measure can also perform well in an authorship verification scenario.

3.5.2 Skew Divergence

We can eliminate the need for a common n-gram subset by smoothing the Q distribution using P and a skew factor α, giving us a "skewed" variant of the

Kullback-Leibler divergence:

$$\mathrm{KLD}_\alpha(P\|Q) = \sum_i P[i] \log \frac{P[i]}{\alpha Q[i] + (1-\alpha)P[i]} \ . \tag{3.2}$$

The parameter α can be chosen arbitrarily in $(0, 1]$ (where $\alpha = 1$ results in the normal KLD). For lack of evidence which value works best, $\alpha = 0.5$ was chosen. A value $\alpha \ll 0.5$ should be avoided because very small values of α defeat the purpose of the KLD.

3.5.3 Jensen-Shannon Divergence

A symmetric variant of the Kullback-Leibler divergence is the Jensen-Shannon divergence (JSD). The idea behind the JSD is to use an artificial third distribution M which is a smoothed combination of P and Q:

$$M = \frac{1}{2}(P + Q) \ . \tag{3.3}$$

The Jensen-Shannon divergence is then defined as:

$$\mathrm{JSD}(P\|Q) = \frac{1}{2}\mathrm{KLD}(P\|M) + \frac{1}{2}\mathrm{KLD}(Q\|M) \ . \tag{3.4}$$

Since the JSD is based on the KLD, it is also always non-negative and identical distributions result in a JSD of 0. Together with the symmetry property, this makes the JSD an actual metric.

Another interesting and useful property of the JSD is that (using the base 2 logarithm for the KLD) it is bounded by $[0, 1]$ for probability distributions P and Q. This means we can better see the actual difference between two distributions even for different population sizes.

3.5.4 Hellinger Distance

The Hellinger distance (HD) is another similarity measure between probability distributions. Like the Jensen-Shannon divergence, the Hellinger distance is symmetric, non-negative and bounded by 1 making it also a (bounded) metric.

The general definition of the Hellinger distance for discrete probability distributions is:

$$\mathrm{HD}(P\|Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_i \left(\sqrt{P[i]} - \sqrt{Q[i]}\right)^2} \ . \tag{3.5}$$

The division by $\sqrt{2}$ at the front guarantees that $\mathrm{HD}(P\|Q) \leq 1$ for all probability distributions P and Q. The factor can be omitted for simplicity which changes the bounds to $[0, \sqrt{2}]$.

The above definition can also be written as the the Euclidean norm between the square roots of two probability vectors:

$$\mathrm{HD}(P\|Q) = \frac{1}{\sqrt{2}} \left\| \sqrt{P} - \sqrt{Q} \right\|_2 . \tag{3.6}$$

It also holds that

$$1 - (\mathrm{HD}(P\|Q))^2 = \sum_i \sqrt{P[i]Q[i]} , \tag{3.7}$$

where $\sum_i \sqrt{P[i]Q[i]}$ is the *Bhattacharyya coefficient* $\mathrm{BC}(P\|Q)$ so that the Hellinger distance can be defined as:

$$\mathrm{HD}(P\|Q) = \sqrt{1 - \mathrm{BC}(P\|Q)} . \tag{3.8}$$

3.5.5 Cosine Similarity

Besides measuring the difference between two n-gram distributions, also a geometric feature from information retrieval was added to the list of classifier features. By measuring the geometric distance between two term (i.e., n-gram) frequency vectors in a vector space model, we may also obtain some insight into the differences between two texts. It should be noted, though, that this feature needs to be treated with care as it mainly measures the topic of a text. Two texts may be about the same topic but written by different authors and vice versa.

The cosine similarity between two term frequency vectors $\mathbf{p}$ and $\mathbf{q}$ is calculated by

$$\cos\theta = \frac{\mathbf{p} \cdot \mathbf{q}}{\|\mathbf{p}\|\|\mathbf{q}\|} . \tag{3.9}$$

By normalizing $\mathbf{p}$ and $\mathbf{q}$ beforehand, we can omit the denominator.

Inverse Document Frequency

To make the TF vector space model robust against very frequent terms (mainly function words), an element-wise multiplication of $\mathbf{p}$ and $\mathbf{q}$ with a pre-calculated vector of *inverse document frequencies* (IDF) can be performed. It is debatable whether we actually want to compensate for very frequent terms in an authorship verification scenario. But considering that most function or stop words are

expected to appear in similar counts throughout all texts, we don't want them to dominate the sum.

Inverse document frequency is defined as the logarithmic ratio of the total number of documents N and the number of documents n_i in which a term i occurs.

$$\mathrm{IDF}_i = \log \frac{N}{n_i} \ . \tag{3.10}$$

For the effectiveness of a TF-IDF model it is vital to calculate IDF values from a source which is a decent representation of the document language. In general, using a large external corpus such as the Brown corpus (approx. 1 million words) or the ClueWeb collection (approx. 1 billion documents) is a good choice.

For initial experiments, IDF values were calculated on the corpus itself for simplicity reasons. This is something that should be handled carefully and kept in mind for further analysis. By calculating IDF values on a small corpus such as the PAN collections, we can easily bias the classification. In a real-world verification scenario, the classifier would probably also just get a single case at a time making it impossible to calculate meaningful IDF values from it. The PAN setup, however, allows for such an exploit of corpus-relative features and PAN participants were also able to use corpus-relative features during their evaluation phase.

3.5.6 Min-multiset Intersection Ratio

Because the KLD feature only works on the shared subset of n-grams while the other distance features implicitly build this intersection subset by multiplying with zero factors inside the sum (e.g., think of the dot product where a zero component in one vector leads to a zero summand), it is interesting to have an understanding of how much of the text mass we are actually working on.

To get a measure of the common (shared) text mass between both document vectors $\mathbf{p}$ and $\mathbf{q}$, a min-multiset

$$I = \{u_i^j \,|\, j = 2 \times \min(freq(p_i), freq(q_i)) \,\forall\, p_i, q_i, u_i \in \mathbf{p}, \mathbf{q}, \mathbf{p} \cap \mathbf{q}\} \tag{3.11}$$

and a union multiset

$$U = \{u_i^j \,|\, j = freq(p_i) + freq(q_i) \,\forall\, p_i, q_i, u_i \in \mathbf{p}, \mathbf{q}, \mathbf{p} \cap \mathbf{q}\} \tag{3.12}$$

were built. The min-multiset intersection ratio is the ratio between the sizes of both multisets

$$\mathcal{R}_{\min}(\mathbf{p}, \mathbf{q}) = \frac{|I|}{|U|} \ . \tag{3.13}$$

We expected this feature to be generally correlated to the cosine similarity feature because a higher number of common n-grams lead to both a higher min-multiset intersection ratio as well as a larger dot product. However, using this multiset, we get a measure that resembles the actual common text mass more closely. Having a measure for the common text mass is also useful as a meta feature for refining the classifier performance. Texts written by the same author are generally expected to have a larger intersection ratio than texts written by different authors allowing for a coarse pre-classification. Texts with very small ratios, on the other hand, may be excluded from classification due to too little evidence which optimizes precision of the classifier at the cost of recall.

3.5.7 Average Sentence Length Difference

As a last simple stylometry feature, the logarithmic difference between the average sentence length of two texts was added to the list. The logarithm was chosen because the feature can assume extremely large values which should be dampened. This is also a non-distributional feature and was mainly added to enrich the classifier with another feature that is independent of the n-gram distribution and therefore expected not to be correlated with any other feature.

For performing the sentence tokenization, the English-language NLTK punkt tokenizer was used.

3.5.8 Discarded Features

A number of additional features were evaluated, but eventually discarded because they proved either ineffective or redundant.

Bhattacharyya coefficient / distance

The Bhattacharyya coefficient

$$\mathrm{BC}(P\|Q) = \sum_i \sqrt{P[i]Q[i]} \tag{3.14}$$

is another difference measure for probability distributions. There is also a definition of a Bhattacharyya distance, which is defined as

$$\mathrm{BD}(P\|Q) = -\ln \mathrm{BC}(P\|Q)\ . \tag{3.15}$$

Since the classifier already uses the Hellinger distance and the Bhattacharyya coefficient is part of its definition, this feature was discarded. The Hellinger distance also proved more effective than the Bhattacharyya coefficient alone.

Cross Entropy

Cross entropy describes the number of bits required when encoding a sample of a distribution P with an optimal encoding for another distribution Q. It is defined as

$$\mathrm{H}(P\|Q) = -\sum_i P[i] \log Q[i] \ . \tag{3.16}$$

This feature is redundant, because the definition can be rewritten as

$$\mathrm{H}(P\|Q) = \mathrm{H}(P) + \mathrm{KLD}(P\|Q) \ , \tag{3.17}$$

where $\mathrm{H}(P)$ is the Shannon entropy of P and independent of Q.

Euclidean Distance

The Euclidean distance

$$\mathrm{d} = \sqrt{\mathbf{p}^2 + \mathbf{q}^2} \tag{3.18}$$

between two term vectors is another geometric feature. However, with cosine similarity, the classifier already uses a similar feature.

Kendall's tau

The Kendall's tau distance measures the pairwise disagreements between two population samples. An application of Kendall's tau to n-gram frequency vectors proved ineffective in solving an authorship verification problem (with accuracy around 50 %) and with an (unoptimized) runtime in $\mathcal{O}(n^2)$ also computationally infeasible.

3.5.9 Feature Summary and Initial Evaluation

Summarizing the preceding sections, eight features were selected for the reference classifier:

- Kullback-Leibler divergence
- Skew divergence
- Jensen-Shannon divergence
- Hellinger distance
- Cosine similarity (TF)
- Cosine similarity (TF-IDF)

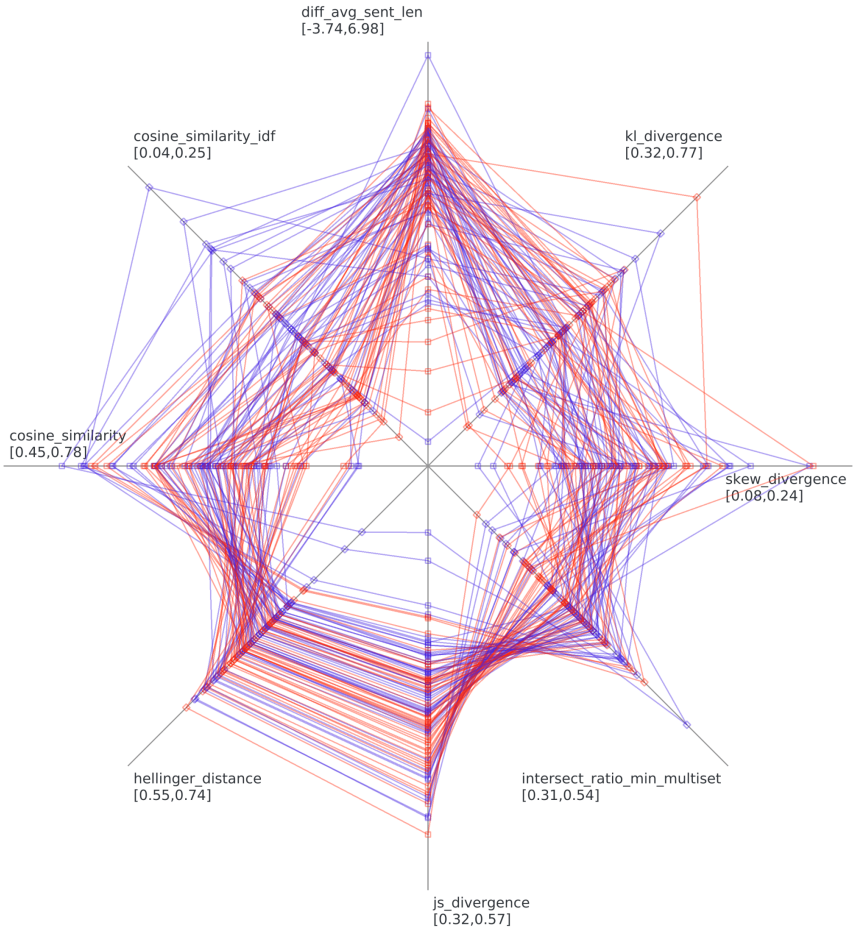


Figure 3.1: Star plot of classifier features for the PAN15 training corpus

- Min-multiset intersection ratio
- Average logarithmic sentence length difference

An initial star plot visualization of these features on the PAN15 training and test corpora (Figure 3.1 and Figure 3.2) give a general impression of how well the selected features discriminate between cases of the same (blue lines) and different authors (red lines). The plot axes were individually scaled to an interval $[0, 1]$ to give a better visualization result. Otherwise the sentence length difference feature would dominate too much making other features visually indistinguishable. Actual ranges were noted alongside axis labels.

It is clearly visible that the training corpus is rather small while the test

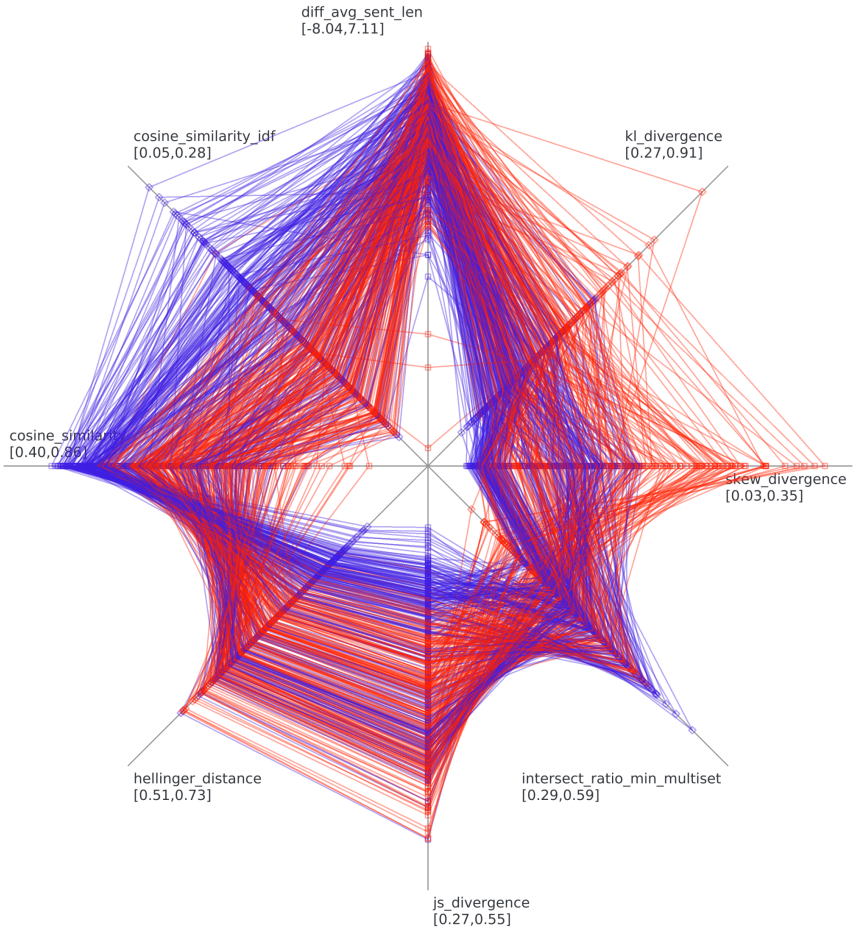


Figure 3.2: Star plot of classifier features for the PAN15 test corpus

corpus contains a lot more cases and immediately suggests certain discrimination patterns. Switching test and train corpora would make sense to improve the prediction model with more training samples, but this break comparability to the results of other PAN participants.

It is rather obvious in both corpora that the cosine similarity (TF-IDF) feature is very strong and already allows separation of a large portion of cases on its own. Because TF-IDF is generally designed to measure topic, this is a rather worrisome result and will be subject of discussion later.

It is also apparent that Jensen-Shannon divergence and Hellinger distance are almost completely correlated (with a correlation coefficient of 0.99) making one of these features redundant. Considering that both are actual distance metrics between probability distributions, this is not too surprising, but we expected a little more dissociation.

Jensen-Shannon divergence and min-multiset intersection ratio show a moderate to high negative correlation (training corpus: -0.96, test corpus: -0.47). The correlation between cosine similarity and intersection ratio cannot directly be seen in the plot, but is also about moderate (training corpus: 0.84, test corpus: 0.42).

The training corpus tends to show higher correlation between features, but this has to be treated with care because of the much smaller data set.

The best classifier on the PAN 15 corpus appears to be a C4.5 decision tree with a classification accuracy way above the accuracy of any other learning algorithm.

3.6 Classification Algorithms and Results

Using the above features, four machine learning algorithms from the WEKA[1] toolkit were trained: Naive Bayes Simple, SVM (libSVM), C4.5 decision tree (J48graft) and random forest. Using the trained model, classification accuracy on the test corpus varies a lot between different classifiers (Table 3.2).

In an attempt to improve classification accuracy, a linear normalization to an interval of $[0, 1]$ of each column in the feature matrices of training and test corpus can be performed. This pre-processing filter compensates for the large variance in feature ranges (the sentence length feature has an especially large range compared to other features, while the skew divergence range is rather small). However, this is also an exploitation of corpus-relative features, just like IDF calculation on the PAN corpus. Such an exploit is possible in the PAN setup, but not a valid strategy in real-world authorship verification settings.

[1] http://www.cs.waikato.ac.nz/ml/weka/

Table 3.2: Classification results on the PAN15 corpus.

	Accuracy	Precision	F-Measure	ROC AUC
Naive Bayes Simple	0.588	0.605	0.571	**0.760**
SVM	0.540	0.550	0.571	0.540
Decision Tree	**0.662**	**0.684**	**0.651**	0.639
Random Forest	0.568	0.580	0.551	0.627

Table 3.3: Classification results on the PAN15 corpus after exploiting the PAN setup with corpus-relative feature normalization.

	Accuracy	Precision	F-Measure	ROC AUC
Naive Bayes Simple	0.674	0.675	0.674	**0.771**
SVM	0.700	0.700	0.700	0.700
Decision Tree	**0.768**	**0.773**	**0.768**	0.748
Random Forest	0.660	0.661	0.660	0.717

Using the above-mentioned normalization trick, a greatly improved classification performance across all learning algorithms can be achieved. This way it is possible to get much better results in the competition setup, but the classification is heavily corpus-dependent which puts the generalizability of results from all participants into question.

