Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Computer Science and Media

Authorship Obfuscation Using Heuristic Search

Master's Thesis

Janek Bevendorff

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

Submission date: June 15, 2018

Declaration

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

Weimar, June 15, 2018

. . . . . .
Janek Bevendorff

Abstract

In this thesis, we discuss the Jensen-Shannon divergence as a model for authorship and based on it, we present an approach to *obfuscating* a text's authorship in order to impede automated authorship detection. The incentive behind obfuscating a text is that automated detection of its authorship can potentially pose a privacy issue or even life-threatening security risk to the author. We define methods and thresholds that allow this kind of authorship obfuscation with minimal effort and develop an obfuscation framework, which uses heuristic search to cautiously and inconspicuously paraphrase a text with the goal to remove stylistic properties which would allow conclusions to be drawn about the text's original author.

Contents

Chapter 1

Introduction

Authorship analysis, and more specifically authorship verification and its related discipline authorship attribution, is a class of forensic techniques for determining whether two texts were written by the same author or for attributing an unknown text to a known author. An common scenario in which authorship analysis is an indisputably valuable tool is attribution of historical publications to a suspected known author, but also contemporary forensic applications in the domain of crime solving, fraud detection, or detection of fake online reviews and comments benefit from authorship analysis. But what is a decent tool in these scenarios may also be abused by governments and regimes for malicious prosecution. In a less critical context, the ability to deduce authorship of a piece of text may also raise general privacy concerns in our information age, in which a large part of our lives evolves around social media. These circumstances hereby create a legitimate demand for anonymization of a person's writing. On the other hand, understanding how an adversary could circumvent detection by their writing style is crucial for building more reliable verification schemes.

The modification of a text in order to disguise the original author is called *author masking*. Instead of only masking an author, we might also try to imitate another one, which would be *authorship imitation*. Either technique is a form of what we call *authorship obfuscation*, which is the general term we will use. During our previous research [4], we sketched an approach to obfuscating a text by attacking the Kullback-Leibler divergence between the character trigram distributions of the text in question and a pool of one or multiple other known texts by the same author. With the Kullback-Leibler divergence (and other measures based on it), we found a simple and effective authorship model. This model can be systematically attacked by removal or insertion of specific n-grams and therefore allows for target-oriented obfuscation, rather than random text modifications which we can only hope to be effective.

While this basic approach worked well against all tested state-of-the-art authorship verification schemes, it still remains a proof of concept and obfuscated parts of a text are rather easily noticeable and—up to a certain extent—even reversible by a human reader or an automatic spell checker. The next step in successfully obfuscating a text's authorship is not only fooling an automated standard verification scheme, but also fooling more advanced or specialized verification methods and possibly human readers. It is clear that we need better text transformations than simple deletion or insertion of individual n-grams, which obviously lacks even the most basic syntactic or grammatical considerations. The way we can achieve this is automatic paraphrasing of the parts of a text which need obfuscation. Paraphrasing has to be done in a way so that the final result looks like an original text with no or at least as few obvious gaps or other spelling and grammar mistakes as possible.

For this thesis, we first gather more advanced insights into obfuscation strategies based on the Jensen-Shannon divergence, a symmetric variant of the Kullback-Leibler divergence, and develop a generalized version of the unmasking verification scheme by Koppel and Schler [31] as a robust adversarial baseline to test our obfuscation against. We can show the effectiveness of our proof-of-concept approach against this unmasking variant and other state-of-the-art authorship verification techniques. As a first constructive obfuscation approach, we then formulate systematic obfuscative paraphrase generation as a search task solvable using the A* algorithm [20]. A* is an informed search algorithm which utilizes a heuristic to find an optimal solution in a state space graph much faster than an uninformed exhaustive search. Since we can generate almost arbitrarily many paraphrases for a selected word or paragraph and then again arbitrarily many for the next one, our state space grows exponentially. If we want to find the best-possible sequence of paraphrases in this space, an informed search is crucial for success, whereas finding a solution by uninformed exhaustive search is practically impossible. We implemented this constructive approach as an efficient C++ search framework and find that, although sensible heuristic paraphrase generation remains a very difficult task, we can already generate obfuscations of higher text quality with much fewer operations than our previous proof-of-concept approach.

Chapter 2

Related Work

Forensic authorship analysis is a field of research that dates back to at least the late 19th century [5]. While early methods relied exclusively on manual analysis by linguistic experts, modern approaches tend to use computers for automatic and deeper text analysis.

Various stylometric features have been proposed in the literature so far for automatic authorship analysis. Abbasi and Chen [1] propose *writeprints*, a comprehensive set of over twenty lexical, syntactic, and structural text features, which has since been used extensively as a basis for further research in the field of authorship attribution, verification and obfuscation [21, 33, 37, 39, 54]. A similar but smaller and more condensed feature set has been proposed by Brennan et al. [7] under the name *Basic-9*.

Instead of relying on classification by a rich set of features, Zhao et al. [53] suggest a slightly different approach. They only extract POS tags of the texts in question and interpret style differences as an encoding problem measurable by the Kullback-Leibler divergence. We used the idea of measuring stylistic differences between texts by relative entropy as a basis for implementing a simple yet competitive verification scheme and a successful obfuscation attack based on character n-gram features [4]. A related approach was published earlier by Khmelev and Teahan [29], Teahan and Harper [49], who use PPM-based [11] compression models for exploiting the better compressibility of stylistically similar and thereby more predictable texts. This method has since been adapted and improved by Halvani et al. [18, 19].

In order to measure deeper underlying stylistic differences, Koppel and Schler [31] developed the unmasking approach for which a number of discriminatory text features are extracted and iteratively removed, effectively reducing the differentiability between the two source texts. The idea behind this approach is that texts written by the same author only differ in few superficial features. By

removing those superficial features, differentiability between texts by the same author is expected to degrade faster than for texts written by different authors.

Besides individual publications, three shared tasks have been organized at PAN [23, 45, 46] in the field of authorship verification since 2013. The winning approach of the most recent competition was submitted by Bagnall [2] and is based on a deep neural network. Software submitted to PAN is archived on the TIRA platform [13, 41] for later reevaluation to allow reproducibility.

One of the first to publish about measures for confusing authorship analysis techniques were Rao and Rohatgi [43], who used round-trip machine translation as a way of masking the original author of a text. This method was later reviewed by Brennan et al. [7], who coined the term *adversarial stylometry* for obfuscating stylometric features of a text. In their study, they found machine translation to be ineffective. Instead of relying on automated obfuscation techniques, they developed the Brennan-Greenstadt corpus of adversarial texts, for which untrained authors deliberately tried to mask their style or imitate the style of another author. Brennan et al. could show that manual obfuscation was more effective against the writeprints technique by Abbasi and Chen [1] and their own Basic-9 feature set than obfuscation by machine translation. Further studies by Juola and Vescovi [24, 25] confirm these results. Beyond these findings, Caliskan and Greenstadt [9] could even attribute translated texts to the used translation engine, which might pose another attack vector for de-anonymization.

While standard machine translation appears to be ineffective and due to its blackbox character also quite uncontrollable, Xu et al. [52] used within-language machine translation techniques to translate directly between writing styles. The applicability of this approach remains questionable, since it requires a rather large mass of parallel training data of texts in different styles.

Another obfuscation technique that directly targets Koppel and Schler's unmasking was published by Kacmarcik and Gamon [26]. In what they call *deep obfuscation*, they iteratively extract and remove the most discriminatory text features, successfully degrading classification quality of an unmasking verifier. They could also show that by attacking the right features, only few obfuscations have to be performed in order to achieve adequate obfuscation performance. As a more general tool to assist authors in writing non-attributable texts, the semi-automatic Anonymouth obfuscation software was published by McDonald et al. [33, 34]. The software determines the most important stylistic features and gives suggestions how to rewrite a text in order to remain anonymous.

Although all mentioned obfuscation approaches successfully degrade the performance of selected verification techniques, a missing piece was a larger-scale comparison of obfuscation softwares against state-of-the-art verification schemes. So, following up on the 2013–2015 verification tasks, the PAN 2016 and 2017

challenges [17, 42] featured an authorship obfuscation task whose submissions were tested against verifiers submitted during previous years. To evaluate the quality of a submitted obfuscator, the task employed three evaluation dimensions:

- **safety**: obfuscated texts cannot be attributed to their original authors,
- **soundness**: obfuscated texts are textually entailed by their originals,
- **sensibleness**: obfuscated texts are well-formed and inconspicuous.

Five participants submitted their obfuscation approaches [3, 10, 28, 32, 35], who were all rated rather low on at least two of these three scales with soundness and sensibleness appearing to be the hardest dimensions to score on.

One of the main insights gained from the PAN obfuscation task is that most state-of-the-art verification approaches are very susceptible to simple obfuscation techniques. However, Thi et al. [50] could show that many, especially deterministic, obfuscation schemes are reversible. Independently, Juola [22] also found that, although an obfuscated text may not necessarily reveal its original author, the fact that it is obfuscated can quite easily be detected. As a result, we have to assume that many existing obfuscation schemes are ultimately unsafe against more sophisticated verifiers or human experts.

In order to score higher in terms of soundness and sensibleness and ultimately also safety against reversal, a text needs to be rephrased more carefully. To avoid the pitfalls of obfuscation by machine translation or other unguided paraphrasing approaches, an applicable paraphrasing technique needs to perform controlled text transformations on specific parts of the text. To the best of our knowledge, not much research has been published in this area. Our main inspiration comes from a publication by Stein et al. [48], who propose using heuristic search for paraphrasing texts in a controlled manner with the goal of generating acrostics. Since then, attention-based neural networks have been shown to be useful for performing targeted edits on a text. In particular, Grangier and Auli [15] suggest a technique for paraphrasing designated parts of a sentence for semi-automatic revision of machine translation results and Guu et al. [16] propose a general editing technique using neural edit vectors. These paraphrasing approaches have not yet been tested as obfuscation techniques, but establish interesting new research directions.

Chapter 3

Authorship Verification

For measuring differences in author style, we assume statistical differences between the feature distributions of two or more texts. These differences can be hidden in a plethora of different feature categories, including the use of function words and non-function words, morphology, punctuation, capitalization etc. We explicitly exclude structural features such as line and paragraph breaks. While they may be significant for lyrics or poems, it is unclear if they can be attributed to the original author in prose texts. The safe choice is to regard them as editorial features and to not include them in our authorship analysis.

3.1 Relative Entropy for Measuring Authorship

Capturing all stylistic markers can lead to an extensive feature set of the size of the writeprints feature set [1], but in our previous studies on verification, we employed the much simpler approach of only considering character trigrams [4]. The advantage of character-level n-grams compared to word-level n-grams is that they capture morphological structures and the use of punctuation. When white space is normalized by collapsing all whitespace characters to a single space symbol, character trigrams are sufficient to not only capture word boundaries, but also link the end of a word to the beginning of the next word. Another interesting property of character-level n-grams is that to a certain extent, they even capture phonological features as the same n-gram may occur in different words which may be semantically different, but phonologically, or at least orthographically,similar. Character n-grams therefore represent a text on a much more fundamental level than word n-grams or other word-based features. Since character n-grams make no explicit assumptions about grammatical and lexical properties of the source document, they are language-independent. Our analysis is done on English texts only, but can most likely be applied to many other languages by utilizing a Unicode alphabet or even just byte-level n-grams.