3.6.1 PAN15 Corpus Results and Ranking

Comparing the results of our own reference classifier (using the decision tree) to the results of the PAN participants, we can see that the reference classifier is not only competitive, but even beats the accuracy of the PAN contest winner Bagnall. Overall, the classifier places second with a rather small margin towards first place.

The final score of every PAN participant is the product of their *Receiver Operator Characteristic AUC* and their *c@1* score. The *c@1* score was developed by Peñas and Rodrigo [23] and extends the classic accuracy measure by a possible *uncertainty* or *non-response* answer.

Participants had to generate an answer file with $[0, 1]$-normalized classification results where an answer of < 0.5 means "different authors" and > 0.5

Table 3.4: Comparison of the decision tree reference classifier with the top-5 PAN15 participants, ordered by final score.

	c@1	ROC AUC	Final Score
Bagnall	0.757	**0.811**	**0.614**
Bevendorff	***0.768***	*0.768*	*0.590*
Castro-Castro et al.	0.694	0.750	0.520
Gutierrez et al.	0.694	0.740	0.513
Kocher and Savoy	0.690	0.738	0.508
Halvani	0.601	0.762	0.458

means "same author". A value of exactly 0.5 is equivalent to "no answer". Using c@1 scoring, a non-answer induces a smaller score penalty than a wrong answer. It therefore allows for further score optimization, depending on the learning algorithm. Since our decision tree reference classifier produces binary classification results, no optimization was performed on our results.

A full list of participants and their results can be found in the PAN15 authorship identification overview paper by Stamatatos et al. [28].

3.6.2 PAN14 Corpora Results and Ranking

Results for the PAN15 corpus were verified by re-training and running the classifier on the PAN14 novels and essays corpora with comparable results that are shown in Table 3.5.

The best classifier on both PAN14 corpora is a random forest (see Table 3.5 and Table 3.6), with the decision tree performing only slightly worse. Since we are more interested in the general classificability of the corpus than in an actual real-world classifier at this moment, the random forest was chosen as classification algorithm for the reference classifier in the PAN14 setup. This random forest classifier was used for further comparison with PAN14 participants.

Note how the c@1 score is slightly higher than the raw accuracy due to the decision tree giving a few very uncertain answers with a probability of 0.5 instead of binary results.

Compared to the top-5 participants' performances in the PAN14 competition (Table 3.7 and Table 3.8), the classifier performs competitively in this setup as well. It places third on the novels corpus and first on the essays corpus.

A full list of participants and their results can be found in the overview

Table 3.5: Classification results on the PAN14 novels corpus.

	Accuracy	Precision	F-Measure	ROC AUC
Naive Bayes Simple	0.605	0.607	0.603	0.693
SVM	0.610	0.613	0.607	0.610
Decision Tree	0.625	0.628	0.623	0.634
Random Forest	**0.650**	**0.650**	**0.650**	**0.715**

Table 3.6: Classification results on the PAN14 essays corpus.

	Accuracy	Precision	F-Measure	ROC AUC
Naive Bayes Simple	0.630	0.630	0.630	0.665
SVM	0.565	0.565	0.565	0.565
Decision Tree	0.665	0.717	0.644	0.665
Random Forest	**0.720**	**0.721**	**0.720**	**0.761**

Table 3.7: Comparison of the random forest classifier with the top-5 PAN14 participants on the novels corpus, ordered by final score.

	c@1	ROC AUC	Final Score
Modaresi and Gross	**0.715**	**0.711**	**0.508**
Zamani et al.	0.650	0.733	0.476
Bevendorff	*0.651*	*0.715*	*0.466*
Khonji and Iraqi	0.610	0.750	0.458
Mayor et al.	0.614	0.664	0.407
Castillo et al.	0.615	0.628	0.386

Table 3.8: Comparison of the random forest classifier with the top-5 PAN14 participants on the essays corpus, ordered by final score.

	c@1	ROC AUC	Final Score
Bevendorff	***0.722***	***0.761***	***0.550***
Frery et al.	0.710	0.723	0.513
Satyam et al.	0.657	0.699	0.459
Moreau et al.	0.600	0.620	0.372
Layton	0.610	0.595	0.363
Modaresi and Gross	0.580	0.603	0.350

paper for the PAN14 authorship identification task by Stamatatos et al. [27].

3.6.3 PAN13

With only 10 training and 30 test cases, the PAN13 corpus was found to be way too small for reliable classification results which is why it is omitted here.

Chapter 4

Obfuscator Implementation

Having a working and competitive reference classifier on each of the PAN corpora, the next step is to distort its features in order to decrease prediction performance. Let us recall that an ideal obfuscator would obfuscate in a way that is *safe* (i.e., prevents successful authorship verification), *sound* (i.e., preserves text semantics) and *sensible* (i.e., produces a legible text without any obvious mistakes).

For the sake of simplicity, we will exclusively focus on *safety* and try to largely preserve either of the other two properties by manipulating the text in only as few places as possible. So instead of rephrasing the whole text, only small portions will be changed. For this to work, it is important as a first step to find the text parts which are most important for a correct classification.

4.1 Feature Analysis

Looking at the decision tree trained on the PAN15 corpus, the three strongest features of the reference classifier are *cosine similarity (TF-IDF)*, *Kullback-Leiber divergence* and *Jensen-Shannon divergence / Hellinger distance*. Leaving aside TF-IDF as a primarily topic-based IR feature for now, this leads to the assumption that the (dis-)similarity between n-gram distributions is the central feature which we can rely on for classification.

Analyzing how we can efficiently and effectively modify the n-gram distribution to influence the Kullback-Leibler divergence appears to be a sane first step with the hypothesis that obfuscation of the KLD feature also affects other distributional features. If this hypothesis holds can be seen later when analyzing the obfuscation results.

The KLD is also a bit simpler to analyze than the Jensen-Shannon divergence, which is based on it, though.

Let us first recall the KLD's formula and properties:

$$\mathrm{KLD}(P\|Q) = \sum_i P[i] \log \frac{P[i]}{Q[i]} \ . \tag{4.1}$$

With P and Q being probability distributions, the KLD is always *non-negative* and *unbounded*. If $P = Q$, then $\mathrm{KLD}(P\|Q) = 0$.

Looking at the plots in Figure 3.1 and Figure 3.2, the observed range of the KLD feature is about $[0.2, 0.9]$. So although the KLD is theoretically unbounded, we are actually dealing with a very limited range of real values.

4.1.1 Maximizing KLD

Given two texts by the same author, we need to make them less similar. We do that by obfuscating one of them in a way that increases the KLD between the n-gram distributions of these two texts. This obfuscation should be done at minimal cost. The cost of an obfuscation is the number of text changes needed to decrease the classification performance significantly (but we leave it open what amount of performance decrease is deemed *significant*).

Let our reference text (t_1) be represented by the distribution P and the text that is to be obfuscated (t_2) be represented by Q. In this setup, we can only perform modifications in Q while P is immutable.

So assuming P is immutable, the KLD is a sum of logarithmic fractions which are multiplied by a constant weight (the $P[i]$). To increase this sum, we need to increase its summands. While it is possible to increase the value of arbitrary summands, such a naive approach is very inefficient and incurs a large obfuscation cost.

To minimize the obfuscation cost, we should only modify the summands with the highest impact on the sum. Figure 4.1 shows a plot of how summands grow depending on P and Q. We can see that a summand (and therefore the sum) is large when $P[i]$ is large and $Q[i]$ is small.

Also note how an individual summand can assume a negative value while the full sum will always be positive due to the constraint of P and Q being probability distributions. The plot was drawn on a full scale between 0 and 1 for both P and Q, although the actual probabilities in a real-world scenario will always be extremely small. However, the plot looks identical for smaller and more realistic scales.

Approaching the maximization problem from a more mathematical point of view, the partial derivative of a single KLD summand (using the base 2 logarithm) confirms the observation made from the plot. Let p and q denote

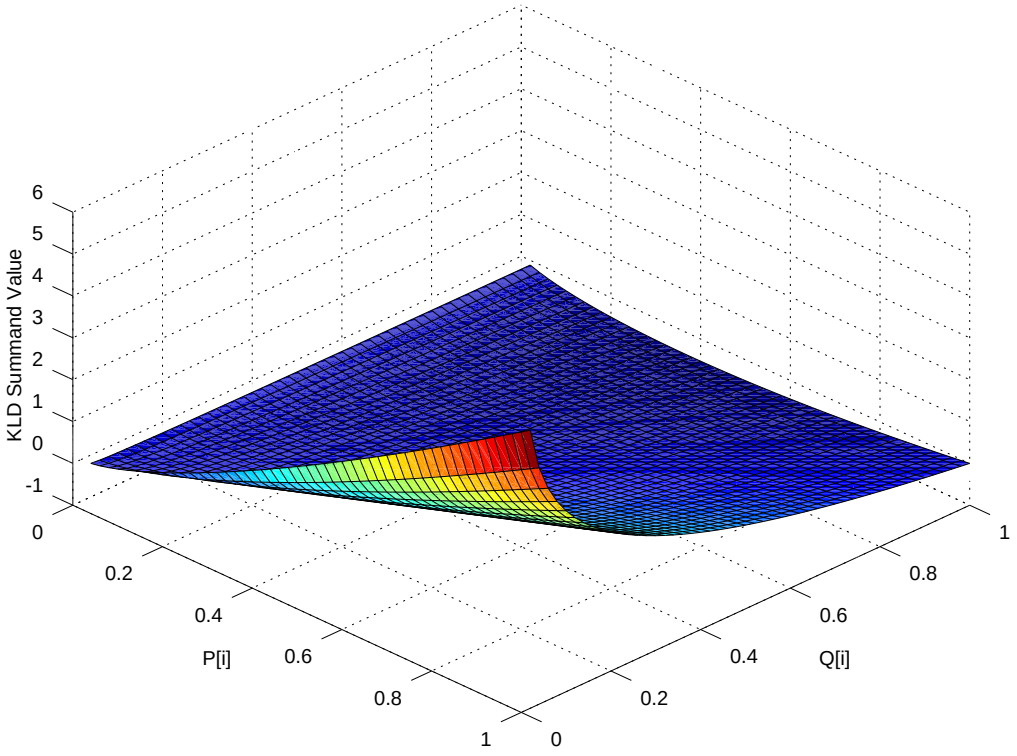


Figure 4.1: Plot of a single KLD summand.

the probabilities of $P[i]$ and $Q[i]$ for any i, then we have

$$\frac{\partial}{\partial q}\left(p \log_2 \frac{p}{q}\right) = -\frac{p}{q \ln 2} \ . \tag{4.2}$$

We can safely drop the $\ln 2$ constant from the denominator as well as the negative sign. A simple analysis of the limit values shows that the ratio p/q with constant p grows towards infinity as q approaches 0:

$$\lim_{q \to 0} \frac{p}{q} = \infty \ . \tag{4.3}$$

4.2 Implementation Strategy I: Reduction

The simplest way to implement an obfuscation method along the KLD gradient is to reduce the likelihood of certain n-grams by removing single occurrences from the text. This slightly reduces the text length, but if needed, additional

n-grams with very low KLD slope can be added instead. Such a simple obfuscation method removes and possibly inserts certain n-grams without taking any orthographic or grammatical rules into account, so we end up with a text with randomly-looking deletions and insertions. Legibility of the resulting text therefore depends heavily on the number of reductions we perform.

To determine which n-grams to reduce, the n-gram vector **q** is sorted by

$$\frac{\text{freq}(p_i)}{\text{freq}(q_i)}$$

in descending order, so n-grams with the highest slope come first. Obfuscation is performed by reducing the frequency count of the first n-gram in the sorted vector of t_2 with a frequency > 1 by 1. Skipping any n-grams with a frequency of only 1 is important. Otherwise we would reduce their count to 0 and take them out of the sum which effectively reduces the KLD instead of increasing it, because the KLD is only measured for the shared n-gram subset of both texts.

After this reduction step, the now highest-rated n-gram with a frequency above 1 is reduced. Reductions are repeated until a predefined number of iterations is reached and the algorithm terminates. The sorting step needs to be done only once because the relative order of unprocessed n-grams inside the sorted vector doesn't change after reductions.

A pseudo code implementation of this obfuscation-by-reduction algorithm looks as follows:

Algorithm 4.1: Reduction

```
1  input: vec p, vec q, int MAX_ITERATIONS
2  output: vec
3  begin
4    int counter := 0
5    vec qs := sorted_desc(q, key:=func(n): p[n] / q[n])
6    while counter < MAX_ITERATIONS:
7      foreach ngram in qs:
8        if qs[ngram] > 1:
9          q[ngram]--
10         break
11       end
12     end
13     counter++
14   end
15   return q
16 end
```

4.2.1 Obfuscation Results

The shown reduction algorithm works directly on the n-gram vector of text 2 without performing real text replacements. Working on the actual text introduces side effects because removal of n-grams always creates new n-grams consisting of parts of the text neighborhood. For a first obfuscation performance evaluation, we wanted to avoid these side effects. It will be shown later that their influence is marginal and can be neglected. Adjusting the frequencies directly inside the compiled n-gram vector without modifying the original text can be seen as an *obfuscation simulation*.

Figures 4.2, 4.3 and 4.4 show how the KLD for cases with the same author changes with an increasing number of obfuscation iterations. Cases with different authors were not modified at all and are shown for comparison.

The KLD histograms of both classes overlap completely at about 20 obfuscation iterations making any discrimination impossible and therefore completely breaking the KLD feature. When absolutely no separation between the two histograms is possible anymore, we say the feature is *fully obfuscated*. Individual texts in the PAN15 test corpus have an average length of 2690 characters which corresponds to about 897 non-overlapping trigrams. Since 20 iterations are needed, that means we only need to modify about 2.2 % of a text for a full KLD obfuscation in the simulated scenario.

After more than 20 iterations, histograms start diverging again with the *same-author* cases having a higher mean KLD than the *different-authors* cases. So by overobfuscating the text, it is actually possible to invert the KLD feature.

KLD by Min-multiset Intersection Ratio

Using the min-multiset intersection ratio (cf. Section 3.5.6) as a meta feature, cases were split around the intersection ratio mean μ by $\pm 1\sigma$ (σ being the standard deviation) for deeper analysis. The resulting histograms can be seen in Figures 4.5 and 4.6.

As already seen in Figure 3.2 (and assumed beforehand), the intersection ratio alone is enough to separate a certain amount of cases with high accuracy. Cases with a very small intersection ratio are almost exclusively cases with texts written by different authors. It is therefore possible to either classify all of them as belonging to the *different authors* class or not classifying those cases at all.

Cases with an intersection ratio larger than $\mu + \sigma$ are less clearly separable, but the majority of cases belongs to the *same author* class as expected.

The overall majority of cases (i.e., cases with an intersection ratio between $\mu - \sigma$ and $\mu + \sigma$) show a similar histogram overlap as the overall KLD distribution.

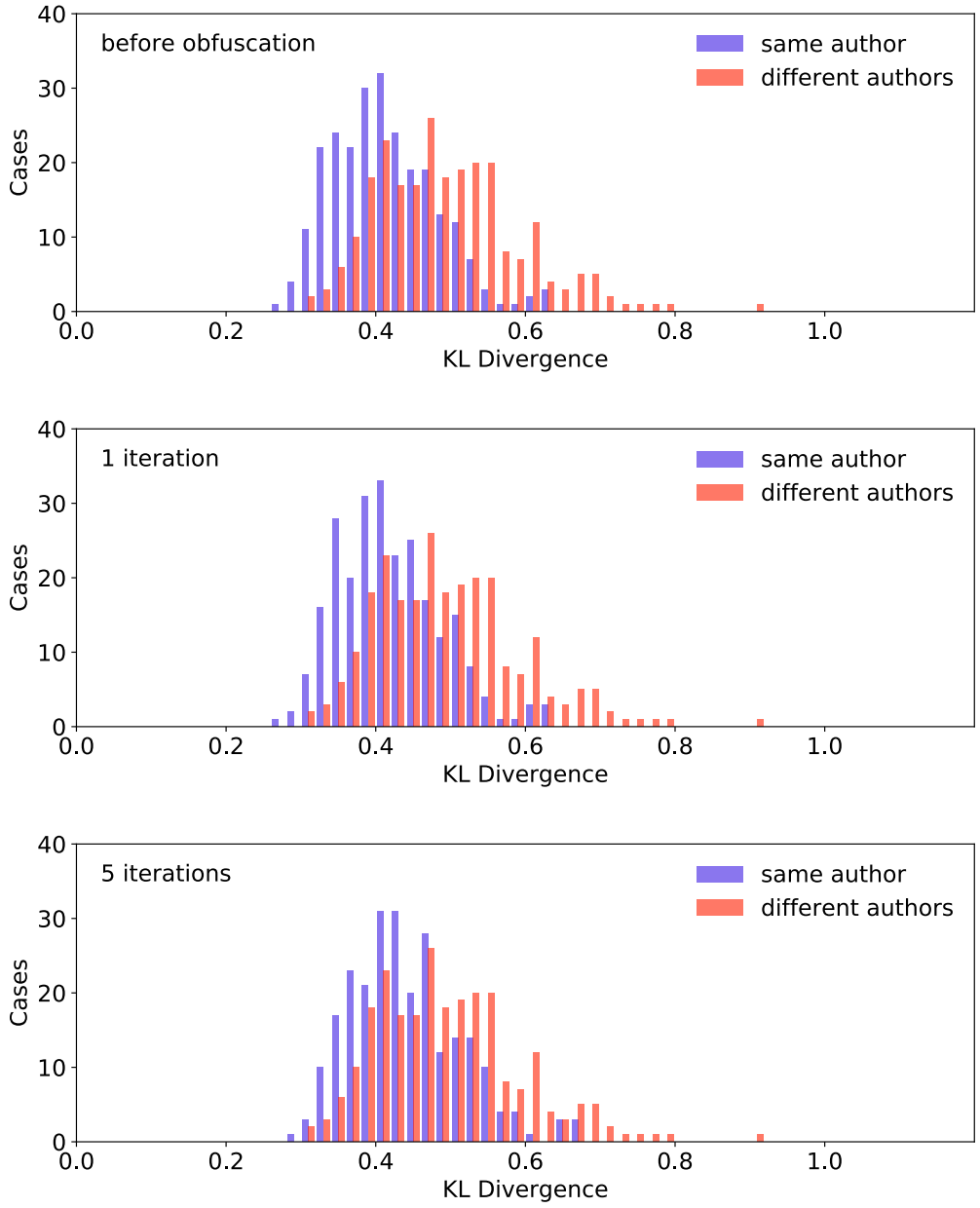


Figure 4.2: PAN15 test corpus KLD before obfuscation and after 1 and 5 reduction iterations.