A disadvantage of character n-grams (and of any other lower-order n-gram model) is that they perform poorly in modelling long-range dependencies like semantic relationships between the beginning and the end of a sentence and between sentences or paragraphs. This may be addressed by using higher-order dependency n-grams as additional features, but for simplicity, the use of character n-grams in our analysis shall suffice.

3.1.1 Kullback-Leibler Divergence as Authorship Model

With character trigrams as features, we need a measure to determine the stylistic similarity between two texts. For our approach, we interpret the authorship problem as an encoding problem. Given an author A, we imagine a code $\mathcal{C}_A$ that optimally encodes documents $\mathbf{d}_j^A \in \{\mathbf{d}_0^A, \mathbf{d}_1^A, \ldots, \mathbf{d}_k^A\}$ written in the style of author A. To measure how much a document $\mathbf{d}_j^B$ written by another author B with code $\mathcal{C}_B$ differs in style, we measure how much encoding overhead representing $\mathbf{d}_j^B$ in $\mathcal{C}_A$ would incur. This additional encoding overhead is called *relative entropy* and can be measured by the *Kullback-Leibler divergence* (KLD), which is defined as

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

where P and Q are discrete probability distributions corresponding to the codes $\mathcal{C}_B$ and $\mathcal{C}_A$. If the base-2 logarithm is used, the unit of the KLD is bits. For true probability distributions, the KLD is always non-negative. An important property of the KLD is its asymmetry, i.e.

$$\mathrm{KLD}(P\|Q) \neq \mathrm{KLD}(Q\|P) \ . \tag{3.2}$$

The KLD is therefore not a metric and the direction, in which probability distributions are compared, matters. KLD is directly related to cross entropy, which is defined as the total number of bits required to encode a sample of P in Q:

$$\mathrm{H}(P\|Q) = \mathrm{H}(P) + \mathrm{KLD}(P\|Q) \ , \tag{3.3}$$

with $\mathrm{H}(P)$ being the Shannon entropy of P.

3.1.2 Jensen-Shannon Distance as Authorship Metric

Considering its asymmetry, the pure KLD has a few disadvantages. When comparing two texts, it is not clear, which character distribution should be P and which should be Q. The answer to the authorship question should stay

the same, no matter the direction in which they are compared. Two texts were either written by the same author or not. Different values depending on the direction of comparison are hard to interpret and to justify under this problem definition.

The KLD is also only defined for distributions P and Q where $Q[i] = 0$ implies $P[i] = 0$. Conversely, $P[i] = 0$ implies a zero summand, eliminating it from the KLD sum completely. As a result, we have a problem with unseen n-grams. To solve this problem, we would either have to renormalize the distributions to only include the subset of shared n-grams or estimate probabilities for unseen n-grams using discounting or smoothing (common smoothing methods being Good-Turing [14] or Kneser-Ney [30] smoothing). The problem with either approach is that we are reducing or artificially modifying our feature set substantially, which gets worse the sparser the data is. Taking into account that we often only have essay-length texts to work with, the shared n-gram subset will most likely only contain half of the total n-grams or perhaps even significantly less. Applying smoothing to the rest of the n-grams, would skew the distribution massively and only considering the shared subset of n-grams would result in an incomplete assessment of the examined texts. To avoid these kinds of problems, we actually use a different KLD-based difference measure, the *Jensen-Shannon divergence* (JSD), which is defined as:

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

with

$$M = \frac{P+Q}{2} \ . \tag{3.5}$$

By constructing an artificial distribution M as the midpoint between P and Q, we circumvent the problem of samples of one distribution being unknown in the other. Since $M[i]$ is never 0 for any i with $P[i] + Q[i] > 0$, one of the terms $\mathrm{KLD}(P\|M)$ or $\mathrm{KLD}(Q\|M)$ will always be non-zero. Thereby if we are missing a sample in one of the distributions (be it because it genuinely does not exist or because the sample size is too small), the sample from the other distribution will still contribute to the final sum. Using the base-2 logarithm, the JSD is bounded by the $[0, 1]$ interval.

A nice side effect of these properties and the commutativity of addition is that the JSD is in fact symmetric. The square root of twice the JSD—called the *Jensen-Shannon distance* JS_Δ—also fulfils the triangle inequality, making it an actual metric [12]:

$$\mathrm{JS}_\Delta(P, Q) = \sqrt{2 \cdot \mathrm{JSD}(P\|Q)} \ . \tag{3.6}$$

As discussed before, a symmetric measure fits our authorship problem much better and is particularly useful in anticipation of our goal of using it as the basis for a heuristic to measure paraphrasing obfuscation progress.

3.2 Distributional Authorship Verification

The simple authorship verification approach we developed earlier [4] served as a proof-of-concept baseline to show that competitive authorship verification based on the JS_Δ is possible with only little effort and our KLD model does indeed capture an author's style. We could also show that while effective, this model is trivial to fool by attacking the n-grams which are most significant for the KLD. Due to its triviality, we only mention it here for completeness and will not consider it further for this thesis.

3.3 Verification by Unmasking

One of the most accurate and robust verification techniques as of today is the previously mentioned unmasking technique by Koppel and Schler [31]. We use unmasking as a representative and robust state-of-the-art adversarial baseline to evaluate our obfuscation approach against. If we manage to confuse unmasking, we obtain evidence that our JSD-based obfuscation technique works on a deeper level than just changing a few superficial text features. We also gain at least anecdotal evidence that distributional differences are fundamental to authorship analysis—even for verification approaches which do not use the KLD or JSD explicitly as a feature. Another advantage of unmasking as a baseline is that it produces descriptive and human-interpretable curve plots. These plots make obfuscation effects and their strength immediately visible without having to rely on blackbox machine learning techniques to make sense of otherwise hard-to-understand numerical features and thresholds.

Given two texts for which we want to decide if they share the same authorship, unmasking works as follows:

1. split the two texts into chunks of at least 500 words each,
2. from each chunk extract a set of features,
3. using a linear model, determine classification accuracy between feature vectors using cross validation,
4. eliminate the m features with highest absolute weights,
5. go to Step 3 if there are still features left.

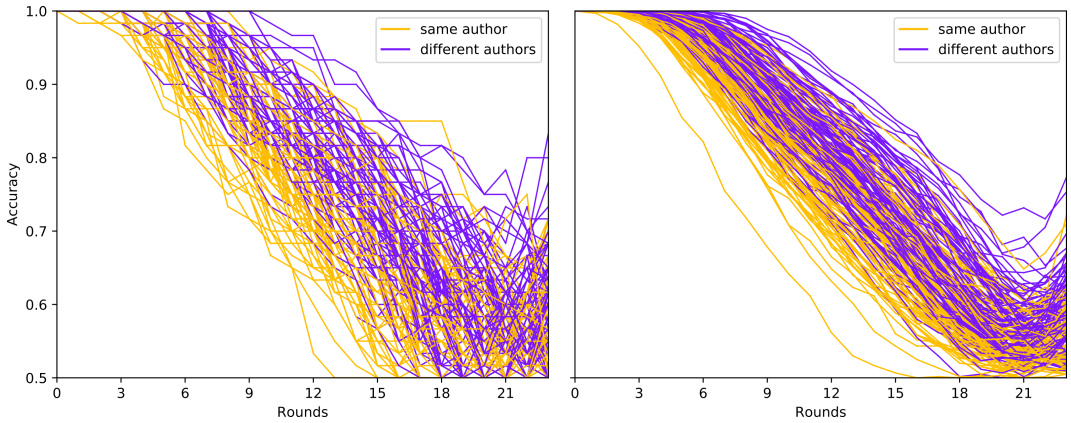

Figure 3.1: Unmasking curves generated on our training corpus by 10-fold cross validation on 30 chunks per text of 700 words each. The left plot shows the results of a single run, the right plot shows the final aggregation of 10 successive runs.

Following this algorithm, every text pair yields a curve of declining accuracy values with the curves for similar texts declining faster than those for inherently different texts. The quickly declining curves are the *same-author* candidates. With enough curves generated on a corpus of labelled text pairs, a meta model can be trained to distinguish between curves for *same-author* and *different-authors* pairs.

3.3.1 Unmasking for Short Texts by Chunk Expansion

The main weakness of unmasking is that it requires 'sufficiently large' text samples to produce reliable results. 'Sufficient' in this context means about 'book-sized'. Unmasking relies on splitting texts into enough chunks so that they can serve as training samples. Since these chunks should have a size of at least 500 words for reliable results, short texts do not provide enough material to produce an adequate number of chunks.

To alleviate this shortcoming, we propose a new variant of unmasking which also works reasonably well on texts much shorter than book length. We tested this new method successfully on the corpus we developed to avoid the pitfalls of the PAN corpora [4], which consists of a class-balanced set of 182 training and 80 test pairs and individual text lengths of about 4,000 words on average.

To create more training samples from these shorter texts, one could generate overlapping chunks, but this would result in many almost identical training samples. Instead, we use the original text as a random word pool and over-sample by drawing without replacement. Once all words have been drawn, the pool is replenished. This way, we can generate an arbitrary number of chunks of sufficiently many words even from short texts. As feature set, we use the

frequencies inside each chunk of the k most frequent words in the word pool. In accordance with Koppel and Schler and our own experiments, we use $k = 250$ and set cross validation to be 10-fold. As chunk size, we recommend 500–700 words. To counteract the high variance introduced by random oversampling, we can smooth the curves by creating very many chunks or averaging multiple successive experiments. The chosen strategy determines the curves' characteristics. More chunks produce shallower curves, less chunks steeper curves. In either case, the curves become smoother the more experiments are averaged. Averaging multiple experiments also allows for variation of the parameters for chunk size, number of features and cross validation folds. For our experiments, we always use 30 chunks of 700 words each.

3.3.2 Precision Over Accuracy

Although working rather reliably on very long texts, authorship verification is far from a solved problem and state-of-the-art approaches keep struggling especially with shorter texts. So it becomes all the more important to find a good measure for assessing the quality of a verifier. Due to high uncertainty of many results, a standard accuracy measure is perhaps not an optimal choice. The winner of the 2015 PAN challenge [46] reached an accuracy of about 75 %, which means that one in four decisions is still wrong. If applied in a real-word forensic scenario, such a verification scheme could give hints to whether two texts share authorship at best. But instead of trying to develop verifiers with perfect accuracy, we could already build more useful tools today by optimizing for precision and sacrificing recall. Unfortunately, this approach—although more useful in general—does not perform well in a setting where we measure accuracy. So in order to provide a more suitable evaluation quantity, PAN adopted the c@1 measure by Peñas and Rodrigo [40]:

$$\frac{1}{n}\left(n_{ac} + \frac{n_{ac}}{n} \cdot n_u\right) \ , \tag{3.7}$$

where n denotes the number of problems, n_{ac} the number of correct answers and n_u the number of non-answers. The c@1 measure solves the quality measure problem by rewarding non-answers in that it assigns them the same accuracy as the rest of the problems. All-correct answers still yield a score of 1, all-wrong or completely unanswered problem sets a score of 0. So with this measure, a verifier can choose not to answer a given problem if it is too uncertain to give a definitive answer and still receive a reasonably good score. This way, we can build systems with high precision and have that fact reflected in the quality measure, even if the overall sensitivity (or recall) of the system is lower.

None of the PAN participants explicitly or successfully exploited the c@1 measure, so c@1 scores of all submitted approaches are roughly the same as their accuracy.

A problem with c@1 is that it remains a measure for binary classification quality. If we are only interested in whether two texts were written by the same author, but do not care about the other case, c@1 does not fit our approach well. However, measuring only precision without taking into account recall, is not helpful either. For that reason, we propose as an alternative the $F_{0.5}$ measure with the only alteration that we treat non-answers as false negatives:

$$\frac{1.25 \cdot n_{tp}}{1.25 \cdot n_{tp} + 0.25 \cdot (n_{fn} + n_u) + n_{fp}} , \tag{3.8}$$

with n_{tp} denoting the number of true positives, n_{fn} the number of false negatives, and n_{fp} the number of false positives. As before, n_u is the number of unanswered problems. We call this modified $F_{0.5}$ measure $F_{0.5u}$.

In order to make use of either measure, a verifier needs to inhibit a tendency to have higher confidence in correct answers than in incorrect answers (which is measured by the area under the ROC curve). Our experiments show that our new unmasking variant has this property and can classify *same-author* cases with very high or even perfect precision, depending on the chosen confidence threshold for deciding whether to answer a case.

3.3.3 Unmasking Verifier Evaluation

As features we used the chunk frequencies of the 250 most common words, determined chunk classification accuracy using 10-fold cross validation, and removed ten features per iteration. In total, we sampled 30 chunks per text with 700 words each. The final curves were generated as an aggregate of ten individual experiments and then used to train a linear SVM model on the curve points and their gradients.

When applying the model to the test set, we set the classifier to only give answers for cases which it can decide with a pre-determined confidence threshold. Thanks to the linear SVM kernel, we can obtain the confidence values directly from a curve's distance to the hyperplane. Table 3.1 lists the results of this unmasking approach on our authorship test corpus at various confidence thresholds in detail. A confidence threshold of 0 means that all cases are classified, resulting in c@1 being equal to the standard accuracy score and $F_{0.5u}$ being equal to $F_{0.5}$. We can see that, although very similar at low confidence thresholds, $F_{0.5u}$ yields much higher scores at high thresholds than c@1 does. The reason for this discrepancy is an imbalance in how well the classifier can assign classes. While the precision for *same-author* cases is very

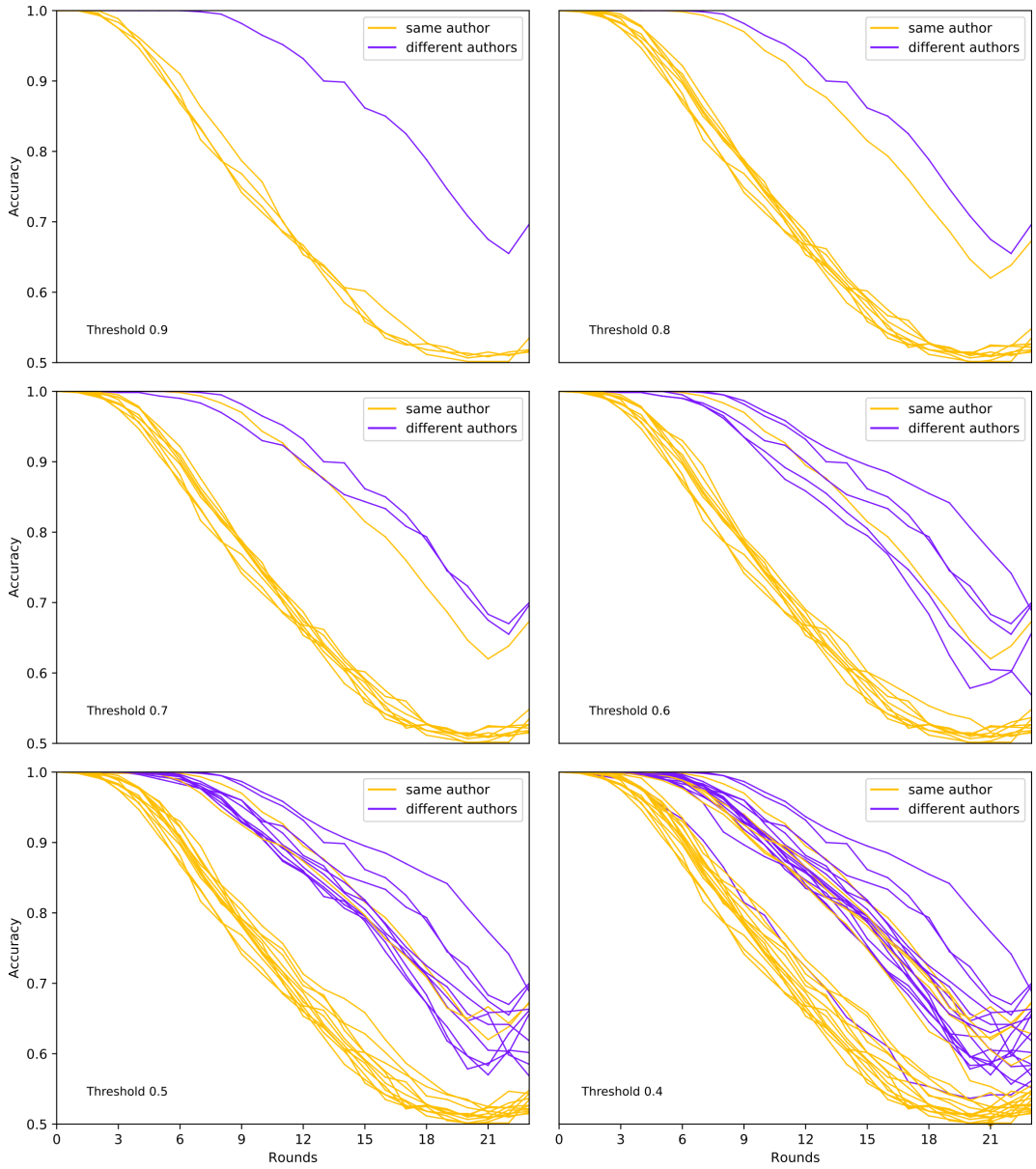

Figure 3.2: Classifiable unmasking curves in our test corpus at high confidence thresholds from 0.9 (upper left) to 0.4 (lower right). Colours represent the true class of a case. The middle of the gap approximately corresponds to the classification hyperplane. Precision for *same-author* cases stays very high for a long time, while *different-authors* are indistinguishable from individual *same-author* cases very early.

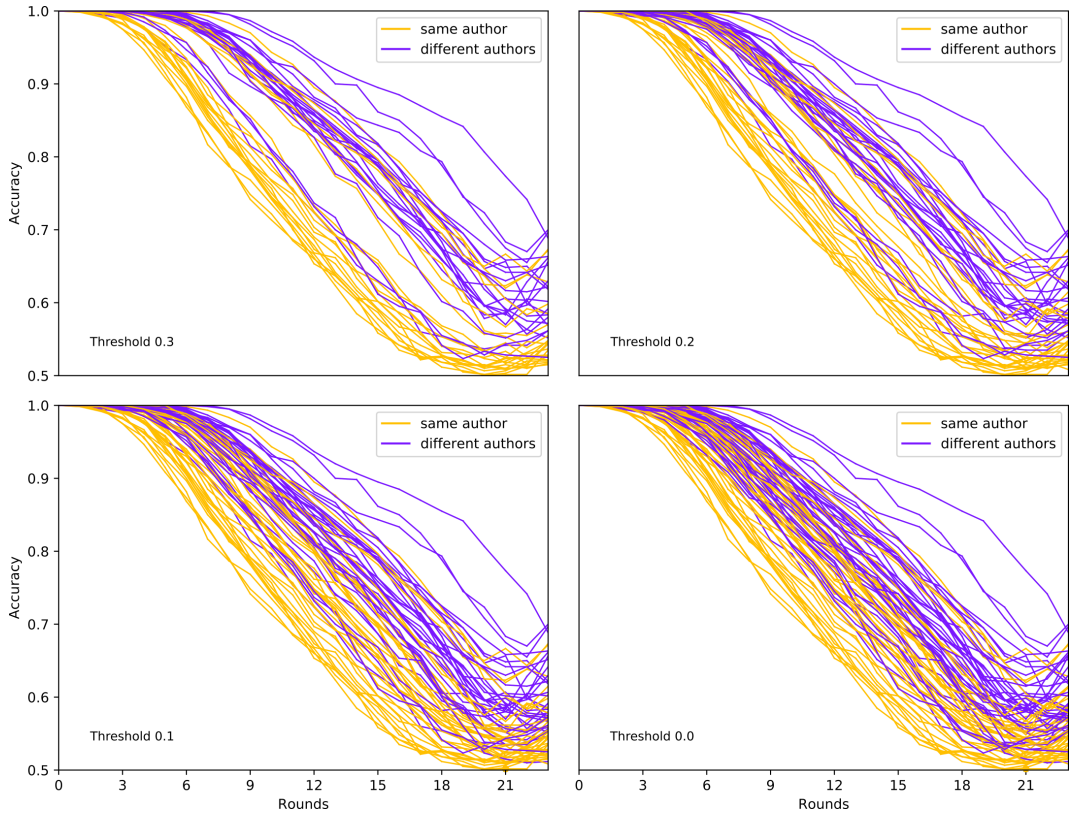

Figure 3.3: Classifiable unmasking curves at low confidence thresholds from 0.3 (upper left) to 0.0 (lower right). With a zero threshold, about 70 % of the cases can still be classified correctly.

high, the opposite is not true. Most *different-authors* cases are indistinguishable from *same-author* cases already at high thresholds (as can be seen in Figure 3.2), making this unmasking variant not a reliable method for determining if texts were written by different authors. Confidence levels have been assigned as a rough estimate of how sure a user can be that two classified texts were actually written by the same author. According to our experiments, thresholds of 0.7 and above reliably give a precision of 1.0, while thresholds of 0.5 and higher still provide a reasonably high level of confidence. Lower thresholds become increasingly less reliable. The actual precision of decisions at the same confidence value may vary slightly due to random noise, but the proportion of precision to the number of classified cases is very stable.

Table 3.1: Unmasking classification results of text pairs on our authorship test corpus at various confidence thresholds.