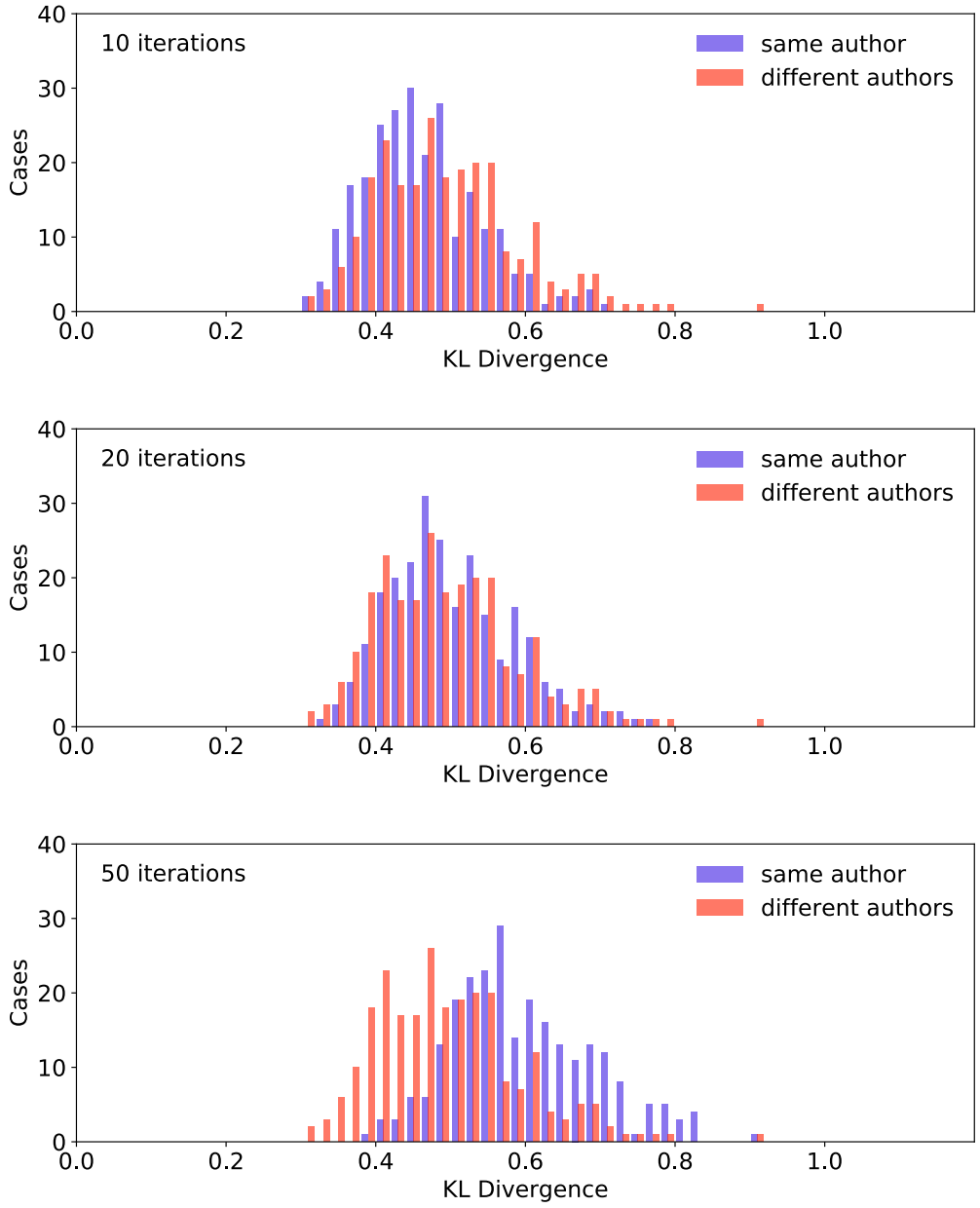


Figure 4.3: PAN15 test corpus KLD after 10, 20 and 50 reduction iterations. After 20 iterations a full histogram overlap is achieved. More iterations invert the feature.

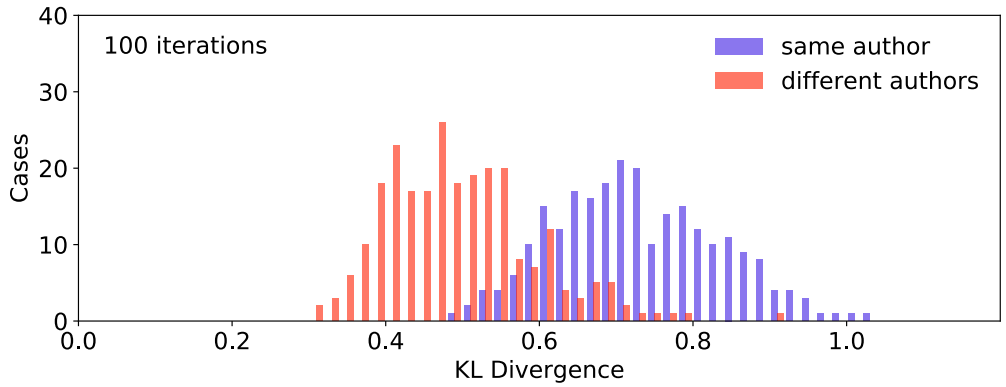


Figure 4.4: PAN15 test corpus KLD after 100 reduction iterations.

A KLD shift for *same author* cases can be observed in all three split windows around the min-multiset intersection ratio mean with increasing obfuscation iterations (Figure 4.5). That means that the obfuscation effect is the same for all cases regardless of their intersection ratio. It can also be seen that the amount of *same author* cases with an intersection ratio below $\mu - \sigma$ increases slightly with more iterations (see especially Figure 4.6), but with only 20 iterations this effect is very minor. The min-multiset intersection feature can therefore still be used to correctly classify a certain number of cases.

Obfuscation With Real Text Replacements

Any obfuscation so far was only performed on the raw n-gram vectors without modifying the actual text. This was done as an initial proof of concept to eliminate any side effects that are introduced by working on text strings. Removing an n-gram from an actual text not only reduces its frequency by 1, but also introduces $n - 1$ new n-grams consisting of the $2(n - 1)$ neighboring characters (example shown in Figure 4.7).

Figure 4.7: Example of trigram deletion in an actual text. By deleting `def`, new trigrams `bcg` and `cgh` are introduced.

How big of an influence such newly introduced n-grams have, is unpredictable. Simply deleting n-grams is likely to produce non-words, so the created n-grams

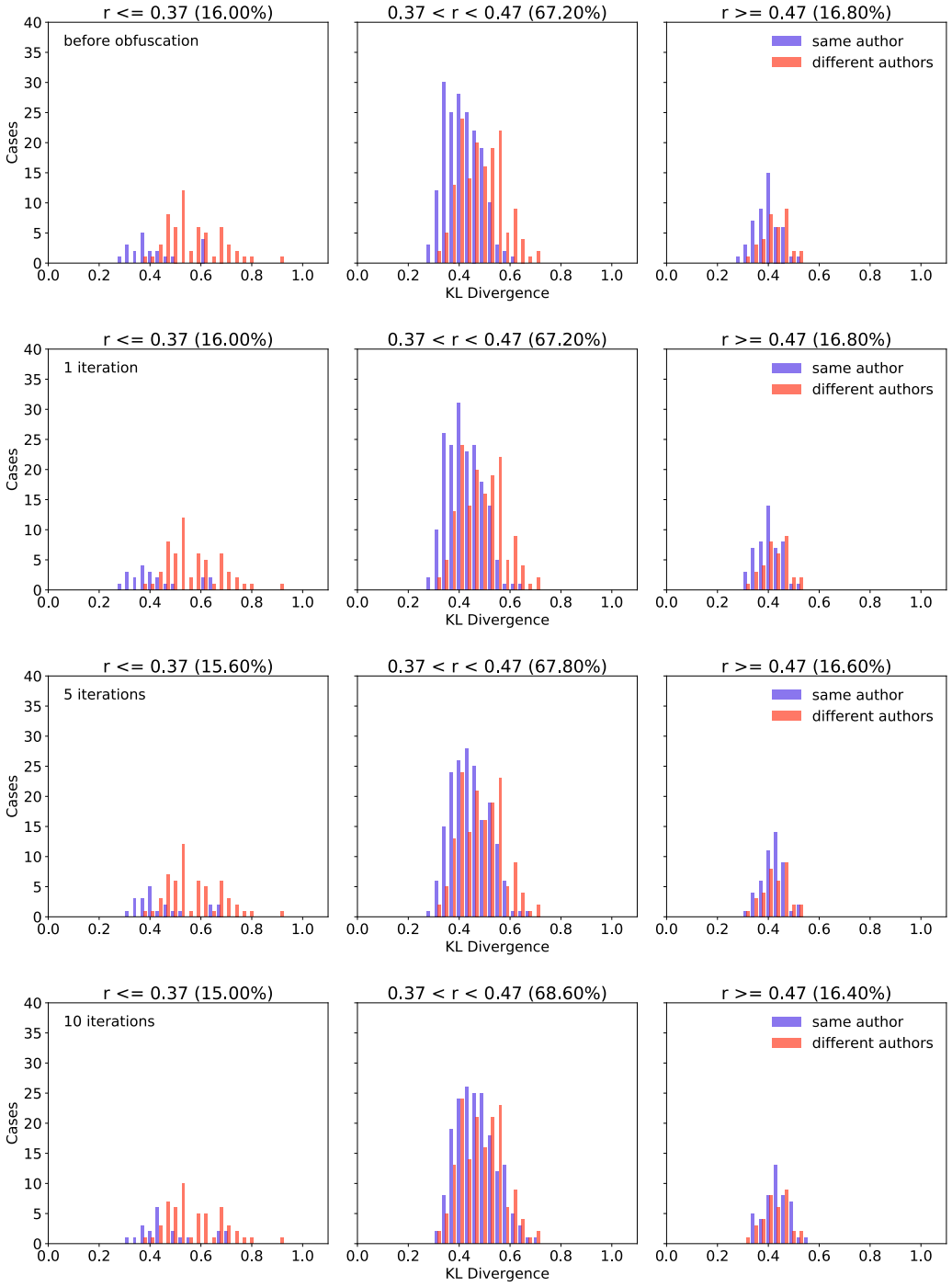


Figure 4.5: PAN15 test corpus KLD before obfuscation and after 1, 5 and 10 reduction iterations. The KLD is split into three windows based on the MMS intersection ratio mean at $\mu - \sigma$ and $\mu + \sigma$.

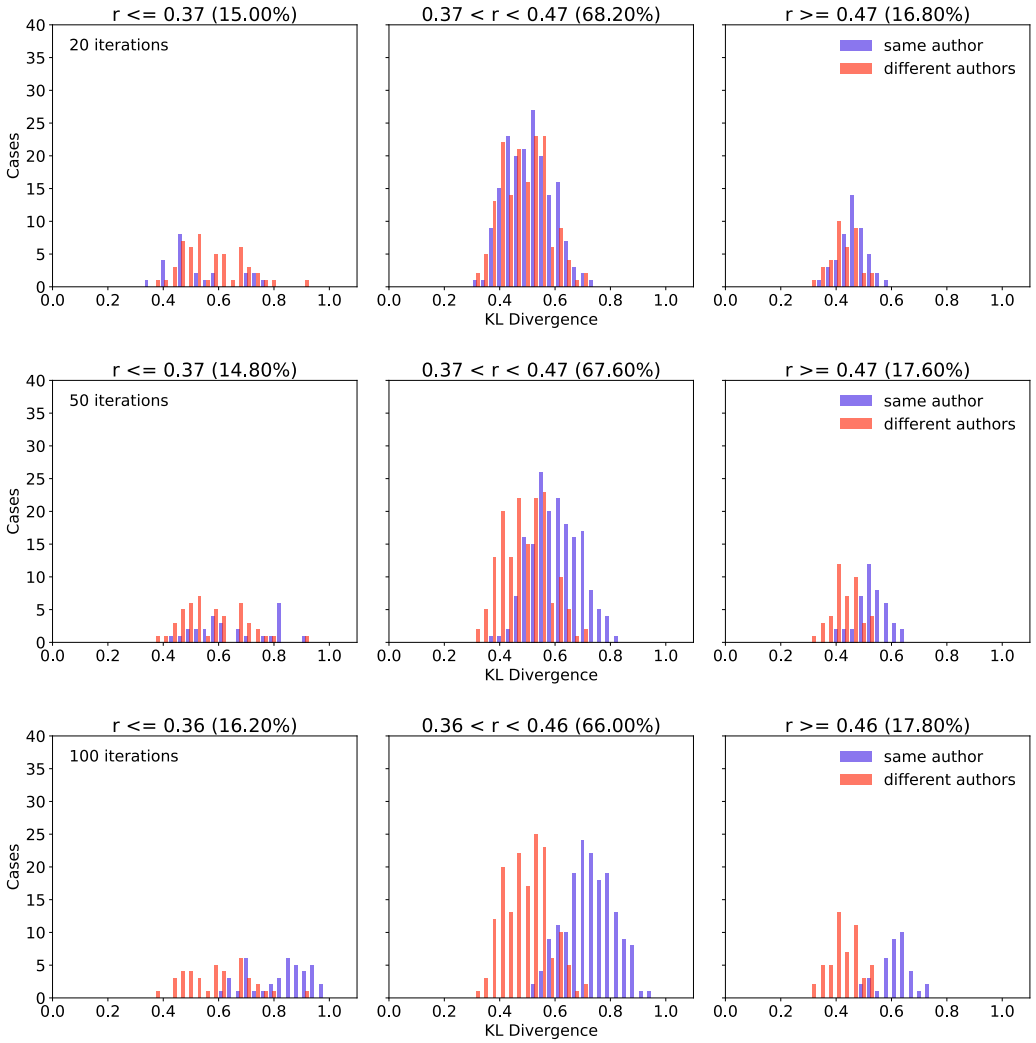


Figure 4.6: PAN15 test corpus KLD after 20, 50 and 100 reduction iterations. The KLD is split into three windows based on the MMS intersection ratio mean at $\mu - \sigma$ and $\mu + \sigma$. It can be seen that many reduction iterations increase the amount of *same author* cases in the left split window while a slight decrease can be seen in the right window.

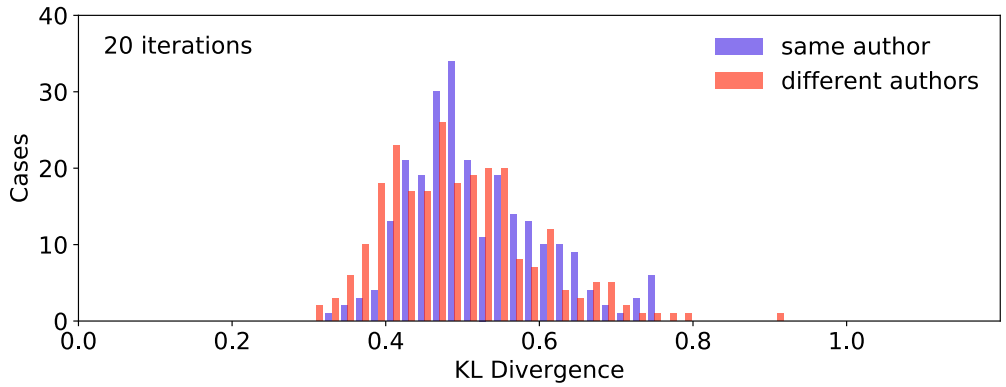


Figure 4.8: PAN15 test corpus KLD after 20 reduction iterations using real text replacements. The result is almost identical to simulated obfuscation.

are likely to be completely new to the text. If the new n-grams don't appear in the reference text, their influence is canceled out because they are not part of the common n-gram subset which the KLD is calculated on. If they do exist, their influence depends largely on their frequency ratio between the two texts. N-grams that appear very often in the reference text but are rare (or even newly introduced) in the obfuscated text have a high influence while n-grams that have a similar frequency in both texts have a low influence.

Both the aforementioned cases do not pose a problem as they actually support the obfuscation. This looks different when n-grams are rare in the reference text and already very common in the obfuscated text. In this case they have a negative impact on the obfuscation because their division produces a number smaller than one, resulting in a negative KLD summand.