Confidence Level	Threshold	Precision	c@1	$F_{0.5u}$	% Classified
Very High	0.9	1.000	0.121	0.211	6.2
	0.8	1.000	0.211	0.360	12.5
	0.7	1.000	0.233	0.364	13.8
High	0.6	1.000	0.317	0.405	18.8
	0.5	1.000	0.468	0.508	30.0
Moderate	0.4	0.933	0.566	0.565	43.8
	0.3	0.833	0.616	0.577	55.0
	0.2	0.826	0.683	0.655	70.0
Low	0.1	0.821	0.717	0.723	87.5
	0.0	0.758	0.700	0.723	100.0

Chapter 4

Authorship Obfuscation

The goal of *authorship obfuscation* is to modify a text so that its contents remain intact, but all stylistic traces that point back to its original author are removed. The definition is deliberately specific to style in order to single out the much harder discipline which we call *authorship anonymization*, for which also the contents are redacted, which may as well change the semantics of a text or produce deliberate gaps in the form of blacked words or paragraphs.

Definition 4.0.1. (Authorship Obfuscation) *The transformation of an author's writing in order to impede automated or manual stylistic authorship analysis is called* authorship obfuscation.

In practice, this means fooling an automated authorship verification or attribution system as well as manual linguistic analysis. We can specify authorship obfuscation further:

Definition 4.0.2. (Authorship Masking) *Authorship obfuscation with the goal to prevent authorship detection is called* authorship masking.

Both definitions may sound similar, but the important difference is that the definition of authorship masking describes a more specific goal. This is to distinguish it from other forms of obfuscation like authorship *imitation*. For imitation, the original style is not necessarily masked, but modified in a specific way so as to resemble the style of another author. In this thesis, we describe our techniques for authorship masking, but most of the presented approaches are quite general and able to perform other kinds of obfuscation with some modification to the procedure. For that reason, we will continue to use the more general term *authorship obfuscation*.

4.1 Obfuscation Quality Assessment

Obfuscation itself is not an operation. It is a possible outcome of modifications of a text. Since these modifications can be of arbitrary kind and effectiveness towards an obfuscation goal, there are also various ways in which we can measuring obfuscation quality.

The most trivial obfuscation approach is deletion of the whole text. It will be impossible for any verification scheme to determine authorship of an empty text, but on the other hand, full deletion is not a desirable text transformation considering that the point of obfuscation is to keep the contents of a text intact. We therefore need a universally applicable framework for evaluating obfuscation schemes with regard to how well they achieve the obfuscation goal. Potthast et al. [42] propose the three evaluation dimensions safety, soundness and sensibleness (see: Chapter 2). Let us recall what these dimensions were about: Safety determines robustness against verification and de-obfuscation, soundness measures textual entailment between original and obfuscation and sensibleness quantifies grammaticality and inconspicuousness of the obfuscated text. The three dimensions were designed to be orthogonal, but we (the author) doubt that it is possible to generally distinguish between safety and sensibleness. Instead, we argue that safety against de-obfuscation and inconspicuousness can be directly related to each other.

While it is true and a well-known principle from cryptography that the security of a system must never rely on the secrecy of an algorithm, this is not the only factor to consider. Statistical dependence within a cipher text or between a cipher text and the plain text directly influences the security of the whole scheme even if the algorithm itself is otherwise secure. A good example to demonstrate this fact is one-time pad (OTP) encryption. For encrypting a message with a one-time pad, the secret plain text is modularly added onto a random stream of equal length. This encryption scheme is provably secure as long as the random stream is truly random, kept secret, and never reused—not even in parts. If created properly, the resulting cipher text gives no information about the plain text and any other string of the same length is equally likely to be the source plain text. On the other hand, if one can find statistical anomalies in the cipher text (e.g. because another natural-language text was used as the key instead of a random stream), it is quite easy to break the encryption by making simple assumptions about character distributions in the plain text or key. More generally, a cryptographic algorithm whose resulting cipher text leaks information about the original plain text must be considered weak. This also holds for one-way hash and trapdoor functions instead of only bidirectional ciphers (which imply that we can uniquely decrypt a cipher text back into its original plain text again).

To translate the example back to the obfuscation problem, there may exist an obfuscation scheme that is perfectly secure (safe) if and only if an attacker cannot determine which parts of the text were modified and how, i.e. if *any* other obfuscation (which includes no obfuscation at all) is equally likely. We even argue that showing no statistical abnormality (implying inconspicuousness) is a predominant safety requirement for minimally invasive obfuscation. A paraphrase that bears obvious (not further specified) properties of obfuscation, but does not allow attribution to any specific author can be imagined, but we are generally far from being able to create such paraphrases automatically. We therefore try to obfuscate with minimal invasiveness, but what works against us here is that texts in a natural language are very predictable and any kind of statistical anomaly is relatively easy to detect and revert as shown to some extent by Thi et al. [50]. A de-obfuscation attack gets even easier if anomalies in the text are not just purely statistical (which may very well elude a human reader, but not a thorough statistical analysis), but also detectable by simple spell or grammar checks. Following this rationale, we want to reduce the three dimensions to only two, namely *safety* and *soundness*, and make a slight distinction between *first-order safety* (safe against a-priori authorship detection) and *second-order safety* (safe against heuristic de-obfuscation attacks). A totally safe obfuscation must fulfil both requirements, but first-order safety may already be enough in certain scenarios to evade more thorough authorship analysis in the first place. The two remaining dimensions are truly orthogonal as can be shown by trivial example: a totally safe but unsound obfuscation be replacement of the text with an unrelated text by another author or a random string—an unsafe yet totally sound obfuscation be no obfuscation at all.

This adjusted definition of safety comprises most of what sensibleness was meant for, but one aspect which has not been covered and remains to be addressed is grammaticality of an obfuscated text (which was probably one of the main reasons for proposing sensibleness). We argue that poor grammaticality is not an inherent issue of obfuscation. It tangents soundness at the point where a text becomes incomprehensible and safety at the point where grammar and spelling mistakes allow to pinpoint obfuscative modifications. Besides these cases, grammaticality has no direct relation to our obfuscation goal and since we cannot establish orthogonality here, it does not qualify as a separate dimension. Instead, it is a general problem of natural language generation and must be assessed separately. In this sense, we do acknowledge sensibleness as an intuitive concept to quickly judge text quality, but do not see it as a fully qualified evaluation dimension. It is also worth noting that an original text may already be ungrammatical, in which case it would not make sense to judge an automated obfuscation by it, unless is makes the issue considerably worse so that it points to obvious machine-assisted tampering.

4.2 Distributional Authorship Obfuscation

We sketched the basic obfuscation workflow against KLD-based authorship models in our preceding work [4]. The procedure aims to increase the relative entropy between pairs of texts by increasing the KLD sum contribution of the most influential n-grams.

To find the n-gram with the highest influence, we rank all n-grams by their partial KLD derivative with respect to their probabilities in the to-be-obfuscated text. For every n-gram i with probabilities $p_i = P[i]$ and $q_i = Q[i]$ (lowercase notation chosen for better readability), its partial derivative with respect to q_i is:

$$\frac{\partial}{\partial\, q_i}\left(p_i \log_2 \frac{p_i}{q_i}\right) = -\frac{p_i}{q_i \ln 2}\ . \tag{4.1}$$

After dropping the constant $\ln 2$ from the denominator and the negative sign in front of the fraction, we end up with the very simple ranking function

$$\mathrm{R}_{\mathrm{KL}}(i) = \frac{P[i]}{Q[i]}\ . \tag{4.2}$$

Given that we can only change probabilities in Q (the to-be-obfuscated text), the rank of i approaches its maximum with $Q[i]$ going to 0. As the most simple, effective, and scalable method for achieving this effect, we established the *obfuscation by reduction* strategy, for which we iteratively reduce the frequency $freq_i$ of the highest-ranked n-gram i with $freq_i > 1$ to $freq_i - 1$. As a proof of concept, this was done by simply choosing one occurrence in the text and deleting it. After each deletion, ranks were recalculated and a new n-gram was chosen based on the updated ranks. By applying this obfuscation strategy, we were able to increase the KLD of *same-author* cases significantly. The effect carries over to JSD measures, since its derivative, although a little more involved, shows the same characteristics. Deleting the highest-ranked n-grams up to a mass of 2–4 % of a text was enough to fool our own JSD-based authorship verifier and cause a substantial drop in accuracy of PAN 2015's winning submission by Bagnall [2].

4.2.1 Effect on Unmasking

Until now, we have only observed the effect of the above-illustrated obfuscation strategy on our own model and Bagnall's PAN submission. To gather further evidence of its effectiveness, we now also test it against our new unmasking scheme. We have seen before that *same-author* cases can be detected with

Table 4.1: Unmasking meta classification results after obfuscation by +0.06 JSD. Not a single case was predicted as *same-author* down to a confidence threshold of 0.4.

Confidence Level	Threshold	Precision	c@1	$F_{0.5u}$	% Classified
Very High	0.9	0.000	0.025	0.000	2.5
	0.8	0.000	0.024	0.000	5.0
	0.7	0.000	0.096	0.000	8.7
High	0.6	0.000	0.160	0.000	17.5
	0.5	0.000	0.216	0.000	27.5
Medium	0.4	0.000	0.315	0.000	42.5
	0.3	0.667	0.410	0.143	66.7
	0.2	0.500	0.455	0.197	70.0
Low	0.1	0.417	0.460	0.266	85.0
	0.0	0.526	0.500	0.427	100.0

very high precision as their curves drop faster than the curves of *different-authors* cases. If our obfuscation approach works, *same-author* curves should decline as slowly as *different-authors* curves, making them indistinguishable. We obfuscated the *same-author* cases by deleting high-impact n-grams in one text until we reached a JSD increase of +0.06 compared to the original JSD between the two texts (which is enough to obfuscate even the most similar cases sufficiently). The number of n-gram deletions required to achieve this is on average 550–600. With text lengths of 23,000 characters on average, this corresponds to deleting about 6–8 % of a text. Our previous empirical rule of thumb of 2–4 % text deletion was sufficient to make texts indistinguishable by their KLD. For unmasking, however, we realised that we need to do a little more to make the most similar cases undecidable as well, although cases which are already farther apart from each other, could be obfuscated with less effort (more on this in Section 4.2.3 and Chapter 5).