Assuming n-grams introduced by text deletions are random, positive and negative effects should cancel each other out. This is unpredictable, but experiments on the actual corpus support this assumption. Figure 4.8 shows the KLD distribution of the PAN15 corpus after 20 obfuscation iterations. Compared to the histogram alignment after simulated obfuscation in Figure 4.3, the results look comparable. When looking at the KLD split by intersection ratio (Figure 4.9), a shift of the mean by 1 % to the left can be observed after 20 iterations. Because the performed obfuscation adds and removes n-grams, a certain change in mean and standard deviation is not surprising. As mentioned before, the left MMS split contains more *same author* cases after obfuscation. This is expected because by reducing the frequency of an n-gram, we also lower the MMS intersection ratio. However, assuming the side-effect n-grams exist in both texts, they should (at least) compensate this effect. The fact that a compensation cannot be seen, leads to the assumption that side-effect

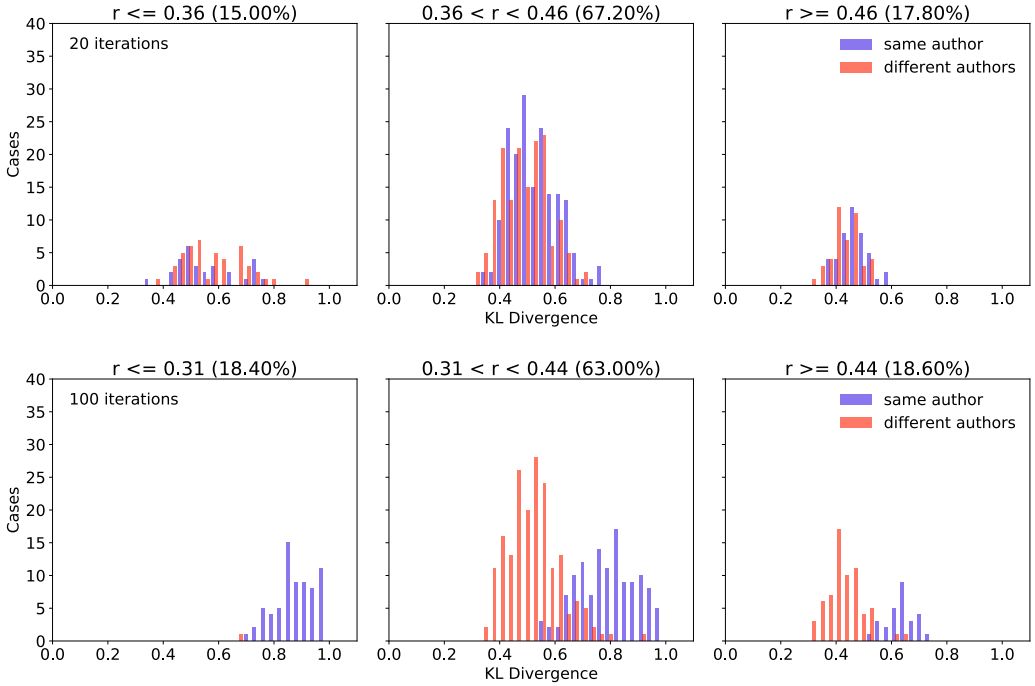


Figure 4.9: PAN15 test corpus KLD split into windows by MMS intersection ratio mean and standard deviation after 20 and 100 reduction iterations using real text replacements.

n-grams mostly do not exist in the reference text t_1. Figure 4.9 shows the KLD distribution by intersection ratio after an (extreme) number of 100 iterations. We can see that the left subset now consists of *same author* cases almost exclusively. This strongly supports the assumption that most newly created n-grams do not exist in the reference text.

4.3 Implementation Strategy II: Extension

An alternative approach is not to reduce the frequency of certain n-grams, but to introduce new n-grams in text t_2 which are already common in t_1 but have a zero frequency in t_2.

This strategy looks very promising because we do not only increase the value of an already existing summand by a certain amount but add a completely new one to the sum. Therefore we expect a generally higher KLD gain compared to the reduction strategy. On the other hand, this strategy supports the intersection ratio feature in the sense that it increases the ratio for *same author* cases.

Implementation of an extension strategy is quite similar to the implementation of a reduction strategy, but this time, we need to sort the frequency / probability vector **p** by decreasing frequency. A pseudo-code implementation looks as follows:

Algorithm 4.2: Extension

```
1  input: vec p, vec q, int MAX_ITERATIONS
2  output: vec
3  begin
4    int counter := 0
5    vec ps := sorted_desc(p, key:=func(n): p[n])
6    while counter < MAX_ITERATIONS:
7      foreach ngram in ps:
8        if q[ngram] = 0:
9          q[ngram] := 1
10         break
11       end
12     end
13     counter++
14   end
15   return q
16 end
```

4.3.1 Obfuscation Results

Figure 4.10 shows the KLD distribution after 10, 20 and 50 iterations using a simulated extension strategy. The histogram after 10 and 20 extension iterations appears to be about the same as after 10 and 20 reduction iterations. Is is very obvious, however, that the histogram after 50 extension iterations is almost exactly the same as after only 20 iterations, although a significant difference could be seen after obfuscation by reduction.

This result at first appears to be erroneous, but a look at the actually modified n-gram frequencies confirms that the extension approach is really that limited. At 50 iterations, the reduction strategy has an (absolute) mean p/q frequency ratio of 9.2 while the extension strategy only has a mean frequency ratio of 5.9. The reason is that the reference text t_1 contains only few very high-frequency n-grams which have zero frequencies in the obfuscated text. For both strategies, the mean p/q frequency ratio is expected to approach 1 with more iterations as the obfuscator runs out of n-grams with high ratios, but the extension strategy converges a lot faster. Also, for texts with a high intersection ratio (common for *same author* cases), the initial set of n-grams to

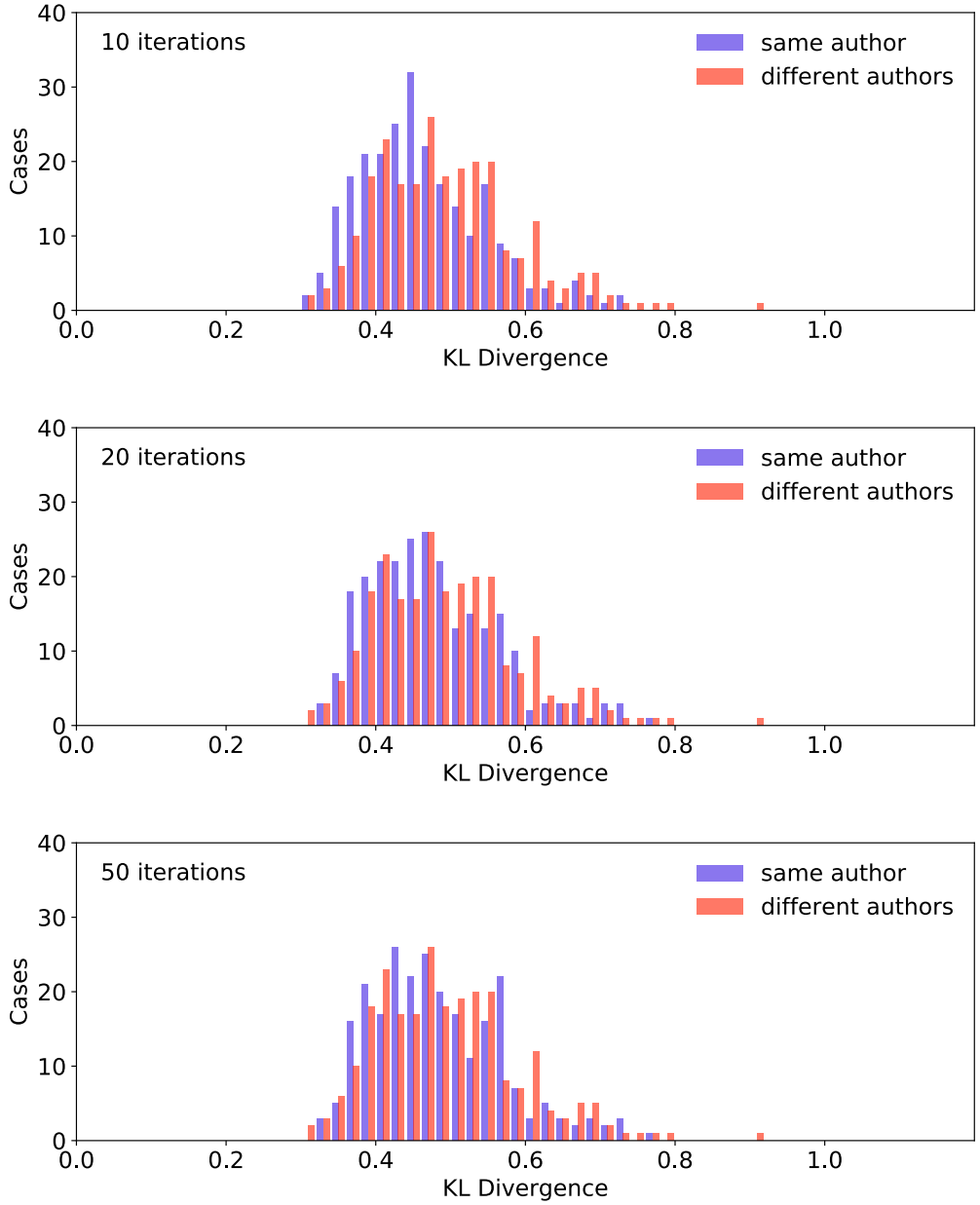


Figure 4.10: PAN15 test corpus KLD after 10, 20 and 50 simulated extension iterations. The results after 10 and 20 iterations are similar to the results after obfuscation by reduction, but no further shift can be observed for 50 extension iterations.

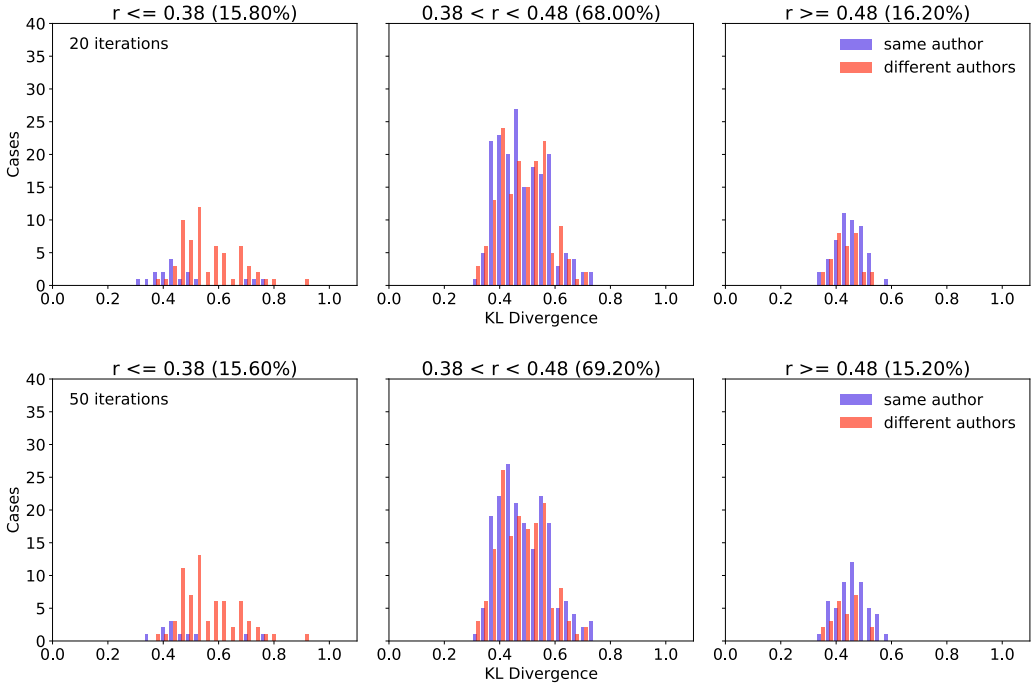


Figure 4.11: PAN15 test corpus KLD by MMS intersection ratio after 20 and 50 simulated extension iterations.

work with which have zero frequencies in t_2 is generally small. Furthermore, a reduction strategy can work on the same n-gram more than once as long as its frequency in the obfuscated text is above 1. An extension strategy can use an n-gram only once (although with initially greater effect).

With this limitation, the extension strategy is well-suited for full KLD obfuscation (i.e., total histogram overlap), but a feature inversion (i.e., shifting the mean of *same author* cases beyond the mean of *different authors* cases) using more iterations is not possible.

Obfuscation With Real Text Replacements

Side effects introduced by real text replacements are marginal as expected (cf. Figure 4.12). To keep the implementation simple, n-grams were just appended to the text string. Inserting them in random places would make the obfuscation less obvious. However, when inserting in random places, one should be careful not to destroy any high-effect n-grams (n-grams with steep KLD summand slope) as this would lower the KLD considerably. But looking at the limitations of the extension approach, using reductions is more effective anyway (contrary to our initial expectation).

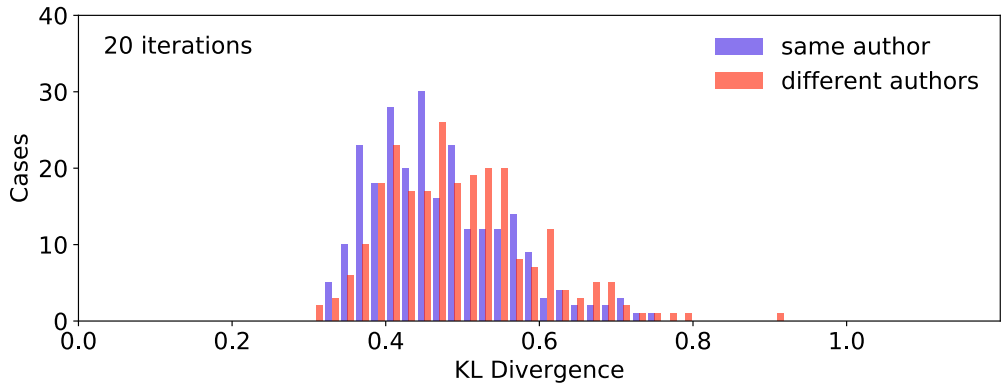


Figure 4.12: PAN15 test corpus KLD after 20 extension iterations using real text replacements.

4.4 Implementation Strategy III: Hybrid

To circumvent the limitations of an extension strategy but still use the benefits of this approach, a hybrid or dynamic strategy can be applied. The gradient for both strategies is calculated in every iteration and whichever strategy shows the higher slope is used for obfuscation. Using a hybrid strategy, we should get the best KLD obfuscation performance.

4.4.1 Obfuscation Results

Figure 4.13 shows the results of applying a hybrid strategy. Since it could be shown that simulated obfuscation and real text replacements produce almost identical results, only the simulated obfuscation was performed for simplicity reasons.

The performance seems to be slightly better than a pure reduction or extension strategy with only few iterations. Five iterations produce a result that is comparable to 10 iterations using only one strategy. With more iterations, the results are not much different than the results of reductions only.

Going beyond 20 iterations is possible using a hybrid approach (Figure 4.14). While the obfuscator prefers extensions for the first few iterations, reductions usually take over for further iterations.

4.5 Classifier Performance

The previous sections showed two basic and a hybrid KLD obfuscation approach. All three approaches allow at least full KLD obfuscation and with the reduction

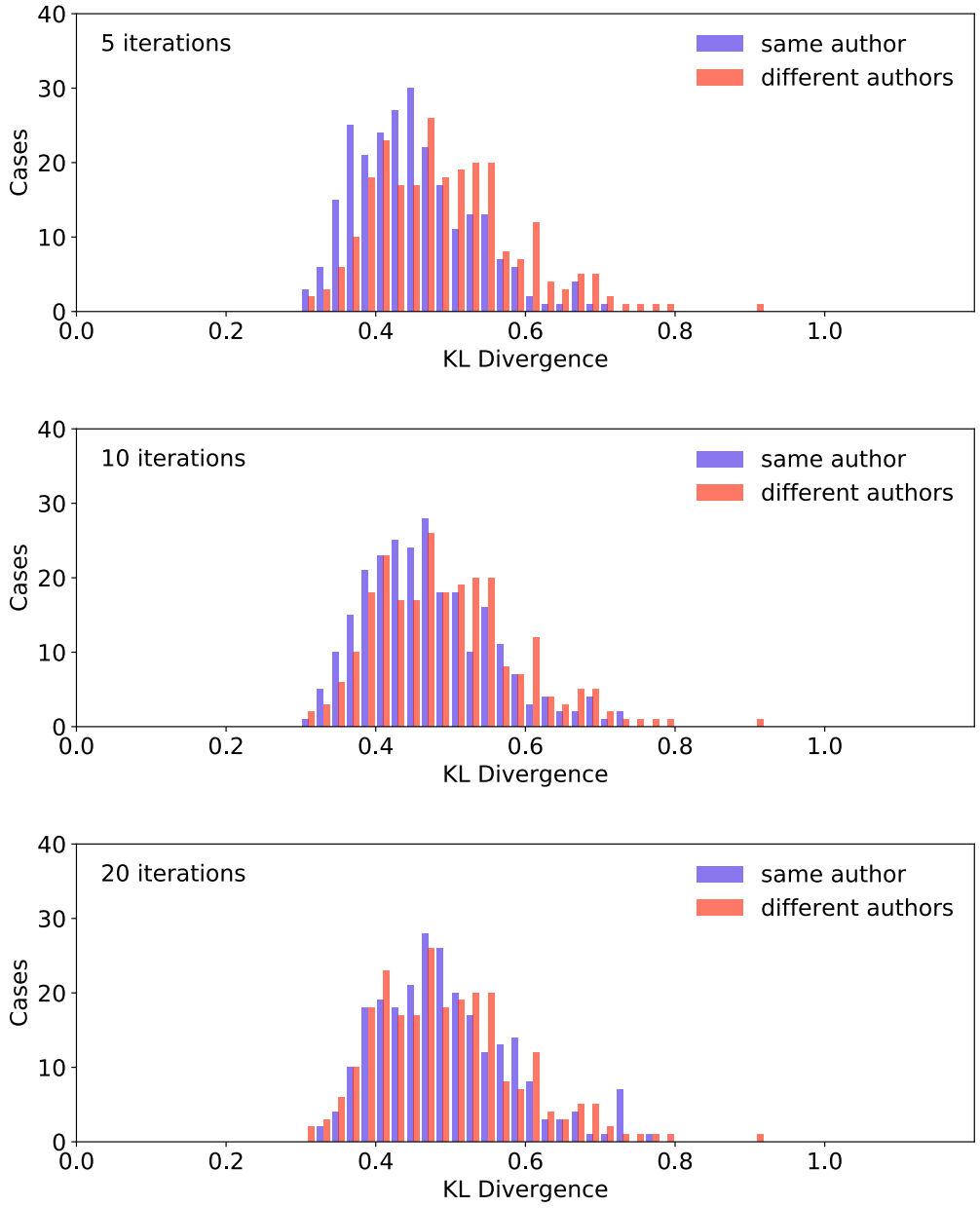


Figure 4.13: PAN15 test corpus KLD after 5, 10, 20 iterations using a hybrid strategy and simulated obfuscation.