Table 4.1 shows the classification results of the pre-trained meta model after obfuscation. Not a single case was predicted as *same-author* down to a confidence threshold of 0.4 as indicated by a precision and $F_{0.5u}$ score of 0.0. Compared to before (cf. Table 3.1), the total number of classifiable cases remains roughly the same at low thresholds, but has become worse at high thresholds as the formerly most similar cases were shifted into the uncertainty range. If we set the obfuscation target to +0.03 JSD (which requires about half the number of n-gram deletions, corresponding to about 3–4 % of a text), we can correctly classify a few remaining *same-author* cases at a confidence threshold of 0.6 or lower. In Figure 4.1, we see the unmasking curves of all cases in our test corpus

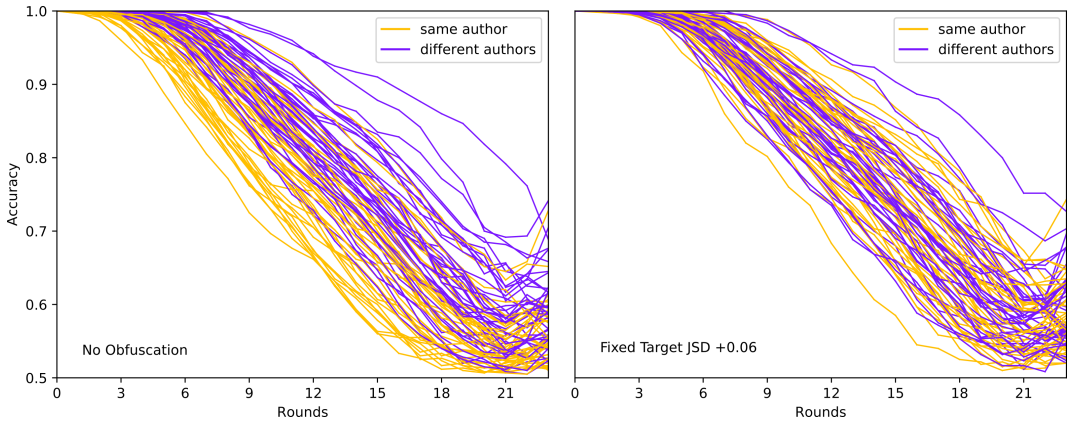

Figure 4.1: Unmasking curves of our test corpus before obfuscation and after *same-author* pairs were obfuscated by increasing their JSD by +0.06. Differences between *different-authors* curves are due to randomness in our unmasking implementation.

before and after obfuscation. The differences are striking. It is easy to see that distinguishing between curves of the two classes has become impossible as a result of how much they intersect.

4.2.2 Effect on Compression Models

Beyond unmasking, compression-based models appear to be a popular choice for text categorization and authorship verification, especially if the texts are small. Khmelev and Teahan [29] were one of the first to suggest this approach, but we will focus on the more recent implementations by Halvani et al. [19] to demonstrate obfuscation effects on another state-of-the-art authorship verification technique. Halvani et al. use off-the-shelf compressors based on prediction by partial matching (PPM) and various similarity measures to detect authorship. Based on their research, they name compression-based cosine (CBC or CosS as it is abbreviated in the original paper) and the Chen-Li metric (CLM) as the most successful measures. Both measures were originally proposed by Sculley and Brodley [44]. For two texts x, y, their concatenation xy, and a function C returning the compressed length of its input, the compression-based cosine is defined as:

$$\mathrm{CBC}(x, y) = 1 - \frac{C(x) + C(y) - C(xy)}{\sqrt{C(x)C(y)}} \tag{4.3}$$

and the Chen-Li metric is defined as:

$$\mathrm{CLM}(x, y) = 1 - \frac{C(x) - C(x|y)}{C(xy)} \ . \tag{4.4}$$

The conditional compression length $C(x|y)$ is estimated as $C(xy) - C(y)$.

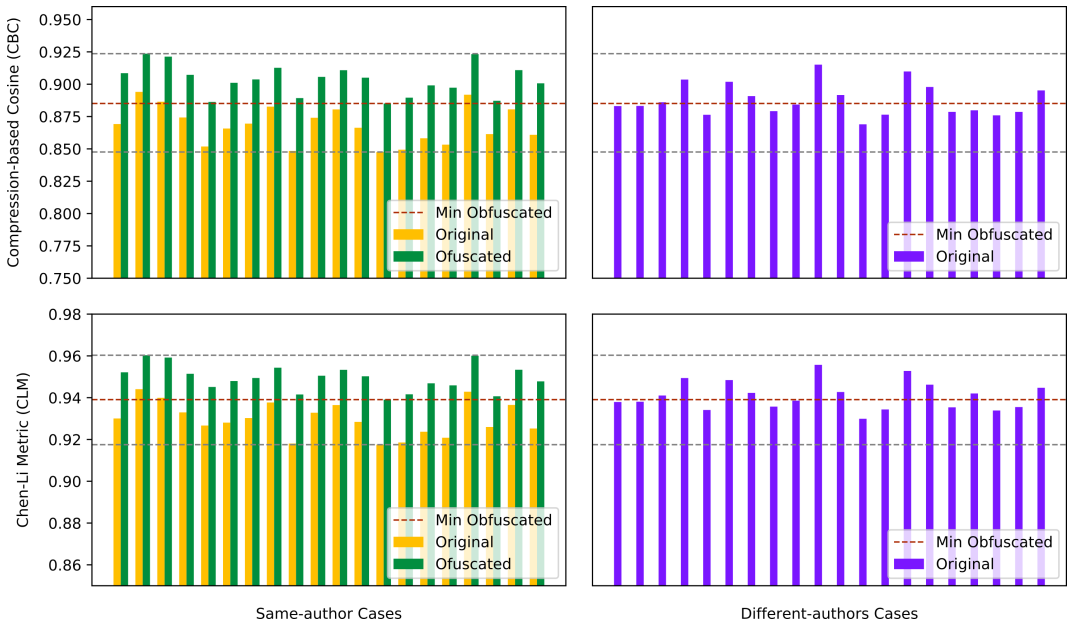

Figure 4.2: Obfuscation effect on compression models. The top row shows the compression-based cosine (CBC), the bottom row the Chen-Li metric. To the left are original *same-author* cases (yellow) and their obfuscation (green). To the right are real *different-authors* cases for comparison. A JSD increase of +0.06 increases both measures for *same-author* cases beyond those of *different-authors* cases.

Using these measures and PPMd as compression algorithm (a variant of PPM which accounts for unseen symbols), Halvani et al. yield competitive results on the 2013–2015 PAN corpora (with the exception of the 2014 essays corpus) and similar numbers on other self-created corpora. We show obfuscation results against a reimplementation of this technique on the subset of the first 20 *same-author* pairs from our training corpus and the first 20 *different-authors* pairs for comparison in Figure 4.2. The same n-grams were removed as in the previous section to achieve a JSD increase of +0.06 for each pair. While the original text pairs can be correctly distinguished from *different-authors* pairs in many cases by a learned threshold, CBC and CLM values for obfuscated texts grow beyond any classifiable threshold. This strong obfuscation effect is not unexpected. Sculley and Brodley describe their metrics in terms of Kolmogorov complexity, but Kaltchenko [27] showed that the normalized Kolmogorov complexity of a Markov information source 'almost surely' converges towards its entropy and that distance measures based on the Kolmogorov complexity may be estimated by relative entropy rate, i.e. the Kullback-Leibler divergence. The reason why natural language generally allows for very good compression ratios is that it is predictable (printed English has an entropy of at most 1.75 bits per character [8])

and simple finite-order Markov language models are quite effective in predicting the next characters in a sentence. Since PPMd uses a character-based Markov model for compression, an increase of relative entropy is expected to have an effect on the compression ratios.

4.2.3 Safety of n-Gram Selection Strategies

We have seen before that unmasking does respond as expected to our obfuscation approach, though we must not forget that we are using a deterministic n-gram selection strategy in ranking them by their KLD gradient. This may result in predictable obfuscation patterns and therefore may have severe obfuscation safety implications. Also when we obfuscate a text with the goal to increase its overall JSD, our strategy may achieve this by only modifying parts of the text and leaving other areas untouched. Verification schemes may choose to ignore the changed parts and only focus on the unchanged 'niche' subset of a text pair. Unmasking does not consider the whole text but only the top k most frequent words (we use $k = 250$); thus, it makes sense to examine the impact of a global obfuscation strategy on only the subset of the k most frequent words.

To assess the obfuscative effect on the top k most frequent words of a text pair, we obfuscated the *same-author* pairs in our training corpus utilizing multiple alternative n-gram rankings functions:

- **Normal ranking:** This is our previously established ranking function $\mathrm{R_{KL}}$, which ranks n-grams according to their KLD gradient. It is the most efficient ranking for increasing the JSD in terms of needed operations.
- **Inverse ranking:** Inverse gradient ranking, i.e. $-\mathrm{R_{KL}}$. This should be the least efficient ranking and we need to compensate by performing more text operations.
- **Positive inverse ranking:** This strategy is the same as inverse ranking, but only considers n-grams with $\mathrm{R_{KL}} > 1$. This is to avoid making texts more similar at first by reducing the frequencies of n-grams that are common in the to-be-obfuscated text, but rare in the other text.
- **Random ranking:** This is our baseline ranking. Since it is totally unpredictable, we consider it safe yet not necessarily efficient.

Figure 4.3 shows the JS_Δ distribution of the most frequent word tokens within *same-author* cases in our training corpus before and after obfuscation with the aforementioned ranking approaches. All cases were obfuscated until they showed a global JSD increase of $+0.06$. Most selection strategies perform