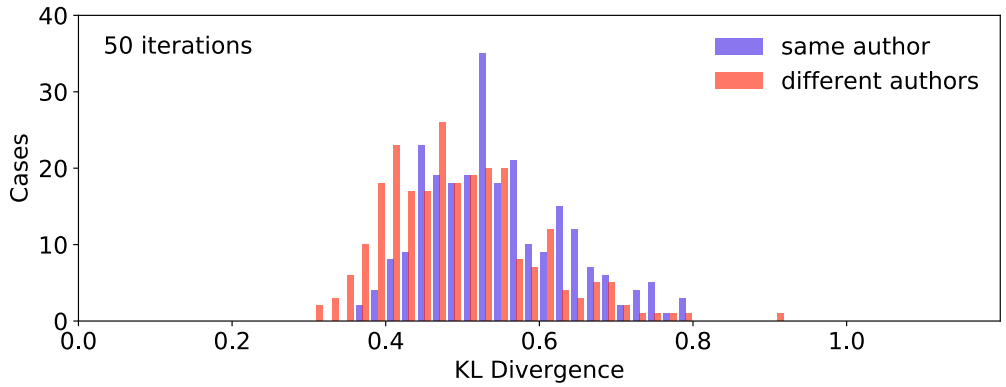


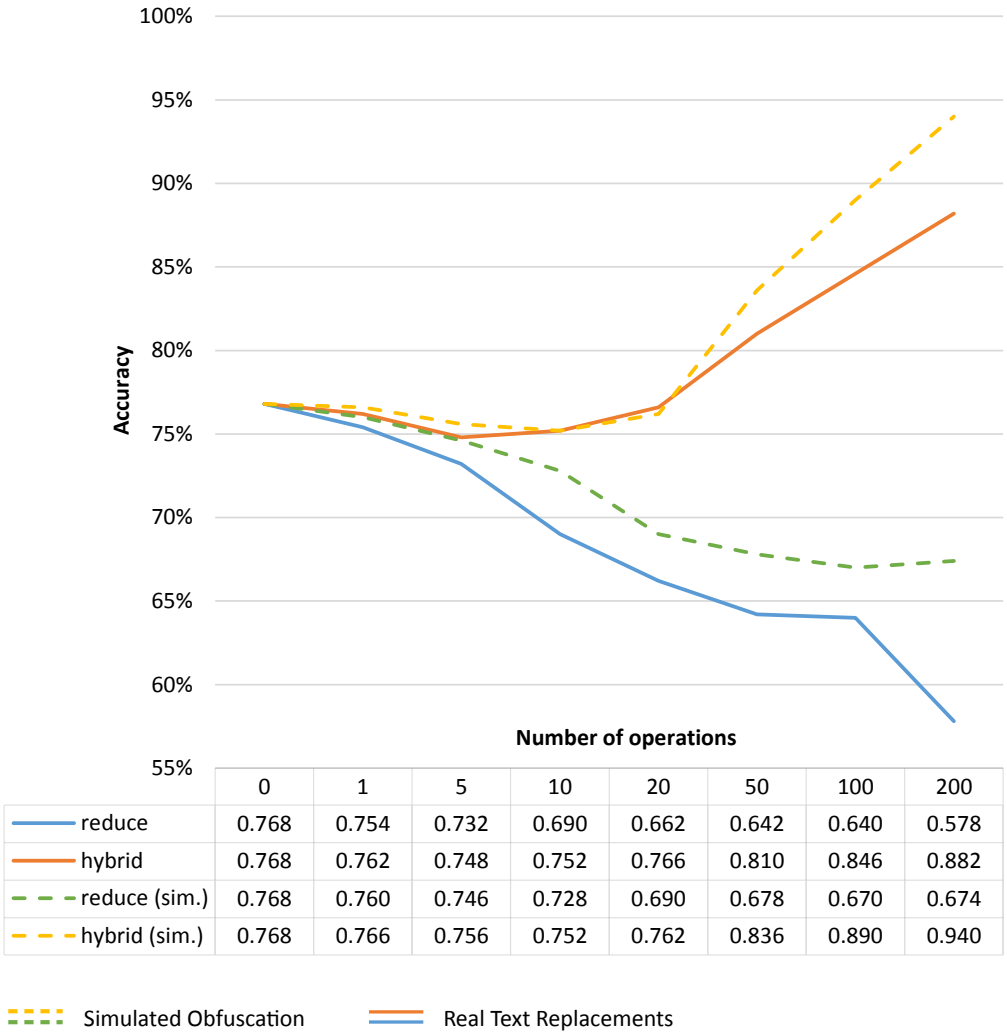
Figure 4.14: PAN15 test corpus KLD after 50 iterations using a hybrid strategy and simulated obfuscation.

strategy even a feature inversion (i.e., a higher KLD mean for *same author* cases than for *different authors* cases) can be achieved. What has not been evaluated yet is the overall performance degradation of the reference classifier. A significant performance decrease is expected, but so far only one feature was really analyzed. The classifier uses an overall set of eight different features. Modifying one of these features can have an impact on the other features, but they may also remain unaffected. If modifying one feature affects other features, it is also not clear what kind of effect such a modification has on the other features.

To evaluate the effect of KLD obfuscation on the overall reference classifier performance, we measured the classification accuracy after a certain number of obfuscation iterations. As classification algorithm, a decision tree was used. Because of the obvious limitations of the extension strategy, only the reduction and hybrid strategy were applied and measured. The results can be seen in Figure 4.15.

The starting point (0 iterations) is the baseline accuracy with no obfuscation. After 20 iterations, the KLD feature is completely broken. Any successful classification has to be the result of other features still allowing for proper case discrimination. Iterations beyond 20 invert the KLD feature. Although this theoretically allows us to distinguish cases again using the KLD, the classifier was trained to regard cases with a low KLD as belonging to the *same author* class. Overobfuscation can therefore be used to deliberately mislead the classifier. Basic rule-based classifiers such as the decision tree reference classifier are very susceptible to such misleading features.

With reductions only, the overall classifier performance can be decreased by 10.6 percentage points from 76.8 % to 66.2 %. It is striking, though, that using a



	0	1	5	10	20	50	100	200
reduce	0.768	0.754	0.732	0.690	0.662	0.642	0.640	0.578
hybrid	0.768	0.762	0.748	0.752	0.766	0.810	0.846	0.882
reduce (sim.)	0.768	0.760	0.746	0.728	0.690	0.678	0.670	0.674
hybrid (sim.)	0.768	0.766	0.756	0.752	0.762	0.836	0.890	0.940

Figure 4.15: Reference classifier performance curves on the PAN15 test corpus after obfuscation. After obfuscation by reduction, the accuracy drops as expected, but to our surprise rises after hybrid obfuscation. It should be noted that after 200 iterations, about 22 % of the text were modified which effectively reduces the shared n-gram subset of both texts by a lot.

hybrid approach, classification performance only decreases very little and then actually increases drastically after 20 iterations. Although the classification accuracy drops below 58 % after 200 reductions, it goes up to over 88 % with reductions and extensions. A simulated obfuscation even leads to an accuracy of 94 %. The exact values vary slightly within ±1 percentage points between obfuscation runs because of non-deterministic sorting of n-grams with equal frequencies.

Analysis of the discrimination quality of individual features explains this rather unexpected result. Using a Naive Bayes classifier with only a single feature at a time and 10-fold cross-validation, the TF-IDF feature can be identified as the main culprit.

Table 4.1: Cross-validation results on the PAN15 test corpus for single-feature Bayes classification before and after hybrid obfuscation.

	Baseline Accuracy	After 20 Iterations
Kullback-Leibler Divergence	**0.672**	0.524
Skew Divergence	0.596	**0.610**
Jensen-Shannon Divergence	**0.688**	0.674
Hellinger Distance	0.692	**0.694**
Cosine Similarity (TF)	**0.652**	0.644
Cosine Similarity (TF-IDF)	0.744	**0.828**
Avg. Sentence Length Diff.	**0.510**	0.504
MMS Intersection Ratio	**0.630**	0.626

Since training and test were both done on the test corpus for this analysis, the accuracy values in Table 4.1 act as a general upper bound for the possible classification accuracy using single features.

Other features seem to be mostly unaffected by the obfuscation which explains why the accuracy doesn't drop to about 50 % immediately.

The rise in accuracy for the TF-IDF feature after hybrid obfuscation can be explained in two ways. The first reason is that by widening the common n-gram subset, a general increase in the dot product between the two n-gram vectors is to be expected. This cannot be the only explanation, though, because then also the TF feature (without IDF) had to show a major performance increase.

The other explanation is that by extending the text, many rare n-grams are introduced that only appear in few documents (mainly in the reference text t_1 and after obfuscation also in t_2). Since IDF values are calculated directly on the test corpus, this leads to high weights for the new n-grams. Considering

that a new n-gram introduced to t_2 had a zero contribution to the sum before obfuscation, the result will be a significantly higher TF-IDF value. The feature range confirms this explanation. Before obfuscation, the TF-IDF feature had a range of $[0.05, 0.28]$. After obfuscation the range expanded to $[0.05, 0.32]$ with the majority of cases with a formerly low feature value seeing a major increase.

Table 4.2: Cross-validation results for single-feature Bayes classification after obfuscation by reduction.

	Baseline Accuracy	After 20 Iterations
Kullback-Leibler Divergence	**0.672**	0.556
Skew Divergence	**0.596**	0.586
Jensen-Shannon Divergence	**0.688**	0.648
Hellinger Distance	**0.692**	0.652
Cosine Similarity (TF)	**0.652**	0.612
Cosine Similarity (TF-IDF)	**0.744**	0.722
Avg. Sentence Length Diff.	0.510	**0.520**
MMS Intersection Ratio	**0.630**	0.518

Table 4.2 shows cross-validation results after obfuscation by reduction. The KLD obfuscation performance is slightly worse compared to the hybrid obfuscation results from Table 4.1, but the huge boost for the TF-IDF feature is gone. For most features, the accuracy even drops slightly compared to the hybrid strategy.

Considering the higher complexity of a hybrid strategy, its limitations and the classification performance results, the reduction strategy appears to be the more reasonable and stable obfuscation strategy.

4.6 Analysis of the Jensen-Shannon Divergence

The seeming invariance of the JSD and HD features to KLD obfuscation contradicts our initial hypothesis that KLD obfuscation also affects other distributional features. The fact that the JSD is only a symmetrized version of the KLD would lead to the immediate assumption that modifying n-grams for KLD obfuscation would have a similar effect on the JSD which makes these results even more surprising. We can show, though, that the classification performance invariance is a corpus-related issue.

Figure 4.16 shows a plot of a single JSD summand. Compared to the KLD plot in Figure 4.1, the characteristic looks indeed very similar for $P[i] \gg Q[i]$.

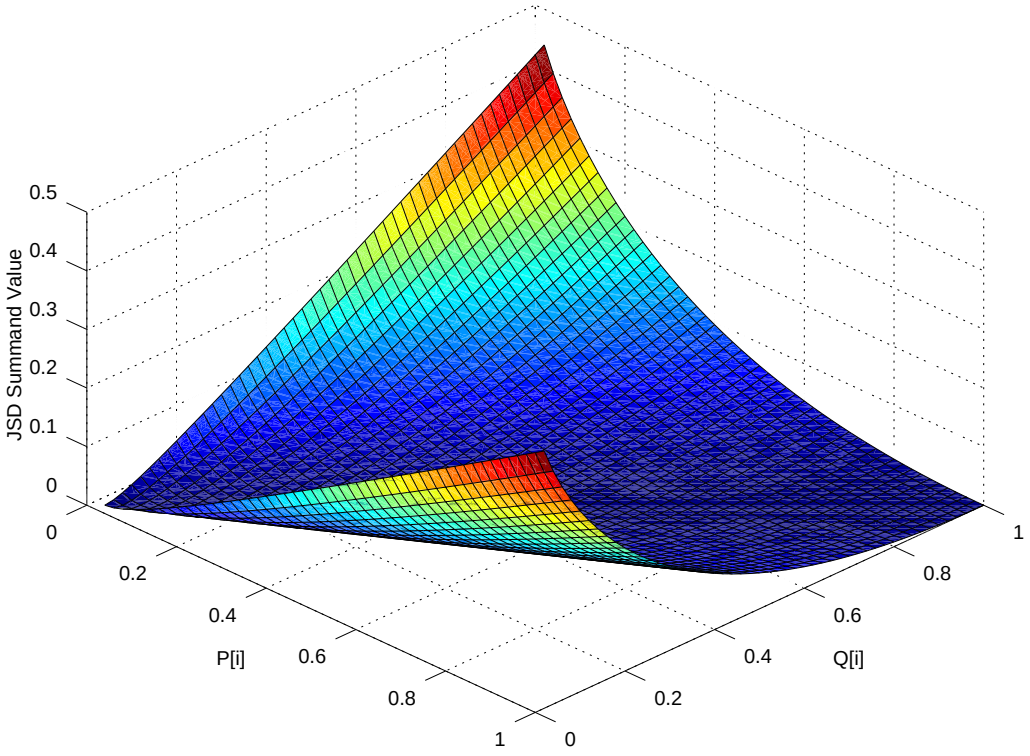


Figure 4.16: Plot of a single JSD summand.

The main difference is that a JSD summand cannot assume negative values and shows a symmetric slope for $Q[i] \gg P[i]$.

According to the plot, the already performed maximization of the ratio $P[i]/Q[i]$ should have a comparable result on the JSD, so there must be another reason why a classifier can still successfully use the JSD for separating cases.

In Figure 4.17 we see the corpus JSD distribution before obfuscation and after 20 reduction iterations. Although the histograms don't overlap perfectly, a pretty significant shift can be seen. The reason why we are still able to tell the two classes apart lies in the histogram shape of the *same author* cases. While the *different authors* histogram has a rather smooth bell shape, the *same author* histogram has two different peaks. The global maximum of the *same author* cases lies to the left of the global maximum of the *different authors* cases allowing a classifier to set a cut which achieves an accuracy above 60 %.

Remember though, that the almost invariant classification performance shown in Table 4.2 is an upper bound. Since a real classifier is trained on an unobfuscated corpus, the actual accuracy will be lower due to the JSD shift.

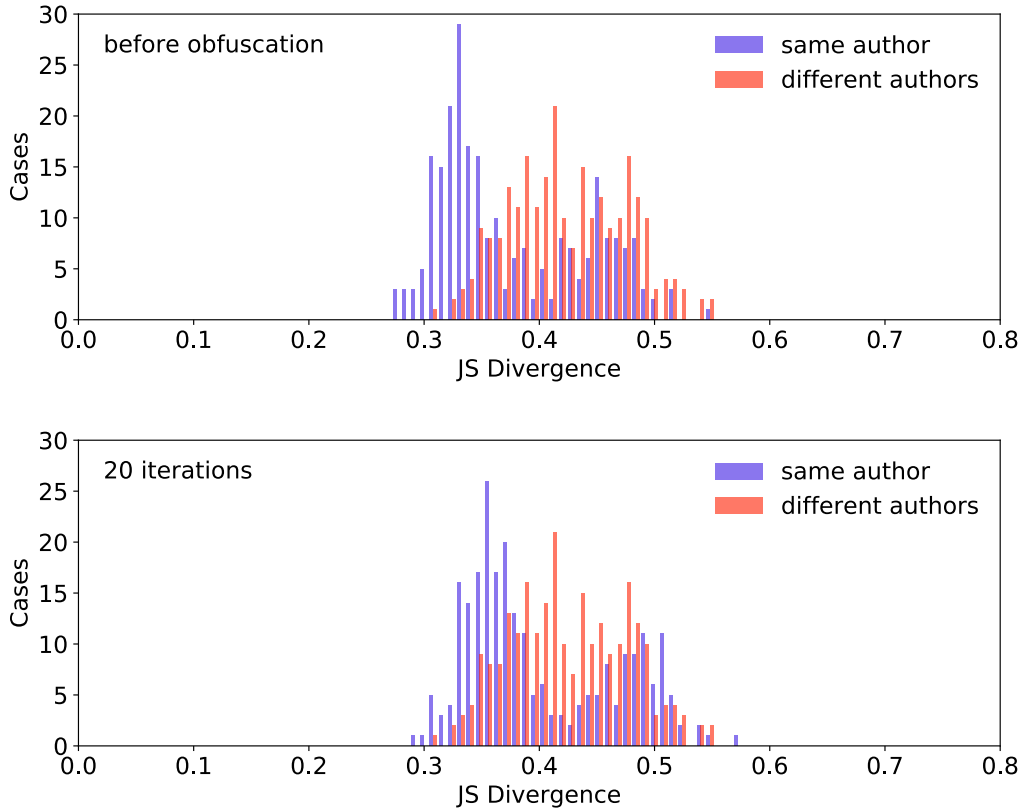


Figure 4.17: PAN15 test corpus JSD before obfuscation and after 20 reduction iterations. Although we can a shift of the histogram, a classifier can still exploit the significantly different JSD distribution characteristic of the *same author* cases.

There is no apparent reason why this histogram shape should be a systematic JSD behavior so must therefore be an artifact of the corpus. In Chapter 6 we will discuss the development of a different corpus without certain flaws of the PAN corpus which doesn't show this characteristic and behaves as expected.

4.7 Excursion: Authorship Boosting

While developing obfuscation techniques, the idea came that shifting the KLD into the opposite direction should also be possible by performing *inverse obfuscation*. By boosting authorship features, texts are made more similar instead of making them diverge. Applications for boosting authorship are mainly imitation and impersonation of other authors. Instead of hiding the fact that author A wrote a text, author A may want to make the text look like as if author B actually wrote it. Authorship boosting therefore has strong forensic implications because it allows forging of historic documents or falsification of evidence in a criminal case.

Inverse obfuscation is performed much the same way as a normal reduction obfuscation. The only difference is that instead of reducing the frequency of the n-gram with the highest gradient, its frequency is increased:

Algorithm 4.3: Boosting

```
1  input: vec p, vec q, int MAX_ITERATIONS
2  output: vec
3  begin
4    int counter := 0
5    while counter < MAX_ITERATIONS:
6      vec qs := sorted_desc(q, key:=func(n): p[n] / q[n])
7      ngram_t ngram := first(qs)
8      q[ngram]++
9      counter++
10   end
11   return q
12 end
```

For comparison, Figure 4.18 shows the KLD before obfuscation again and Figure 4.19 the boosting results after 20, 50 and 100 iterations.

Whether boosting the *same author* cases in a corpus leads to better classification results, depends mainly on the learning algorithm. The reference decision tree classifier trained on an unboosted corpus does not benefit from it, because the rules tend to be too specific and tailored towards the stronger

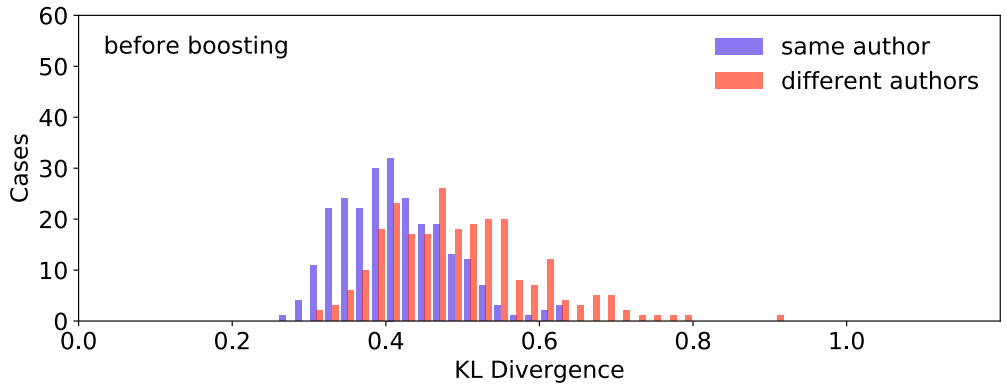


Figure 4.18: PAN15 test corpus KLD before boosting.

Table 4.3: Cross-validation results for single-feature Bayes classification.

	Baseline Accuracy	After 20 Iterations
Kullback-Leibler Divergence	0.672	**0.814**
Skew Divergence	0.596	**0.606**
Jensen-Shannon Divergence	0.688	**0.690**
Hellinger Distance	0.692	**0.706**
Cosine Similarity (TF)	0.652	**0.662**
Cosine Similarity (TF-IDF)	0.744	**0.754**
Avg. Sentence Length Diff.	0.510	0.510
MMS Intersection Ratio	0.630	0.630

TF-IDF feature. However, by using a cross-validated decision tree classifier trained on the boosted PAN15 test corpus, an accuracy increase from 76.8 % to 86 % could be achieved with only 20 iterations.

Table 4.3 shows the discrimination quality of individual features using a Naive Bayes classifier. After boosting, the KLD takes over as best discriminating feature. Furthermore, a general slight increase in other distributional features can be observed.

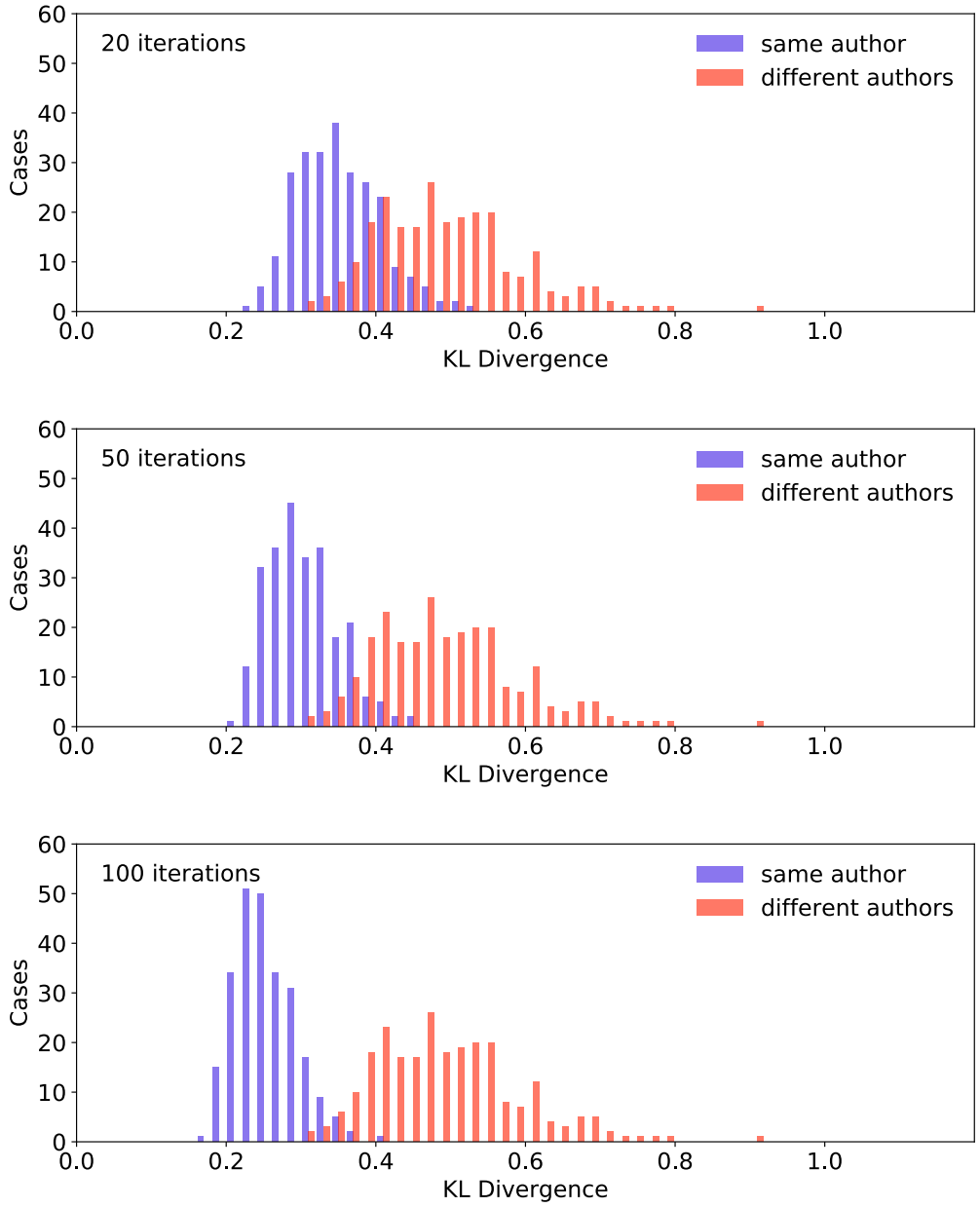


Figure 4.19: PAN15 test corpus KLD after 20, 50 and 100 simulated boosting iterations.

Table 4.4: Caravel results on the PAN15 corpus before obfuscation and after 20 and 50 reduction iterations. Additionally, the results after a corpus-wide text normalization as described in Section 3.3 are shown.

	c@1	ROC AUC	Final Score
Baseline Accuracy	0.757	0.811	0.614
20 Iterations	0.675	0.695	0.469
50 Iterations	0.640	0.669	0.429
Text Normalization	0.716	0.747	0.535

4.8 Caravel

For comparison, Caravel, the winning PAN15 verification system by Bagnall [3], was also tested on the obfuscated PAN15 corpus using TIRA[1].

We can see a similar classification performance decrease as in our own classifier. After 20 reduction iterations, the c@1 score drops to 67.5 % and after 50 iterations to 64 % which reflects our own classification performance after 20 and 50 iterations (cf. Figure 4.15). We can therefore conclude that KLD obfuscation works not only against our own classifier, but also against Caravel.

Additionally, a normalized PAN15 corpus (as described in Section 3.3) was tested with the primary goal of finding out how much of an impact white space formatting has on the classification accuracy. We can observe a drop by about 4 percentage points, so a certain amount of obfuscation can be achieved by removing paragraph structures and indentations from the text as well as converting words to lowercase.

[1] http://www.tira.io/

Chapter 5

PAN Setup Discussion and Flaws

In the previous chapters we showed that it is possible to create a competitive authorship verification classifier in the PAN setup using only very simple distributional features. We also showed that the performance of such a classifier can be degraded by breaking at least one of its features using basic modifications of a text's n-gram distribution.

However, while working on these classification and obfuscation methods, doubts about the quality of the PAN corpora and the general training and test setup arose.

5.1 Text Length and Test Splits

Revisiting the corpus statistics table from Section 3.2, we can see that the average text length is rather short. The PAN14 novels corpus provides text lengths of 3090–6000 words which is a decent text mass to work with, but both the PAN15 as well as the PAN14 essays corpus provide significantly shorter texts. The PAN15 training corpus with 340 words per text has by far the shortest average text length.

Table 5.1: PAN14/15 corpus statistics. Average word counts are per text.

	Cases Training	Cases Test	Avg. Words Training	Avg. Words Test
PAN15	100	500	340	510
PAN14 Novels	100	200	3090	6000
PAN14 Essays	200	200	830	820

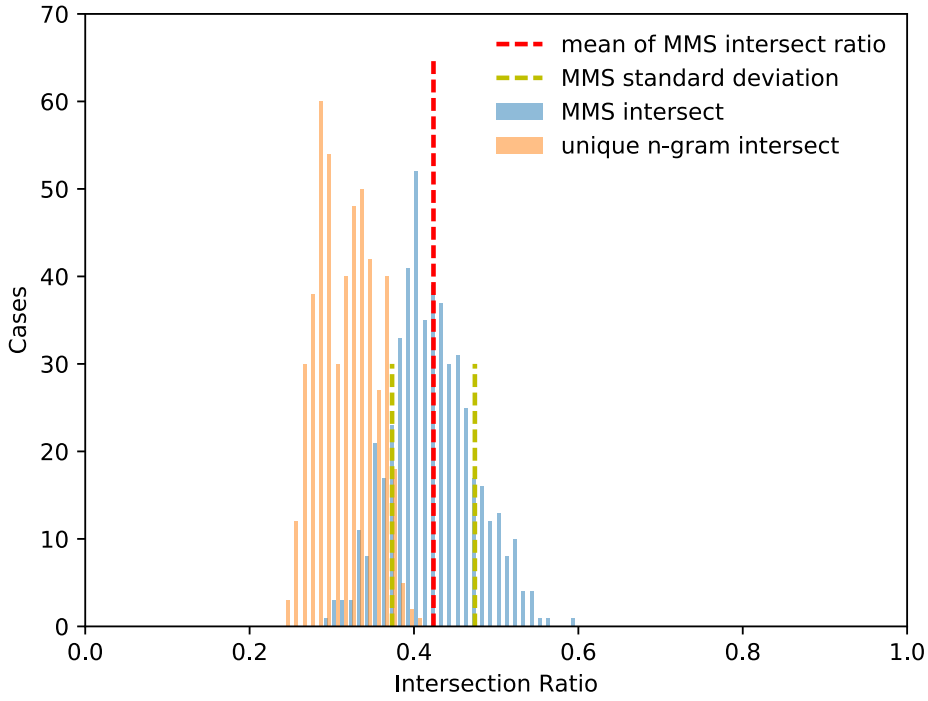


Figure 5.1: PAN15 test corpus intersection ratios of min-multiset (MMS) and unique n-grams

In Table 5.1 it can also be seen that the test sets of the PAN15 and PAN14 novels corpora are much larger than the training sets. The reason for the decision to provide only a very small training set but a much larger test set is unclear. It can only be assumed that time constraints played a role because the test set wasn't released before participants handed in their classifiers, leaving more time for creating a larger corpus.

The very short texts also explain why no participant provided a classifier directly based on the unmasking approach by Koppel and Schler [17] which has proved to be an effective method, but explicitly requires a much larger shared set of n-grams and therefore longer texts.

The short average text length, however, results in a very small common text mass. If large parts of the two texts do not have any n-grams in common, it becomes hard to assess a stylistic similarity measure between them. Figure 5.1 shows a histogram of the min-multiset intersection ratio and the set of common unique n-grams per case for the PAN15 test corpus. With an average MMS ratio below 0.5, a feature such as the KLD is calculated on less than half of

the total text mass of a case on average. The mean of unique common n-grams is even lower which means that most unique n-grams only appear in one of the texts, resulting in a very small shared vocabulary.

With texts that short it is very questionable how well the classifier features actually represent the style of an author. It is also unclear how the classifier behaves when presented with longer texts.

5.2 Number of Different Authors and Texts

It is also very concerning that the cosine similarity (TF-IDF) feature works so well for separating cases. This feature is normally used for information retrieval in large text collections and allows to search for documents which are relevant for a certain topic [25]. The fact that this feature allows to separate authors in the PAN corpus so well implies that texts written by the same author are very similar in topic.

For assessing the diversity of the PAN15 test corpus, we reconstructed the origin of the corpus texts. It turned out that all 500 cases consist of chunks of only 15 different stage plays by five unique playwrights:

- *J.M. Barrie*:
  The Admirable Crichton; Echoes of the War; Der Tag
- *W.S. Gilbert, A. Sullivan*:
  The Gondoliers; The Grand Duke
- *W.B. Hare*:
  Anita's Secret or Christmas in the Steerage; Christmas With the Mulligan's; The White Christmas; The Wishing Man
- *E. O'Neill*:
  Anna Christie; Ile; The Straw
- *J.M. Synge*:
  Deirdre of the Sorrows; In the Shadow of the Glen; The Playboy of the Western World

The training set also consists largely of the plays from the list above.

With so few authors (and therefore actual classes), it becomes questionable if a classifier that is trained on such a corpus actually solves an authorship verification problem. It appears much more likely that the classifier learns characteristics of each author (which would be authorship attribution) or even just the content of the texts and not what generally distinguishes two different authors.

Considering that we are dealing with chunks of only 15 different works, it doesn't appear too surprising anymore that TF-IDF works so well as a feature. A single *same author* case is always made of two different plays by the same author. But with only between two and four different plays per author and five authors in total to choose from, a classifier can easily just learn the topic differences between texts. Looking at the plays by Hare, certain plays by the same author are also very closely related to each other. Thus to solve the authorship problem on this corpus, a classifier doesn't have to learn anything about authors and how their styles differ. It suffices to learn the content differences between the texts.

5.3 Text Normalization

To avoid that editorial differences between texts become features, it is important to normalize corpus texts to a certain extent. Line lengths, line breaks and spaces between paragraphs can be specific for a certain author, but most likely they rather identify the editor instead of the author. The PAN corpora haven't been normalized or cleansed properly. Especially for stage plays, paragraph structures and indentations are very characteristic. The reference classifier doesn't use this feature because it performs a white space normalization itself, but exploitation of it to identify which play a text chunk comes from appears to be possible.

Other editorial features such as the number of hyphens an em dash or the number of dots an ellipsis is represented by also vary throughout the corpus.

Role names in front of lines were remove in most parts of the PAN15 corpus, but some exceptions remain. This would probably not pose a problem in a very diverse corpus, but with only 15 different sources, it may be a significant editorial feature..

The PAN14 novels corpus also still contains chapter titles whose formats vary between different books and are probably also an editorial feature and not part of an author's style.

5.4 Exploitation of Corpus-relative Features

Corpus-relative IDF weight calculation and feature normalization is an exploit of the PAN setup in which classifiers get the full test corpus instead of only one case at a time and a memory wipe in between. In a real-world AV scenario, there would most likely be only a very small set of cases to be classified and not a large corpus. On an infinitely large and perfectly diverse corpus, corpus-relative features shouldn't help improving the classification accuracy. But

with so few different literary works and authors, we are very likely to learn characteristics about these specific authors from the full corpus. In Section 3.6 I already showed how big an influence corpus-relative feature normalization has. Without normalization, the best classification accuracy was 66.2 %. With normalization, it rose to 76.8 %.

An untagged version of the Brown corpus was used to calculate IDF weights instead to assess the influence of corpus-relative IDF weights. Because of the low diversity of the PAN corpus, the accuracy was expected to drop significantly and experiments confirmed this:

Table 5.2: Classification accuracy with Brown corpus IDF weights. Shown is the accuracy of the reference classifier with corpus-relative IDF and corpus-relative normalization, corpus-relative IDF without normalization, Brown IDF with normalization and Brown IDF without normalization. Classification was done using the decision tree classifier.

	Accuracy	Precision	F-Measure	ROC AUC
Rel. IDF w/ Norm.	0.768	0.773	0.768	0.748
Rel. IDF w/o Norm.	0.622	0.684	0.651	0.639
Brown IDF w/ Norm.	0.598	0.611	0.586	0.598
Brown IDF w/o Norm.	0.590	0.605	0.575	0.590

A total accuracy decrease of 17.8 percentage points can be observed with IDF weights calculated on a larger external corpus and without corpus-relative feature normalization.

The winner of the PAN15 competition achieved an overall c@1 score of 75.7 %. Judging from this participant's freely-available implementation [3], corpus-relative features may play a role in this result. The used deep learning software trains its model on the full input corpus together with a PAN14 corpus as control corpus instead of classifying each case independently of other test cases.

Chapter 6

Developing an Alternative Corpus

To get meaningful classification (and obfuscation) results, a new corpus needed to be developed which does not inhere the mistakes of the PAN corpora. The new corpus should contain texts with much longer and more homogeneous text lengths. To avoid classification by genre or period-specific features, all texts should be of the same basic genre and time period. Additionally, no author should appear in two different cases for best possible diversity. Texts inside the corpus should also be normalized properly to avoid classification by editorial features.

6.1 Crawling and Case Building

To construct a corpus, three sets of books from Project Gutenberg[1] were crawled which written by science-fiction authors born in the 19th century, adventure authors born in the 19th century and science-fiction authors born in the 20th century. Within one set, no author appears more than once. Authors between the 19th-century collections and the 20th-century collection are also guaranteed to be unique. Two collections from the same period may contain an author at most twice. The collections were named *guten-ad-19*, *guten-sf-19* and *guten-sf-20* and will be referred to by these names in the following.

For each collection, a balanced set of cases was constructed. To avoid classification by topic-specific features, the *same author* cases were built by crawling a different publication by the same author from Project Gutenberg, requiring one third of the initially crawled texts for building text pairs. The other two thirds were used for building the remaining *different author* cases. Table 6.1 shows an overview of how many authors were crawled for each set.

All three collections were combined into on larger *gutenberg* corpus, but

[1] https://www.gutenberg.org/

Table 6.1: Number of authors crawled from Project Gutenberg and number of constructed cases. The total number of unique authors was left blank because authors are not necessarily unique across sets.