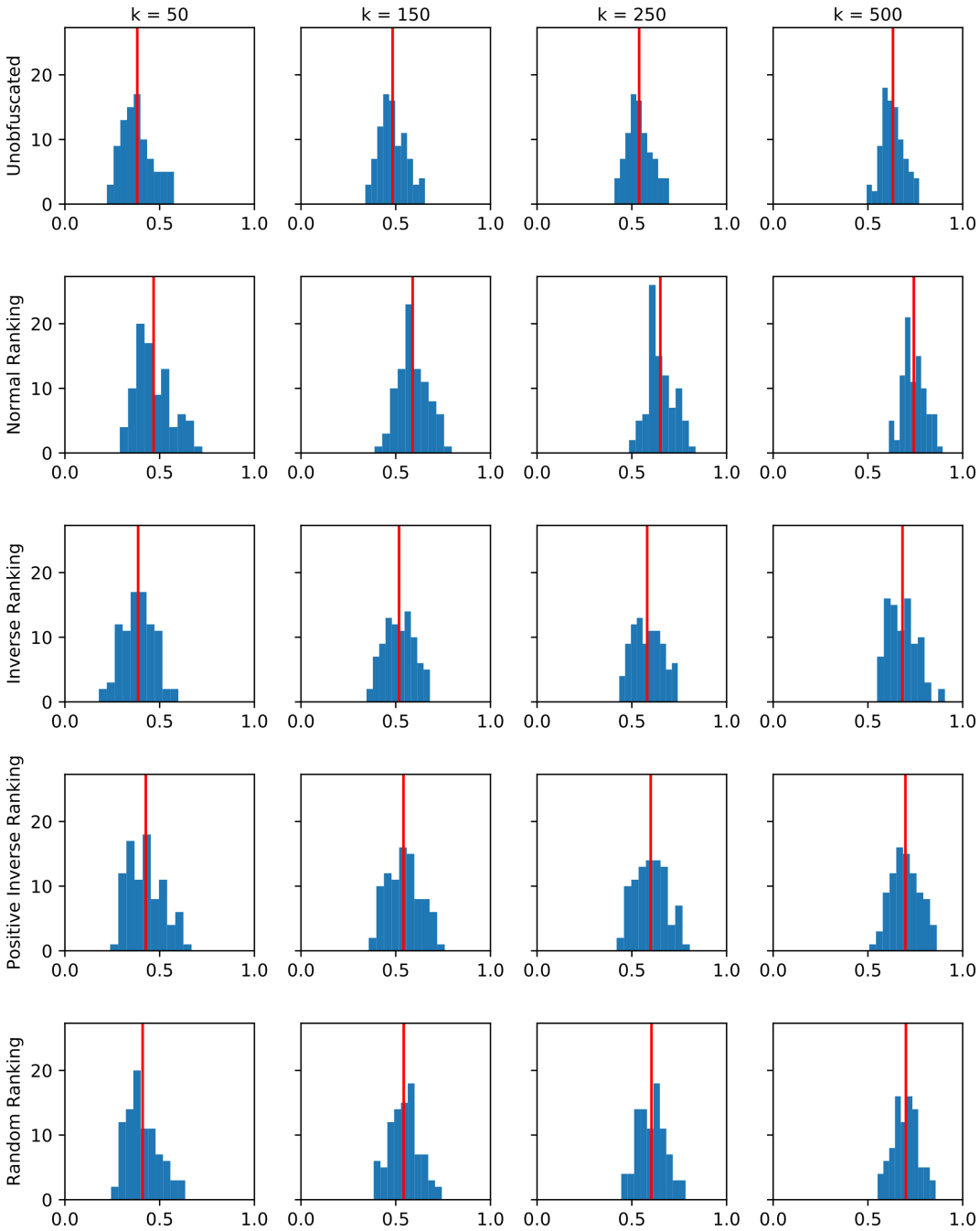

Figure 4.3: Jensen-Shannon distances of the k most frequent words within *same-author* pairs before and after obfuscation with different n-gram selection strategies. We can see that positive inverse and random ranking perform reasonably well (measured by how normal ranking performs), but inverse ranking obfuscates significantly worse (or not at all) for low k. The red line denotes the distribution mean.

Table 4.2: Results of one-sided paired t tests between JS_Δ values of the k most frequent words for unobfuscated vs. obfuscated cases and for obfuscation with inverse vs. normal ranking. Differences in the data are highly significant ($p \ll 0.001$) with very large effect sizes ($d > 1.2$), except for unobfuscated vs. inverse at $k = 50$. The p values are generally much higher and effect sizes lower for inverse ranking. Normal ranking performs best for all listed k.

		Unobfuscated vs.				Inverse vs.
k	Statistic	Normal	Inverse	Pos. Inverse	Random	Normal
50	$p =$	4×10^{-30}	0.202	1×10^{-20}	7×10^{-19}	3×10^{-19}
	$t =$	17.139	0.838	12.052	11.170	11.382
	$d =$	1.817	0.089	1.277	1.184	1.206
150	$p =$	2×10^{-46}	1×10^{-9}	6×10^{-24}	8×10^{-29}	6×10^{-19}
	$t =$	28.415	6.648	13.759	16.421	11.212
	$d =$	3.012	0.705	1.458	1.741	1.188
250	$p =$	5×10^{-52}	2×10^{-12}	6×10^{-25}	5×10^{-30}	3×10^{-19}
	$t =$	33.397	8.089	14.304	17.095	11.382
	$d =$	3.540	0.857	1.516	1.812	1.206
500	$p =$	1×10^{-58}	4×10^{-14}	5×10^{-26}	5×10^{-32}	1×10^{-16}
	$t =$	40.069	8.886	14.877	18.317	9.824
	$d =$	4.247	0.942	1.577	1.942	1.041

similarly on the top k subset, except inverse ranking, which does considerably worse. One-sided paired t tests (Table 4.2) support our observations. All strategies show a highly significant JS_Δ increase ($p \ll 0.001$) with huge effect sizes ($d \gg 0.8$). The exception is inverse ranking, which shows no significant effect at $k = 50$ and performs generally much worse than any other ranking approach. Inverse ranking has higher p values by several magnitudes and much lower effect sizes. As shown in the last column, normal ranking performs significantly better than inverse ranking with strong effect sizes at all k, but also seems to be the most effective ranking overall. With larger k, all rankings obviously converge. As a more intuitive visualization of the pairwise effects, Figure 4.4 shows the relative JS_Δ increase within *same-author* cases for the k most frequent words. Here we can see quite clearly that all strategies produce similar results at higher k, but differ at low k. For individual pairs, inverse ranking even shows a JS_Δ decrease, which is most extreme at $k = 50$.

The results make sense when we think about the way we affect a text's n-gram distribution. The majority of the k most frequent words are words which tend to appear with similarly high (relative) frequencies in both texts. A

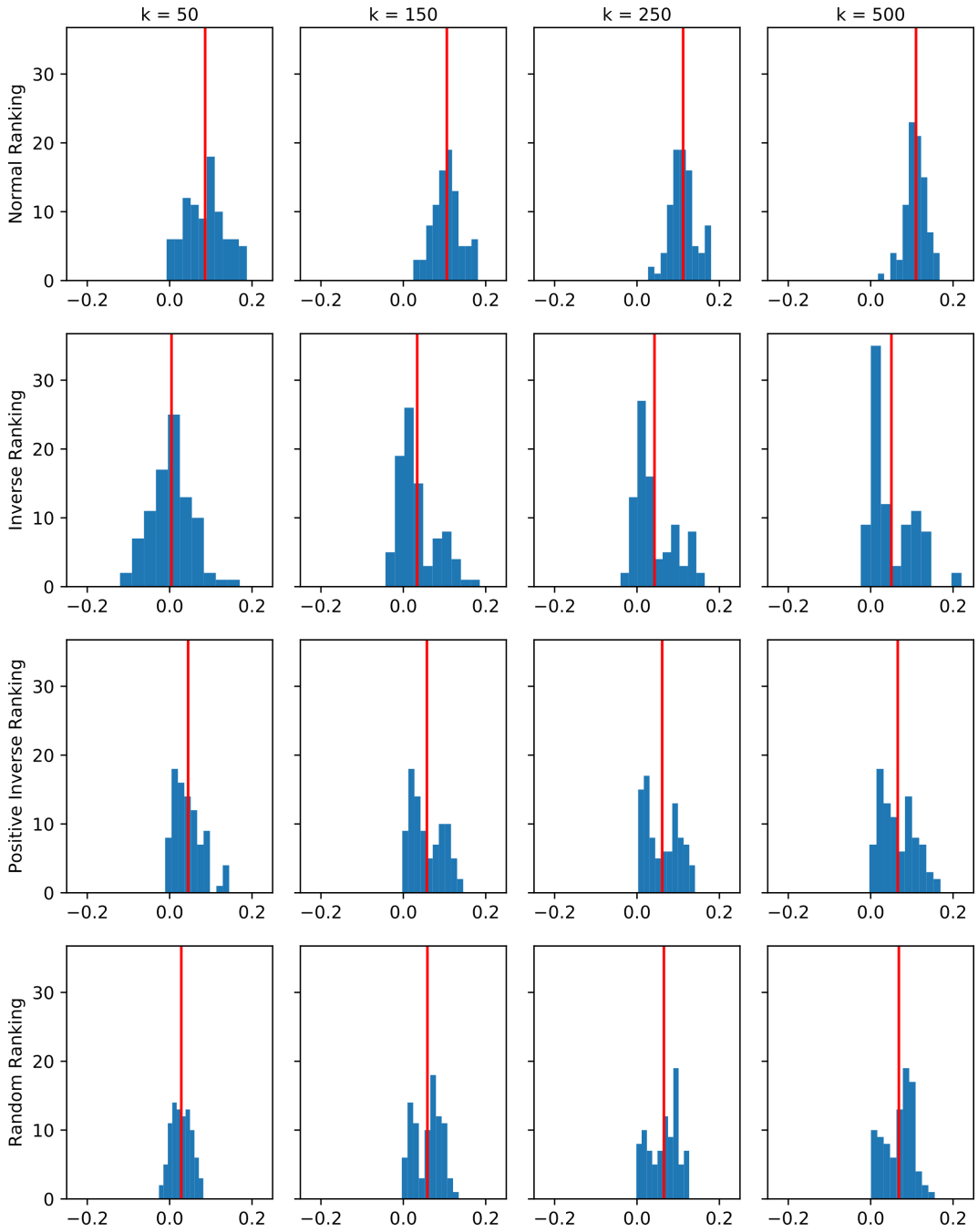

Figure 4.4: Relative differences of the Jensen-Shannon distances between the obfuscated and unobfuscated *same-author* pairs using various n-gram selection strategies. Most strategies perform similarly with the exception of plain inverse ranking, which shows by far the worst performance and even causes a JS_Δ decrease for some cases (which are negative on the x axis, most notably at small k). Normal ranking shows the largest positive mean difference.

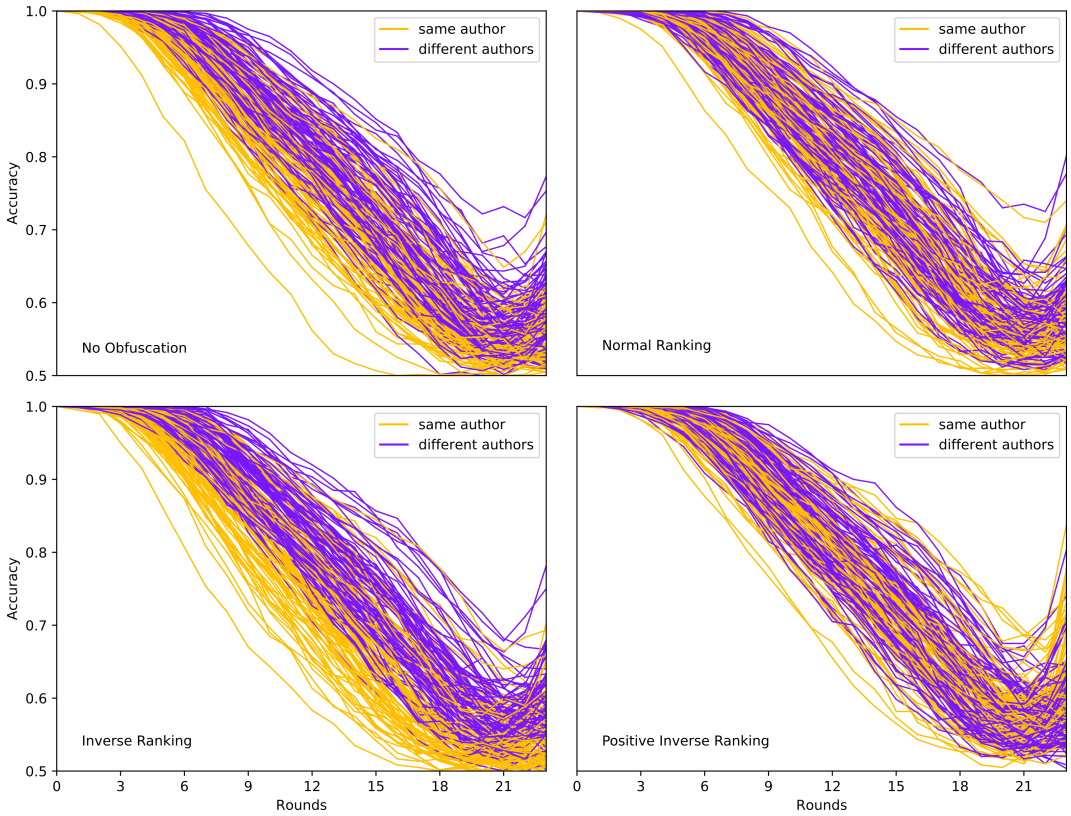

Figure 4.5: Unmasking curves on our training corpus before obfuscation and after obfuscation up to a JSD increase of +0.03 with normal, inverse, and positive inverse ranking. Differences between *different-authors* curves are again due to randomness in our unmasking implementation.

few words may also be extremely frequent in one text and less so in the other, but more often, low frequencies in one text will amortize high frequencies in the other one. It makes sense to assume that word frequencies are correlated with n-gram frequencies (to a certain extent), so n-grams occurring in the top k words will probably also have similar frequencies in both texts. If we sort all n-grams of the distribution by $\mathrm{R_{KL}}$, we call $\mathrm{R_{KL}} = 1$ the midpoint (where all n-grams are located whose frequencies in both texts are identical), $\mathrm{R_{KL}} \ll 1$ the negative, and $\mathrm{R_{KL}} \gg 1$ the positive end. Since many of the top k words will be around the midpoint, a ranking which prefers n-grams around it (which positive inverse ranking does) will perform quite well. But since all these n-grams will have very little effect, we can expect it not to perform best. The best-performing ranking—normal ranking—will directly influence only very few common n-grams, but with much higher effect, resulting in overall better performance. The n-grams selected by normal ranking are rare, so we cannot

reuse the same n-grams very often and have to move towards the midpoint more quickly, which increases the chances to affect another top word slightly. The least effective and at low k even counterproductive ranking is clearly plain inverse ranking. The reason why this performs worse is that it starts at the negative end and reducing frequencies of these n-grams will make texts more similar at first, which overall *reduces* the JSD. Only then will the selection strategy cross the midpoint and slowly move towards the positive end. But as with positive inverse ranking, selected n-grams from the positive half will initially have only very little impact. After a while, a lot of n-grams will have been modified, many of them occurring in the top k words, but most of them with negative effect. Even if we are not concerned about the top k words, it is obvious that inverse ranking is less than ideal under all circumstances and can be seen as an unlikely worst case for random ranking. Random ranking will select n-grams from all different parts of the distribution equally. At low k, we may also see negative effects, but with higher k, we will almost certainly see a positive effect as a result of the JSD's gradient properties: an n-gram at the far positive end of the distribution will always outweigh an n-gram at the far negative end, so uniformly sampled n-grams will on average affect the JSD positively.

In Figure 4.5, we show unmasking curves for normal, inverse, and positive inverse ranking. The amount to which *same-author* curves are mixed with *different-authors* curves corresponds to what we see in the word token JSD plots. Random ranking, although not shown here, performs similarly to normal and positive inverse ranking in terms of obfuscation. In terms of operations needed, normal ranking is clearly the most efficient method with 590 removals on average, followed closely by positive inverse ranking with 770 removals. All other selection strategies require significantly more operations to achieve these results. Inverse ranking requires by far the most of all strategies with 1,900 removals, but fails to achieve any obfuscation at all and must therefore be considered unsafe. In order to increase safety, it is also entirely possible to combine approaches. The efficiency gap between normal and positive inverse ranking is small enough that one might consider drawing (semi-)randomly from the positive half of the n-gram distribution to harden an obfuscation against reversal.

4.2.4 Estimating Text Operation Side Effects

Text operations always produce side effects. As a side effect we define the newly introduced n-grams at the boundaries of an intended operation. Our simple proof-of-concept obfuscation approach for which we simply delete n-grams, suffers particularly from such side effects. For every n-gram that we

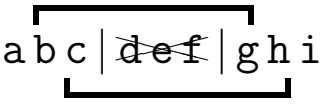

Figure 4.6: Schematic visualization of removing the character trigram `def`. `bcg` and `cgh` are the two new side-effect trigrams.

delete, we introduce two new side-effect n-grams. For example, if we delete the trigram `def` from the string `abcdefghi`, we introduce the new side-effect trigrams `bcg` and `cgh`. Besides the obvious lack of second-order safety for such simple deletions (see: Section 4.1), it is important to assess the influence of side effects on our obfuscation to validate the usefulness of our technique. During our previous research [4], we compared actual text obfuscation to *ideal* side effect–free obfuscation on hypothetical n-gram vectors and could show empirically that side effects only marginally affect obfuscation performance. Yet we have not really quantified the influence of side effects on obfuscation. As deletions of n-grams tend to produce syntactically unusual n-grams which are unlikely to appear anywhere else, we first want to examine the contribution of a 'maximally rare' n-gram, i.e. an n-gram which we introduced for the first time and which did not exist before in either text.

First, let us recall the definition of the Jensen-Shannon divergence:

$$\mathrm{JSD}(P\|Q) = \frac{1}{2}\cdot\mathrm{KLD}(P\|\tfrac{P+Q}{2}) + \frac{1}{2}\cdot\mathrm{KLD}(Q\|\tfrac{P+Q}{2}) \ ,$$

and the Kullback-Leibler divergence:

$$\mathrm{KLD}(P\|Q) = \sum_i P[i]\log_2\frac{P[i]}{Q[i]} \ .$$

As mentioned, the side-effect n-gram is maximally rare, therefore its probability in P is 0 and we denote its probability in Q by $\zeta_Q = \frac{1}{N_Q}$ with N_Q being the normalizing constant of Q. For texts of identical lengths, it is probably safe to assume that $\zeta_Q \approx \zeta_P$, so we will only write ζ from now on. Because $P[i_s] = 0$, the first half of the JSD formula vanishes for i_s and the rest reads as:

$$\frac{1}{2}\,\zeta\log_2\frac{2\zeta}{\zeta} = \frac{1}{2}\,\zeta\log_2 2 = \frac{\zeta}{2} \ . \tag{4.5}$$

For very small sample sizes, this term will always dominate the sum, but we presume a sufficiently large number of samples (n-grams) here. To compare this (absolute) JSD contribution to the impact of a selected n-gram, we can

establish estimates for the needed proportion between the n-gram's probabilities $p_i = P[i]$ and $q_i = Q[i]$ using the JSD's partial derivative with respect to q_i:

$$\frac{\partial}{\partial\, q_i}\left(\frac{p_i}{2}\log_2\frac{2p_i}{p_i+q_i}+\frac{q_i}{2}\log_2\frac{2q_i}{p_i+q_i}\right)=\frac{1}{2}\log_2\frac{2q_i}{p_i+q_i}\ . \tag{4.6}$$

It is easy to see that for any n-gram i with $P[i] = 0$, this term will always be 0.5, which is what we obtain for a totally new side-effect n-gram i_s. As 0.5 is the highest positive slope an n-gram can have with $p_i, q_i \in [0, 1]$, we cannot find an n-gram whose addition would have a higher impact. Fortunately, we are pursuing a reduction strategy, so we need a negative slope. To outweigh side effects, an n-gram selected for reduction needs to have a JSD slope < -0.5. If we rearrange the gradient inequality for p_i and q_i

$$-0.5 > \frac{1+\log_2 q_i-\log_2\left(p_i+q_i\right)}{2}\ , \tag{4.7}$$

we obtain $0 < Q[i] < \frac{P[i]}{3}$ as relative frequency bounds for high-impact n-grams. By ranking n-grams according to the KLD slope ranking function R_{KL}, we will pick n-grams where $P[i] \gg Q[i] > \zeta$ first and so we can conclude that a newly introduced side-effect n-gram will most likely not outweigh a selected high-impact n-gram.

In practice, however, we will probably see a lot of side-effects which already exist in the texts, especially if we employ more advanced paraphrasing mechanisms that produce syntactically correct obfuscations. If a side-effect n-gram already exists in both texts, we may see an effect that works in the opposite direction to what we aim for with the reduction strategy (i.e. it makes the texts more similar). Depending on the proportion of $P[i_s]$ and $Q[i_s]$, the contribution to the JSD may be either positive or negative. We expect side-effect n-grams to be almost random, so their JSD contributions will on average be lower than for a selected n-gram and cancel each other out. Hence, we can expect these kinds of side effects to also not outweigh a selected n-gram. If anything, they will work against our efforts due to the largely negative JSD slope. A special case is a side-effect n-gram which is only new in the to-be-obfuscated text, but already exists often in the other one. Here we may once see a very large contribution if $P[i_s]$ is very large. This is equivalent to the *obfuscation by extension* strategy, which we described as an alternative (but less scalable) obfuscation strategy in our previous studies [4]. This obfuscation strategy increases the JSD by changing a maximally rare n-gram with a sum contribution of $\frac{\zeta}{2}$ into a 'normal' high-impact n-gram.

4.2.5 Empirical Side Effect Analysis

To verify the theoretical estimates of side effect contribution in practice, we also perform a quick empirical analysis here. We have already shown that inverse n-gram ranking requires far more text operations than normal ranking. This empirically proves that side effects cannot outweigh selected high-impact n-grams, otherwise we would not see this massive increase in text operations. We can assume side effects to be mostly random, so the selection strategy should not influence their impact on the JSD.

Recall that we needed about 300 n-gram deletions for a JSD increase of $+0.03$ and 600 for a JSD increase of $+0.06$ on our corpus, which amounts to approximately 4 % and 8 % of a text. Running the obfuscation algorithm with inverted n-gram ranking has shown a massive increase of required n-gram removals. To reach a global obfuscation target of $+0.03$ JSD, we had to perform on average 1,650 n-gram deletions for *same-author* pairs in our training corpus and for $+0.06$ JSD, we needed roughly 1,900 deletions (for some pairs even as many as 3,600). To visualize the amount of work required, we show a very short excerpt of *Winter Adventures of Three Boys in the Great Lone Land* by Egerton Ryerson Young. The original text is:

> With this question of the old Indian ringing in their ears the party in the kitchen broke up, and as the day had been a long one they all soon retired to rest. The boys were more than delighted with the day's experience, and were full of joyful anticipation for the morrow, for then it was that they were to select the dogs that were to constitute their own trains and at once to begin the work of breaking them in. So long and soundly did they sleep the next morning that the second breakfast bell was ringing when they awoke, and so they had but little time in which to dress ere breakfast was served. However, to their joy they found that others had also overslept themselves. [...]