	Cases	Unique Authors (1.5 × Cases)
guten-ad-19	118	177
guten-sf-19	60	90
guten-sf-20	96	144
total	**274**	**-**

references to which collection a case belongs to were kept. This way, a relatively large corpus could be created while still being able to work individually on subsets separated by genre and period. The Gutenberg ID number was also stored in an index file to allow retrieval of meta data (such as title and author) for each text later on.

6.2 Text Cleansing and Normalization

As a first step, chapter titles, illustration placeholders, footnotes (and their references in the text) and quotations by other real or fictional people at the beginning of chapters were removed. Any editorial introductions at the beginning of a book were also removed together with any technical meta data, transcription notes and licensing information. ASCII art in the form of hyphen and pipe character runs, dots or asterisks between or around paragraphs were removed as well.

Texts crawled from Project Gutenberg were found to make heavy use of underscores to denote italic typesetting which is why any occurrences of underscores were also stripped from the texts.

Each text in all three sets was then cut to an average length of 4000-4100 white space separated words. The gutenberg-sf-20 collection shows more variation with 2000-4080 words and few individual texts as short as 1000 words. This is due to generally shorter books. Within a case, a short text was paired with another short text to avoid very unbalanced cases. The final corpus has an average text length of 21870 characters.

If possible, texts were truncated at paragraph endings. If this would lead to too short or too long texts, a truncation after the next possible sentence was performed.

After that an automatic normalization of em dashes to two hyphens and ellipses to three dots was performed.

A normalization of quotation marks proved more difficult and had to be done semi-automatically and largely manually because of major notation inconsistencies (or even notation errors) across texts and many ambiguities between quotation marks and apostrophes. All single quotes (`'`) representing quotation marks were replaced by double quotes (`"`) while preserving apostrophes as `'`. Other quotation mark notations (such as `` `' `` or `'"`) were normalized the same way. Quotations inside direct speech were normalized to double quotes also, so double quotes within double quotes do occur. Quotation marks at the beginning of a paragraph as a continuation of direct speech from the previous paragraph without a preceding closing quotation mark were found infeasible to cleanse properly and were left as is (but still replaced by double quotes).

As a last normalization step, all line breaks were replaced with a space character and all occurrences of multiple consecutive spaces were collapsed into a single space. This way the full text is in a single long line with all paragraph formatting information stripped.

6.3 Corpus Format

After all normalization was done, the cases were put into a PAN-compatible corpus format so that any PAN classifier would be able to work on it without modifications. This includes the previously developed reference classifier.

The corpus consist of a randomly shuffled and enumerated set of case folders containing the files `known01.txt` and `unknown.txt` with the two texts. The ground truth about which class a case belongs to is contained in a file `truth.txt` at the same directory level as the case folders.

6.4 Training and Test Splits

For the training splits 70 % of the cases from each collection were used. The remaining 30 % of each collection were used as test splits. It was made sure that both training and test splits always contained the same amount of *same author* and *different author* cases. With the splits from all three collections combined, this results in a training corpus of 192 cases and a test corpus of 82 cases. The obfuscator was run with a pure reduction strategy and real text replacements.

Chapter 7

Repetition of Experiments

Experiments done in the previous chapters were repeated on the new Gutenberg corpus. With generally longer, more diverse and better preprocessed texts, the gained results are expected to be more meaningful and to generalize better if repeated on another equally well-designed corpus.

7.1 Verification of Topic Diversity

As a very first step, the effectiveness of the TF-IDF feature (using both corpus-relative and Brown IDF weights) was verified using a Naive Bayes classifier with only this feature and 10-fold cross-validation on the Gutenberg training corpus. As can be seen in Table 7.1, TF-IDF with Brown IDF weights can still be used as a feature, but it's not as dominant as before (cf. Table 4.1).

The reason why the feature achieves cross-validation results above 50 % is most likely that two texts by the same author are sometimes still similar in topic, although not as closely related as in the PAN corpus. *Same author* cases were always built using two different publications by one author, but a certain relation between the two couldn't be ruled out. However, an accuracy of not even 62 % seems appropriate compared to over 74 % on the PAN corpus. With

Table 7.1: Naive Bayes 10-fold cross-validation accuracy on new training corpus using TF-IDF as only feature with corpus-relative and Brown corpus IDF weights.

	Accuracy	Precision	F-Measure	ROC AUC
Brown IDF	0.615	0.615	0.614	0.644
Corpus-rel. IDF	0.578	0.644	0.524	0.559

Table 7.2: Results for single-feature Bayes classification on the Gutenberg training corpus using 10-fold cross-validation in comparison with the results on the PAN15 test corpus.

	Accuracy Gutenberg	Accuracy PAN15
Kullback-Leibler Divergence	0.651	0.672
Skew Divergence	0.583	0.596
Jensen-Shannon Divergence	0.620	0.688
Hellinger Distance	0.615	0.692
Avg. Sentence Length Diff.	0.594	0.510
MMS Intersection Ratio	0.557	0.630

corpus-relative IDF weights, the accuracy is actually even lower.

With the knowledge that discrimination by topic is still not impossible, but also not a dominating corpus feature anymore, we can exclude this feature from the classifier from now on. This way we can ensure that we are not accidentally learning how to discriminate cases based on their topic instead of their authors' styles.

7.2 Individual Feature Evaluation

After removing the TF-IDF feature, the classification strength of each remaining individual feature was tested on the training corpus using 10-fold cross-validation. This way we can better compare the performance of individual features to their performance on the PAN corpus. The results can be seen in Table 7.2.

It can be seen that the accuracy of the distributional features are very comparable. The biggest difference can be observed for the sentence length and intersection ratio features. The sentence lengths discriminate better on the Gutenberg corpus while the intersection ratio discriminates a little worse. This can be explained with the much longer texts which allow for a better averaging of sentence lengths while the intersection ratio will be generally higher for both classes.

Figure 7.1 shows the min-multiset and unique n-gram intersection ratios of the Gutenberg training corpus. The most striking difference between the Gutenberg and PAN corpus is the min-multiset ratio mean which is around 0.7 compared to 0.41 (cf. Figure 5.1). This means that we are working on a (relatively) much larger portion of the texts thanks to more common n-gram

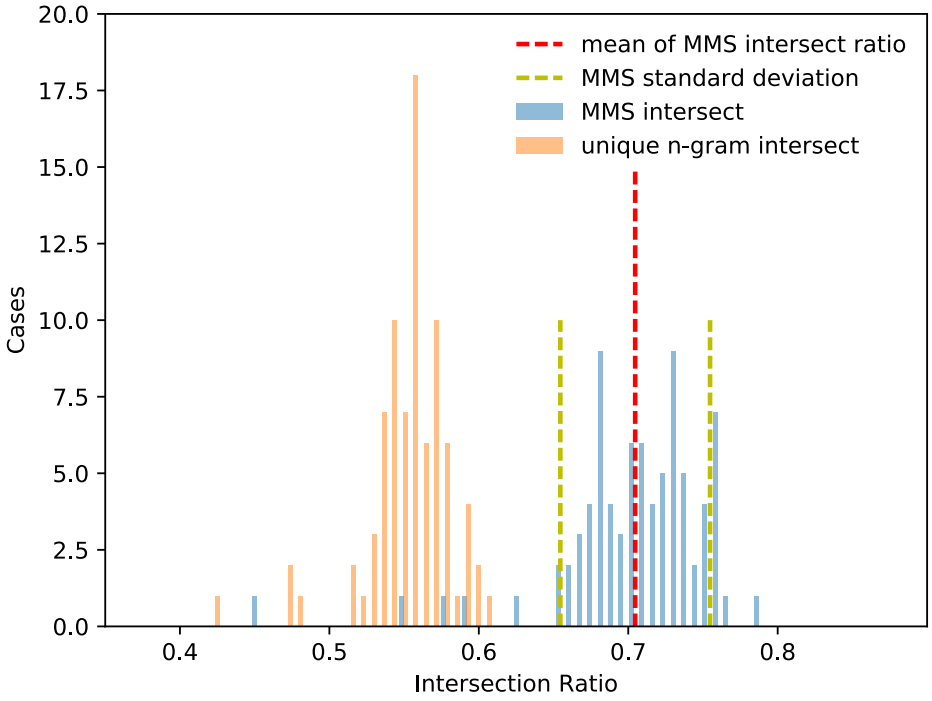


Figure 7.1: Min-multiset and unique n-gram intersection ratio on Gutenberg training corpus.

occurrences and frequencies. The mean of common unique n-grams is also a lot higher (around 0.56 compared to 0.31). Although individual feature performances are comparable, the larger common text mass makes results more reliable and less likely to be random. On the other hand, the standard deviation is a little higher due to a smaller number of cases resulting in a histogram with three peaks instead of a smooth bell-like shape.

7.3 Reference Classifier Performance

After these initial evaluations, the reference classifier with the same four different learning algorithms was tested on the Gutenberg training and test corpora. To ensure a true and unbiased classification, no corpus-relative features were used, i.e., no feature normalization was performed. The cosine similarity (TF and TF-IDF) features were removed also to avoid topical classification as mentioned above.

Table 7.3 shows the results of the reference classifier. The decision tree

Table 7.3: Reference classifier results on Gutenberg corpus without any corpus-relative or topical features.

	Accuracy	Precision	F-Measure	ROC AUC
Naive Bayes Simple	0.634	0.634	0.634	**0.750**
SVM	0.695	0.701	0.693	0.695
Decision Tree	**0.720**	**0.765**	**0.707**	0.720
Random Forest	0.695	0.705	0.695	0.747

is again the best learning algorithm on this corpus yielding an accuracy of 72 %. Compared to the best result that could be achieved on the PAN15 corpus (76.8 % with corpus-relative and topical features), this seems to be a very good result, although still not good enough for practical purposes.

7.4 Obfuscation Performance

A single text in the Gutenberg corpus has an average length of 21870 characters which corresponds to 7290 non-overlapping n-grams. As a starting point for the obfuscation, the empirically gained rule of 2.2 % text modification was applied which means that 160 iterations are needed for a full KLD obfuscation.

After running the obfuscator on the test corpus for 160 iterations, a total drop in classification accuracy by 8.6 percentage points from 72 % to 63.4 % could be observed.

Judging from the histogram shown in Figure 7.2, an iteration count of 160 is indeed enough to shift the two histograms on top of each other.

Each feature was again tested individually using a Bayes classifier with 10-fold cross-validation. The results can be seen in Table 7.4. The most interesting observations are that the KLD feature still allows a good separation while JSD and HD fall off completely. The effects of KLD obfuscation on JSD and HD are in line with our expectations and confirm the results on the PAN corpus to be a corpus-related issue. The JSD is a symmetrized version of the KLD and should therefore respond to KLD changes. On the other hand, the cross-validation accuracy of the KLD feature contradicts our expectations. When looking at the histograms in Figure 7.2, we can see that a significant shift does exist. However, we can also see that it is still possible to separate many of the cases due to an uneven histogram distribution. This is an artifact of the small sample size of only 82 cases compared to 500 in the PAN corpus. Figure 7.3 shows the JSD distribution of the text corpus before and after obfuscation. The overall

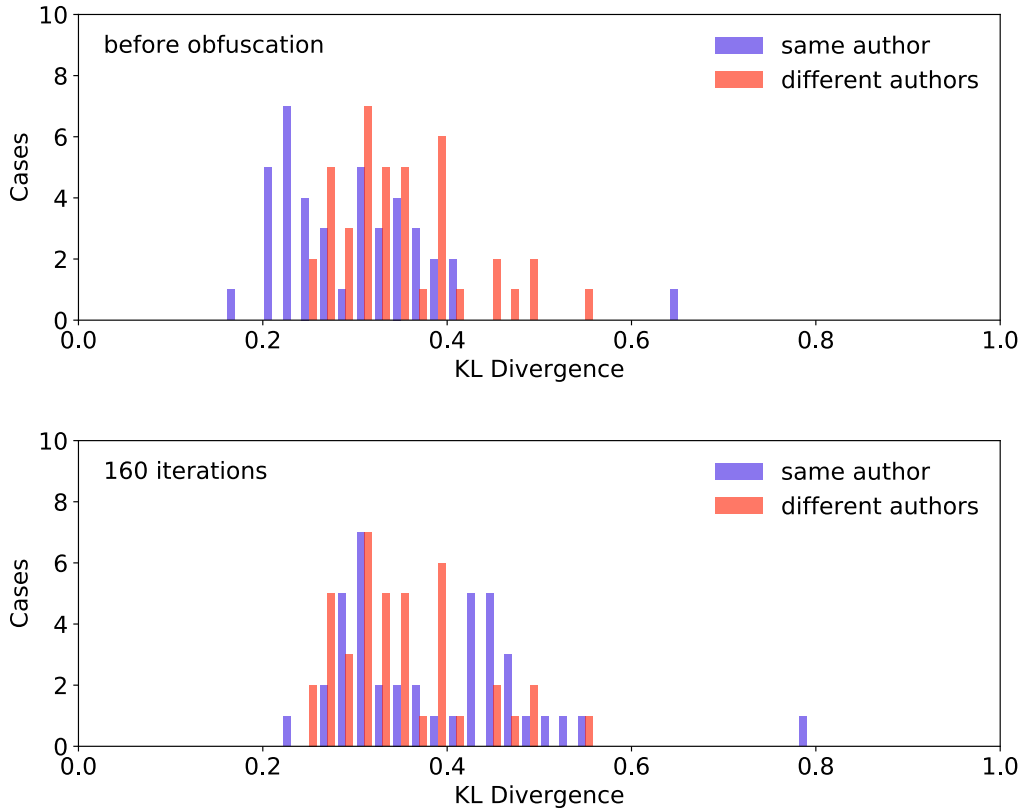


Figure 7.2: Gutenberg corpus KLD before obfuscation and after 160 reduction iterations.

Table 7.4: Cross-validation results for single-feature Bayes classification on the Gutenberg test corpus after 160 obfuscation iterations.

	Baseline Accuracy	After 160 Iterations
Kullback-Leibler Divergence	0.622	**0.646**
Skew Divergence	**0.585**	0.549
Jensen-Shannon Divergence	**0.659**	0.354
Hellinger Distance	**0.659**	0.341
Avg. Sentence Length Diff.	**0.671**	0.646
MMS Intersection Ratio	**0.621**	0.524

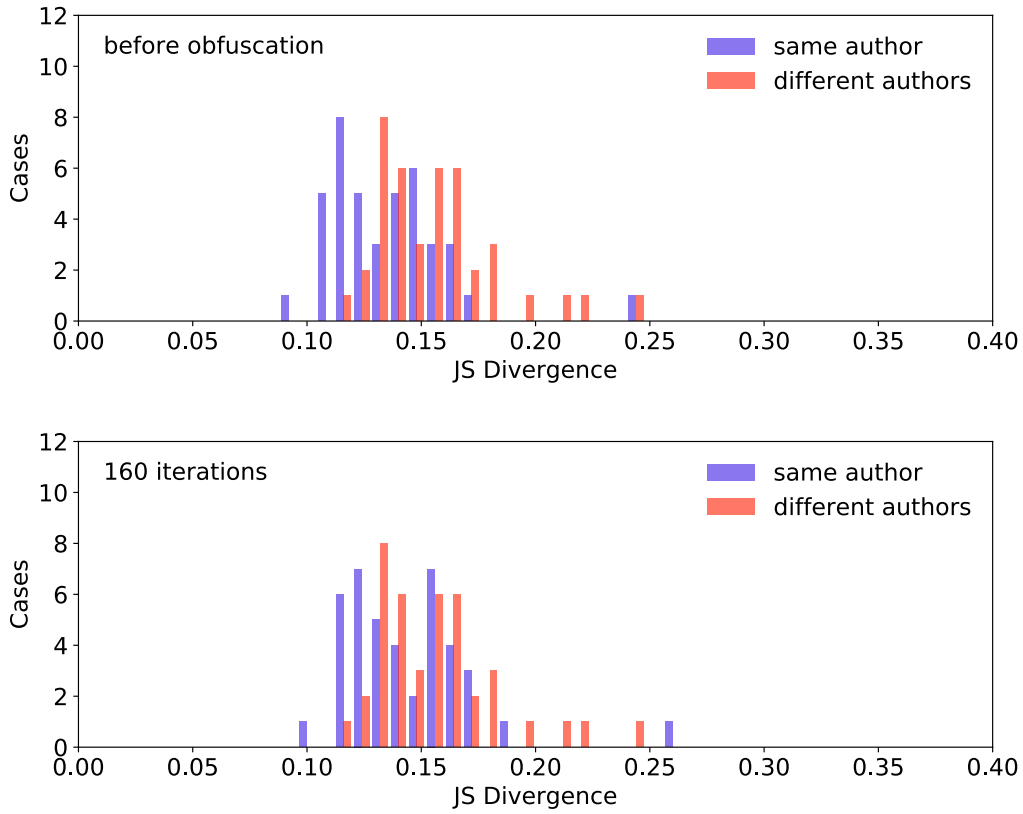


Figure 7.3: Gutenberg corpus JSD before obfuscation and after 160 reduction iterations.

histogram shift is marginally smaller than the KLD shift, but the histogram is denser than the KLD histogram resulting in a much weaker classification performance after obfuscation.

When analyzing the KLD changes by min-multiset intersection ratio (Figure 7.4), we can observe the same characteristics as before. The majority of cases with low ratio are *different author* cases while all cases with high ratio are *same author* cases. Cases around the mean contain similar amounts of both classes. After obfuscation, a shift of the standard deviation can be observed. The largest shift can be seen for high intersection ratios because of our 160 reduction iterations. This observation is in line with the performance decrease of the min-multiset intersection feature seen in Table 7.4. However, we can still classify about 11 % of the cases with 100 % accuracy using only the intersection feature. This shows that KLD obfuscation alone is not enough for reducing the overall classification performance massively. To improve the obfuscation performance, we also need to address other features specifically. The intersection

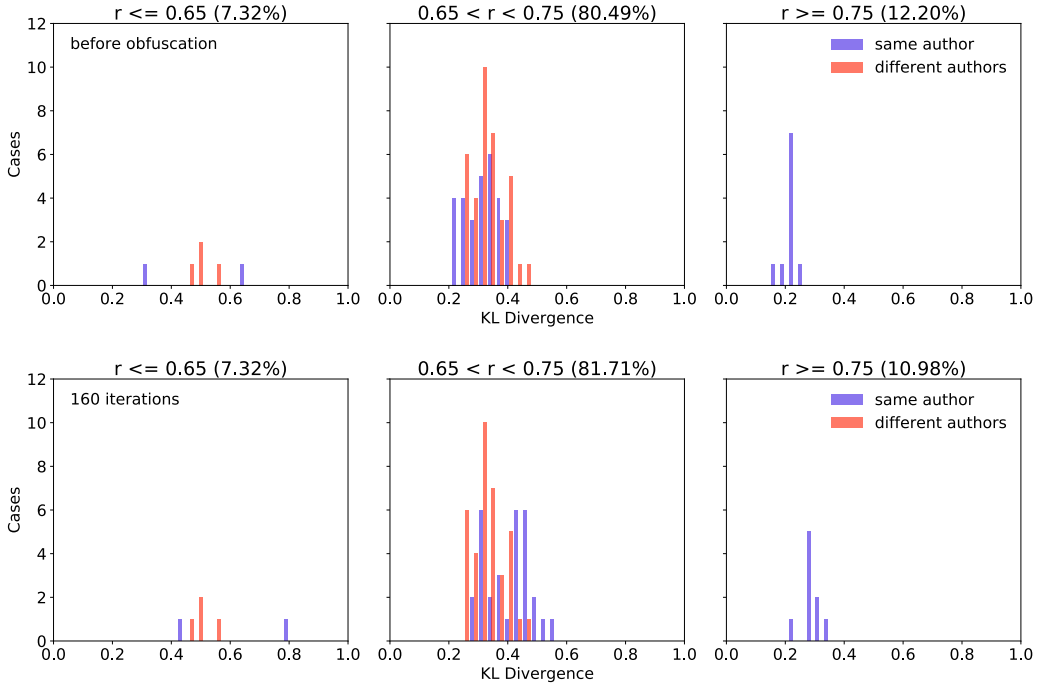


Figure 7.4: Gutenberg corpus KLD before obfuscation and after 160 reduction iterations split into windows by min-multiset intersection ratio. The min-multiset intersection feature appears to be very effective to classify a certain amount of *same author* cases even after obfuscation.

ratio feature in particular can be targeted by either just applying many more KLD obfuscation iterations (and therefore largely overobfuscating the KLD) or by reducing the frequency of n-grams which don't have a large effect on either KLD or JSD. The effects of such modifications remain to be evaluated together with other obfuscation improvements discussed in the following chapter.

7.5 Caravel

As before on the PAN15 corpus (Section 4.8), we also tested the Caravel verification system by Bagnall [3] on the new Gutenberg corpus using TIRA.

Caravel shows a very high baseline accuracy of 79.4 % which beats the 75.7 % on the PAN15 corpus and even our 72 % on the Gutenberg corpus. So it appears that although the system was specifically designed for the PAN15 verification task, it works well (and even better) on a more diverse and less topic-dependent corpus with longer texts. However, its runtime of almost 5 hours is still very high despite having a lot less cases to classify than using the

Table 7.5: Caravel results on the Gutenberg corpus before obfuscation and after 160 and 250 reduction iterations.

	c@1	ROC AUC	Final Score
Baseline Accuracy	0.794	0.845	0.671
160 Iterations	0.715	0.772	0.552
250 Iterations	0.683	0.742	0.506

PAN15 corpus.

After 160 reduction iterations, a drop of the c@1 score by about 8 percentage points to 71.5 % can be seen. After slight overobfuscation with 250 iterations, it drops by another 3 percentage points to 68.3 %. So although the classifier works better on this corpus, we can still decrease its accuracy using KLD obfuscation by an amount comparable to previous experiments on the PAN15 corpus.

Chapter 8

Summary and Future Work

We could show that simple distributional features can be used to solve an authorship verification problem with above 70 % accuracy which proved competitive among existing approaches. However, it was also shown that the actual quality of such a classifier depends heavily on the quality of the corpus. The corpora used for the PAN competitions as well as the general competition setup were not well-suited for the actual task which also puts the results of participants into question. For reliable results, a completely new corpus had to be developed. It was shown that authorship verification on this new corpus works generally better using only distributional and no topical features.

However, it also is worth noting that for real-world applications a much higher accuracy (above 90 %) is desired, so neither the developed reference classifier nor any of the PAN participants provide a classification quality that is satisfying for practical purposes. It therefore still remains an important task to develop a proper classifier which can be used as an actual baseline for further authorship verification and obfuscation experiments.

While distributional features were found to work well in general for solving the verification problem, they are also easy to compromise to achieve significant classification degradation. Obfuscating the Kullback-Leibler divergence feature alone also has an impact on other distributional features such as Jensen-Shannon divergence and Hellinger distance.

The developed obfuscation algorithm serves as a proof of concept for how a verification system relying on distributional features can be broken. It was observed that only about 2.2 % of a text need to be modified for full obfuscation of the KLD feature. Neither soundness nor sensibility of the obfuscation were taken into account at this stage, although so few needed modifications can be considered subtle. Safety of the obfuscation against de-obfuscation remains to be evaluated, but can be expected to be weak due to use of a deterministic algorithm.

8.1 Future Work

Since the task of developing a practical authorship verification baseline which can be used for testing obfuscation approaches against is still not fully solved, improving the used reference classifier appears to be the most important task. This can be achieved using several steps:

First of all, we should take full advantage of the min-multiset intersection feature. This feature allows us to optimize the classification precision at the cost of recall by not classifying cases with a very small ratio. This has been suggested earlier in this thesis but has not actually been integrated into the verification pipeline yet. Such an optimization is suitable for improving the usefulness and reliability of the classifier because it eliminates false classifications.

Furthermore, the classifier has to be extended by more features. One promising feature are co-stems as suggested by Lipka and Stein [19] and already highlighted in Section 2.1. These can be integrated in a similar way as the previously used TF-IDF feature, but don't actually contain any topic information. Further non-topical features from the writeprints feature set can also be incorporated such as the use of punctuation.

As we now have much longer texts thanks to the newly developed Gutenberg corpus, we should also integrate the unmasking approach by Koppel and Schler [17] to decide in cases for which the distributional classifier cannot give an answer with high enough certainty or as a general decision base.

With all these improvements to the classifier, we hope for an accuracy beyond 80 % which comes a lot closer to practical classification quality. Once a working classifier with satisfying classification quality exists, the obfuscation task can be revisited. It has already been shown how distributional features can be broken, but now other features definitely need to be addressed also. Kacmarcik and Gamon [14] showed a working yet unsafe obfuscation method against unmasking. Just as the classifier uses many different features now, the obfuscator also needs to target more of them for satisfying obfuscation results. Also as already discussed in the previous chapter, the min-multiset intersection ratio feature appears to perform quite well on longer texts and therefore needs to be targeted in particular.

As a last obfuscation measure, safety against de-obfuscation needs to be granted. Here the work by Thi et al. [29] can be used as a basis. Additionally, for more satisfying sensibleness of the obfuscation, a more intelligent way to systematically modify a text without breaking its grammaticality needs to be developed. It remains to be shown how automatic paraphrasing can be used to achieve this goal.

Bibliography

[1] Ahmed Abbasi and Hsinchun Chen. Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace. *ACM Trans. Inf. Syst.*, 26(2), 2008. doi: 10.1145/1344411.1344413. URL `http://doi.acm.org/10.1145/1344411.1344413`.

[2] Sadia Afroz, Michael Brennan, and Rachel Greenstadt. Detecting hoaxes, frauds, and deception in writing style online. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 461–475. IEEE Computer Society, 2012. ISBN 978-0-7695-4681-0. doi: 10.1109/SP.2012.34. URL `http://dx.doi.org/10.1109/SP.2012.34`.

[3] Douglas Bagnall. Author identification using multi-headed recurrent neural networks. In Cappellato et al. [9]. URL `http://ceur-ws.org/Vol-1391/150-CR.pdf`.

[4] Edward Gaylord Bourne. The authorship of the federalist. *The American Historical Review*, 2(3):443–460, 1897.

[5] Michael Brennan, Sadia Afroz, and Rachel Greenstadt. Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity. *ACM Trans. Inf. Syst. Secur.*, 15(3):12, 2012. doi: 10.1145/2382448.2382450. URL `http://doi.acm.org/10.1145/2382448.2382450`.

[6] Michael Robert Brennan and Rachel Greenstadt. Practical attacks against authorship recognition techniques. In Karen Zita Haigh and Nestor Rychtyckyj, editors, *Proceedings of the Twenty-First Conference on Innovative Applications of Artificial Intelligence, July 14-16, 2009, Pasadena, California, USA*. AAAI, 2009. URL `http://aaai.org/ocs/index.php/IAAI/IAAI09/paper/view/257`.

[7] Aylin Caliskan and Rachel Greenstadt. Translate once, translate twice, translate thrice and attribute: Identifying authors and machine translation

tools in translated text. In *Sixth IEEE International Conference on Semantic Computing, ICSC 2012, Palermo, Italy, September 19-21, 2012*, pages 121–125. IEEE Computer Society, 2012. ISBN 978-1-4673-4433-3. doi: 10.1109/ICSC.2012.46. URL `http://dx.doi.org/10.1109/ICSC.2012.46`.

[8] Linda Cappellato, Nicola Ferro, Martin Halvey, and Wessel Kraaij, editors. *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014*, volume 1180 of *CEUR Workshop Proceedings*, 2014. CEUR-WS.org. URL `http://ceur-ws.org/Vol-1180`.

[9] Linda Cappellato, Nicola Ferro, Gareth J. F. Jones, and Eric SanJuan, editors. *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015*, volume 1391 of *CEUR Workshop Proceedings*, 2015. CEUR-WS.org. URL `http://ceur-ws.org/Vol-1391`.

[10] Jonathan H. Clark and Charles J. Hannon. A classifier system for author recognition using synonym-based features. In *MICAI*, pages 839–849, 2007. doi: 10.1007/978-3-540-76631-5_80. URL `http://dx.doi.org/10.1007/978-3-540-76631-5_80`.

[11] Jordan Fréry, Christine Largeron, and Mihaela Juganaru-Mathieu. UJM at CLEF in author identification notebook for PAN at CLEF 2014. In Cappellato et al. [8], pages 1042–1048. URL `http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-FreryEt2014.pdf`.

[12] Patrick Juola and Darren Vescovi. Empirical evaluation of authorship obfuscation using JGAAP. In Rachel Greenstadt, editor, *Proceedings of the 3rd ACM Workshop on Security and Artificial Intelligence, AISec 2010, Chicago, Illinois, USA, October 8, 2010*, pages 14–18. ACM, 2010. ISBN 978-1-4503-0088-9. doi: 10.1145/1866423.1866427. URL `http://doi.acm.org/10.1145/1866423.1866427`.

[13] Patrick Juola and Darren Vescovi. Analyzing stylometric approaches to author obfuscation. In Gilbert L. Peterson and Sujeet Shenoi, editors, *Advances in Digital Forensics VII - 7th IFIP WG 11.9 International Conference on Digital Forensics, Orlando, FL, USA, January 31 - February 2, 2011, Revised Selected Papers*, volume 361 of *IFIP Advances in Information and Communication Technology*, pages 115–125. Springer, 2011. ISBN 978-3-642-24211-3. doi: 10.1007/978-3-642-24212-0_9. URL `http://dx.doi.org/10.1007/978-3-642-24212-0_9`.

[14] Gary Kacmarcik and Michael Gamon. Obfuscating document stylometry to preserve author anonymity. In Nicoletta Calzolari, Claire Cardie, and Pierre Isabelle, editors, *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*. The Association for Computer Linguistics, 2006. URL `http://aclweb.org/anthology/P06-2058`.

[15] Mahmoud Khonji and Youssef Iraqi. A slightly-modified gi-based author-verifier with lots of features (ASGALF). In Cappellato et al. [8], pages 977–983. URL `http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-KonijEt2014.pdf`.

[16] Foaad Khosmood and Robert Levinson. Automatic synonym and phrase replacement show promise for style transformation. In Sorin Draghici, Taghi M. Khoshgoftaar, Vasile Palade, Witold Pedrycz, M. Arif Wani, and Xingquan Zhu, editors, *The Ninth International Conference on Machine Learning and Applications, ICMLA 2010, Washington, DC, USA, 12-14 December 2010*, pages 958–961. IEEE Computer Society, 2010. ISBN 978-0-7695-4300-0. doi: 10.1109/ICMLA.2010.153. URL `http://dx.doi.org/10.1109/ICMLA.2010.153`.

[17] Moshe Koppel and Jonathan Schler. Authorship verification as a one-class classification problem. In Carla E. Brodley, editor, *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004. doi: 10.1145/1015330.1015448. URL `http://doi.acm.org/10.1145/1015330.1015448`.

[18] Moshe Koppel and Yaron Winter. Determining if two documents are written by the same author. *JASIST*, 65(1):178–187, 2014. doi: 10.1002/asi.22954. URL `http://dx.doi.org/10.1002/asi.22954`.

[19] Nedim Lipka and Benno Stein. Classifying with co-stems - A new representation for information filtering. In Paul D. Clough, Colum Foley, Cathal Gurrin, Gareth J. F. Jones, Wessel Kraaij, Hyowon Lee, and Vanessa Murdock, editors, *Advances in Information Retrieval - 33rd European Conference on IR Research, ECIR 2011, Dublin, Ireland, April 18-21, 2011. Proceedings*, volume 6611 of *Lecture Notes in Computer Science*, pages 307–313. Springer, 2011. ISBN 978-3-642-20160-8. doi: 10.1007/978-3-642-20161-5_31. URL `http://dx.doi.org/10.1007/978-3-642-20161-5_31`.

[20] Andrew W. E. McDonald, Sadia Afroz, Aylin Caliskan, Ariel Stolerman, and Rachel Greenstadt. Use fewer instances of the letter "i": Toward

writing style anonymization. In Simone Fischer-Hübner and Matthew K. Wright, editors, *Privacy Enhancing Technologies - 12th International Symposium, PETS 2012, Vigo, Spain, July 11-13, 2012. Proceedings*, volume 7384 of *Lecture Notes in Computer Science*, pages 299–318. Springer, 2012. ISBN 978-3-642-31679-1. doi: 10.1007/978-3-642-31680-7_16. URL http://dx.doi.org/10.1007/978-3-642-31680-7_16.

[21] Andrew W. E. McDonald, Jeffrey Ulman, Marc Barrowclift, and Rachel Greenstadt. Anonymouth revamped: Getting closer to stylometric anonymity. In *PETools: Workshop on Privacy Enhancing Tools*, volume 20, 2013. URL http://petools.soic.indiana.edu/files/2013/06/petools2013_submission_7-1.pdf.

[22] Pashutan Modaresi and Philipp Gross. A language independent author verifier using fuzzy c-means clustering. In Cappellato et al. [8], pages 1084–1091. URL http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-ModaresiEt2014.pdf.

[23] Anselmo Peñas and Álvaro Rodrigo. A simple measure to assess non-response. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 1415–1424. The Association for Computer Linguistics, 2011. ISBN 978-1-932432-87-9. URL http://www.aclweb.org/anthology/P11-1142.

[24] Martin Potthast, Matthias Hagen, and Benno Stein. Author obfuscation: Attacking the state of the art in authorship verification. In Krisztian Balog, Linda Cappellato, Nicola Ferro, and Craig Macdonald, editors, *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016.*, volume 1609 of *CEUR Workshop Proceedings*, pages 716–749. CEUR-WS.org, 2016. URL http://ceur-ws.org/Vol-1609/16090716.pdf.

[25] Juan Ramos. Using TF-IDF to Determine Word Relevance in Document Queries. Technical report, 2003.

[26] Efstathios Stamatatos. A survey of modern authorship attribution methods. *JASIST*, 60(3):538–556, 2009. doi: 10.1002/asi.21001. URL http://dx.doi.org/10.1002/asi.21001.

[27] Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Benno Stein, Martin Potthast, Patrick Juola, Miguel A. Sánchez-Pérez, and Alberto

Barrón-Cedeño. Overview of the author identification task at PAN 2014. In Cappellato et al. [8], pages 877–897. URL `http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-StamatosEt2014.pdf`.

[28] Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Patrick Juola, Aurelio López-López, Martin Potthast, and Benno Stein. Overview of the author identification task at PAN 2015. In Cappellato et al. [9]. URL `http://ceur-ws.org/Vol-1391/inv-pap3-CR.pdf`.

[29] Hoi Le Thi, Reihaneh Safavi-Naini, and Asadullah Al Galib. Secure obfuscation of authoring style. In Raja Naeem Akram and Sushil Jajodia, editors, *Information Security Theory and Practice - 9th IFIP WG 11.2 International Conference, WISTP 2015 Heraklion, Crete, Greece, August 24-25, 2015 Proceedings*, volume 9311 of *Lecture Notes in Computer Science*, pages 88–103. Springer, 2015. ISBN 978-3-319-24017-6. doi: 10.1007/978-3-319-24018-3_6. URL `http://dx.doi.org/10.1007/978-3-319-24018-3_6`.

[30] Ying Zhao, Justin Zobel, and Phil Vines. Using relative entropy for authorship attribution. In Hwee Tou Ng, Mun-Kew Leong, Min-Yen Kan, and Dong-Hong Ji, editors, *Information Retrieval Technology, Third Asia Information Retrieval Symposium, AIRS 2006, Singapore, October 16-18, 2006, Proceedings*, volume 4182 of *Lecture Notes in Computer Science*, pages 92–105. Springer, 2006. ISBN 3-540-45780-1. doi: 10.1007/11880592_8. URL `http://dx.doi.org/10.1007/11880592_8`.

[31] Rong Zheng, Jiexun Li, Hsinchun Chen, and Zan Huang. A framework for authorship identification of online messages: Writing-style features and classification techniques. *JASIST*, 57(3):378–393, 2006. doi: 10.1002/asi.20316. URL `http://dx.doi.org/10.1002/asi.20316`.