After obfuscation up to a JSD target of $+0.06$ using normal n-gram ranking, we get the following (shorter) text:

> With this question of the old iainging in their eathe parin the kitchen broke up, and as the day had been a long one they all soon retired to rest. The s were more than delighted with the day's experience, and were full of joyful anticipation for the morrow, for

> then it was that they were to select the dogs that were to constitute their own trad at once to begin the work of breaking them in. Song and soly did they sleep the next morning that the second breakfast bell was ringing when they awoke, and so they had butt time in which to ss erreakfast was served. However, to their joy they fo that othehad also ovlept themselv [...]

Differences between original and obfuscation are highlighted. While the modified parts are clearly identifiable and obscure the meaning at times, the text remains largely readable. Approximately half these changes are required for an obfuscation target of +0.03 JSD. Reversing the n-gram ranking gives very different results. With (in this case) 3,340 required text operations, the final text is almost entirely unreadable:

> Withitiof the old Indian rinn their ears the partn the kitchen ke and the day had been they all n ret. boys werore thigd withe da and were full of joyfpation for w, for then ias that they were tct the re to constitute theiins an to begin the w of bg t. So g and soep the t ming that the sed ak was ging when they awoke, athey had ttleme ihich to dr ere brt was sed. Ho to their thnd thers had verslelves. [...]

Due to the many changes, we skipped highlighting the differences between the original and the obfuscated text. It is quite surprising that while the text is completely illegible, its global JSD increase is exactly the same and not in any way higher than the previous obfuscation which used normal ranking. Even more surprising is that by extracting only a few of the most common words, authorship verification remains possible on texts like this one. If we look more closely, we can actually see that remaining words are largely stop words, whereas almost all topic words have been obscured. This only emphasizes that the way we select n-grams not only has an immense effect on the efficiency but also the safety of an obfuscation algorithm.

Chapter 5

Adaptive Obfuscation

Until now, we have only used empirically obtained fixed JSD thresholds as termination criteria for our obfuscation algorithm. The thresholds were chosen so that either *most* or *all* text pairs in the corpus become undecidable. It is easy to see that these thresholds are only very rough estimates when dealing with short texts, such as the ones in our corpus. Both the JSD measure and unmasking show large variance between text pairs, so some pairs are extremely similar, while others are undecidable already and probably need no obfuscation at all. It is obvious from previous examples that with a fixed threshold, we overobfuscate cases which are already far enough apart from each other while only barely obfuscating the most similar cases. Instead of obfuscating a text up to a fixed and mostly arbitrary threshold, we want to find a more sensible adaptive threshold on a case-by-case basis.

5.1 Decidability and Obfuscation

As a prerequisite to finding an adaptive measure, we first need to make a distinction between *effective author distance* and *necessary obfuscation*. With effective author distance we mean the inherent level to which a text's writing style differs from the style of its original author and therefore how (un)decidable its authorship is. Author distance is independent of whether a text is an original or a modification and it can be measured by a suitable authorship metric. Necessary obfuscation is the amount to which we need to obfuscate a text in order to achieve a desired (higher) *effective author distance*. Both effective author distance and necessary obfuscation only take the authorship metric into consideration and are entirely independent of the actual amount of work required to achieve a particular target value.

Definition 5.1.1. (Effective Author Distance) *The distance between the writing style of a particular text and the true style of a particular author is called* effective author distance.

Definition 5.1.2. (Necessary Obfuscation) *The positive difference between a text's desired effective author distance after obfuscation and its original distance with regard to a reference style is called* necessary obfuscation.

Let effective author distance be denoted by ϕ and necessary obfuscation by δ. With a desired effective target author distance ϕ_t and given that the original effective author distance ϕ_0 is not already beyond the target distance, an authorship *masking* task can be modelled as:

$$\phi_t = \phi_0 + \delta, \quad \delta \geq 0 \ . \tag{5.1}$$

For an authorship *imitation* task, the reference style would be the target author's style and the task would have the following form:

$$\phi_t = \phi_0 - \delta, \quad \phi_0 \geq \delta \geq 0 \ . \tag{5.2}$$

An adaptive obfuscation approach will determine the necessary obfuscation beforehand based on the texts it is given and then only obfuscate until it has achieved the necessary effective target author distance. If a text's style is already sufficiently different (similar), δ can be zero and no obfuscation has to be applied at all.

5.2 Defining Adaptive Targets

An obvious adaptive approach would be to classify text pairs by any number of authorship verification systems and then iteratively obfuscate until none of the verification systems is able to determine authorship any more with confidence above a predetermined threshold θ. This approach, however, is very costly and depends heavily on the used verifiers. We want a much simpler, more quantifiable and reproducible measure of obfuscation strength.

Since the JSD (or KLD) is our basic authorship model and we use it to guide the obfuscation n-gram selection strategy, it makes sense to derive an obfuscation measure from it. A problem with a JSD-based measure that needs to be solved is that JSD values are not necessarily comparable across different text pairs. Long texts are generally more similar to each other than short texts. The shorter a text, the sparser and noisier its n-gram sampling becomes. For that reason, the JSD of long text pairs is generally lower than the JSD of short text pairs, even for texts by the same author. It is therefore of no use to

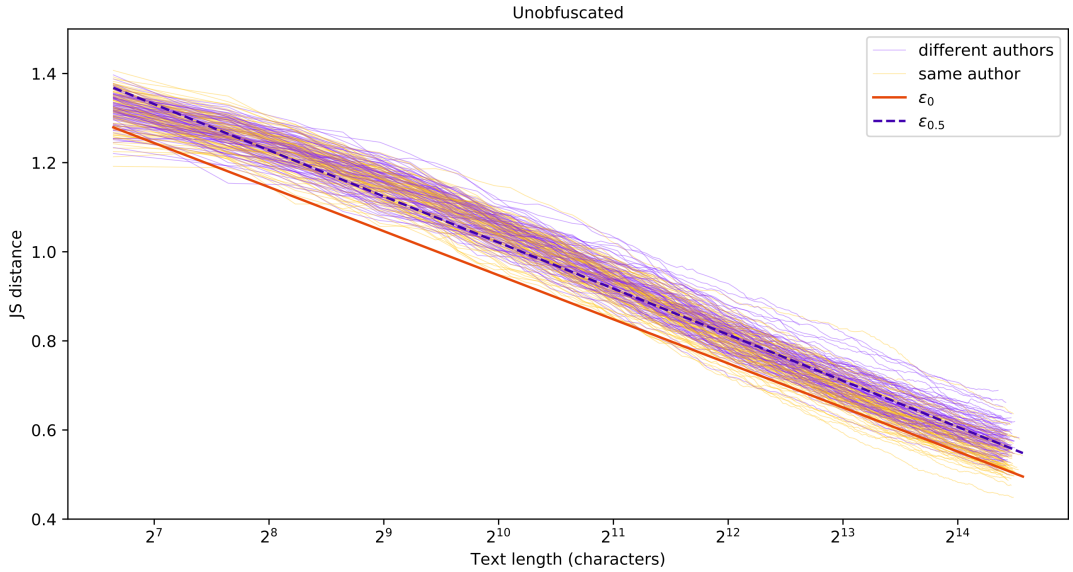

Figure 5.1: Development of the Jensen-Shannon distance on our training corpus with regard to text length. Longer text pairs (above 2,048 characters) are logarithmically more similar than short text pairs, but easier to distinguish due to distance convergence. The two lines ε_0 and $\varepsilon_{0.5}$ indicate the 0th and 50th percentiles of distances within *different-authors* pairs.

compare the JSD values of pairs of different text lengths, unless we normalize them. Because we want a real metric, it also makes sense to use the Jensen-Shannon distance (JS_Δ) instead of the plain JSD. Plotting the JS_Δ with regard to the minimum text length in a pair reveals an approximately logarithmic relationship in our corpus (Figure 5.1). Very short texts (less than 2,048 characters) are extremely noisy, but longer text pairs show a clear logarithmic decrease of their JS_Δ. The most interesting aspect of this relationship is the almost length-invariant spread of cases. Nonetheless, the cases tend to converge towards the upper or lower bounds of this spread depending on their class and thus separate clearly with growing length. We reproduced this plot on the PAN 2014 novels corpus [45] and found the same basic relationship with comparable case variance but stronger case clustering (by class) on the y axis due to the much lower number of unique authors in that corpus.

5.3 Obfuscation Levels by Percentiles

Assuming that the JS_Δ-to-text-length relationship is a general feature of natural texts, we can measure author distance in JS_Δ@L (Jensen-Shannon distance at length) and fit threshold lines to define obfuscation levels. On the y axis, we

Table 5.1: Overview of obfuscation levels by effective author distance thresholds and their log-scale polynomial fit coefficients on our training corpus.

Percentile Threshold	Obfuscation Level	Slope	Intercept
$< \varepsilon_0$	No Obfuscation	-	-
$\geq \varepsilon_0$	Moderate Obfuscation	-0.099	1.936
$\geq \varepsilon_{0.5}$	Strong Obfuscation	-0.103	2.056
$> \varepsilon_{0.99}$	Overobfuscation	-0.107	2.168

eliminated outliers beyond the $[30, 70]$ percentile range using Tukey's [51] interquartile method (IQR factor: 1.5) and then calculated a linear least-squares fit through the 0th percentile of *different-authors* JS_Δ values on the logarithmic scale. We call this line the ε_0 threshold and it will serve as our obfuscation baseline. We use the first text of a pair as an approximation of its author's style and define text pairs with the other text having an effective author distance below ε_0 as *unobfuscated*. Conversely, we define those pairs with an effective author distance of at least ε_0 as *moderately* obfuscated. The line we fit through the 50th percentile of *different-authors* distances is called the $\varepsilon_{0.5}$ threshold. We define text pairs with an effective author distance of at least $\varepsilon_{0.5}$ as *strongly* obfuscated. Pairs beyond the 99th percentile threshold, i.e. beyond $\varepsilon_{0.99}$, are *overobfuscated*.

Table 5.1 shows an overview of all obfuscation levels, their corresponding ε thresholds and the line fit coefficients on our training corpus. The gradients of higher ε thresholds are slightly steeper, so we can clearly see the convergence rate of *different-authors* cases in the numbers. We believe these coefficients are approximately applicable to other corpora of similar nature, but recommend relearning them on an appropriate training corpus for application to texts of very different nature. Generally, if applied to another corpus, increasing the threshold a little for improved safety may be advisable.

When we adjust our obfuscation strategy to use an adaptive threshold, only few cases need the full number of n-gram removals from before to be fully obfuscated and some cases need no or hardly any obfuscation. The JS_Δ spread of *same-authors* cases approximately resembles a normal distribution centred a little below $\varepsilon_{0.5}$, so we can reduce the amortized number of obfuscation operations by about half. Figure 5.2 shows this improvement quite clearly. Instead of shifting JS distances of all cases uniformly into undecidability, *same-author* pairs now gather around the $\varepsilon_{0.5}$ threshold. When we apply adaptive obfuscation to our test corpus and generate unmasking curves for the obfuscated cases, we obtain the results shown in Figure 5.3. The overall effect is slightly stronger than uniform obfuscation by $+0.03$ JSD, but weaker than obfuscation

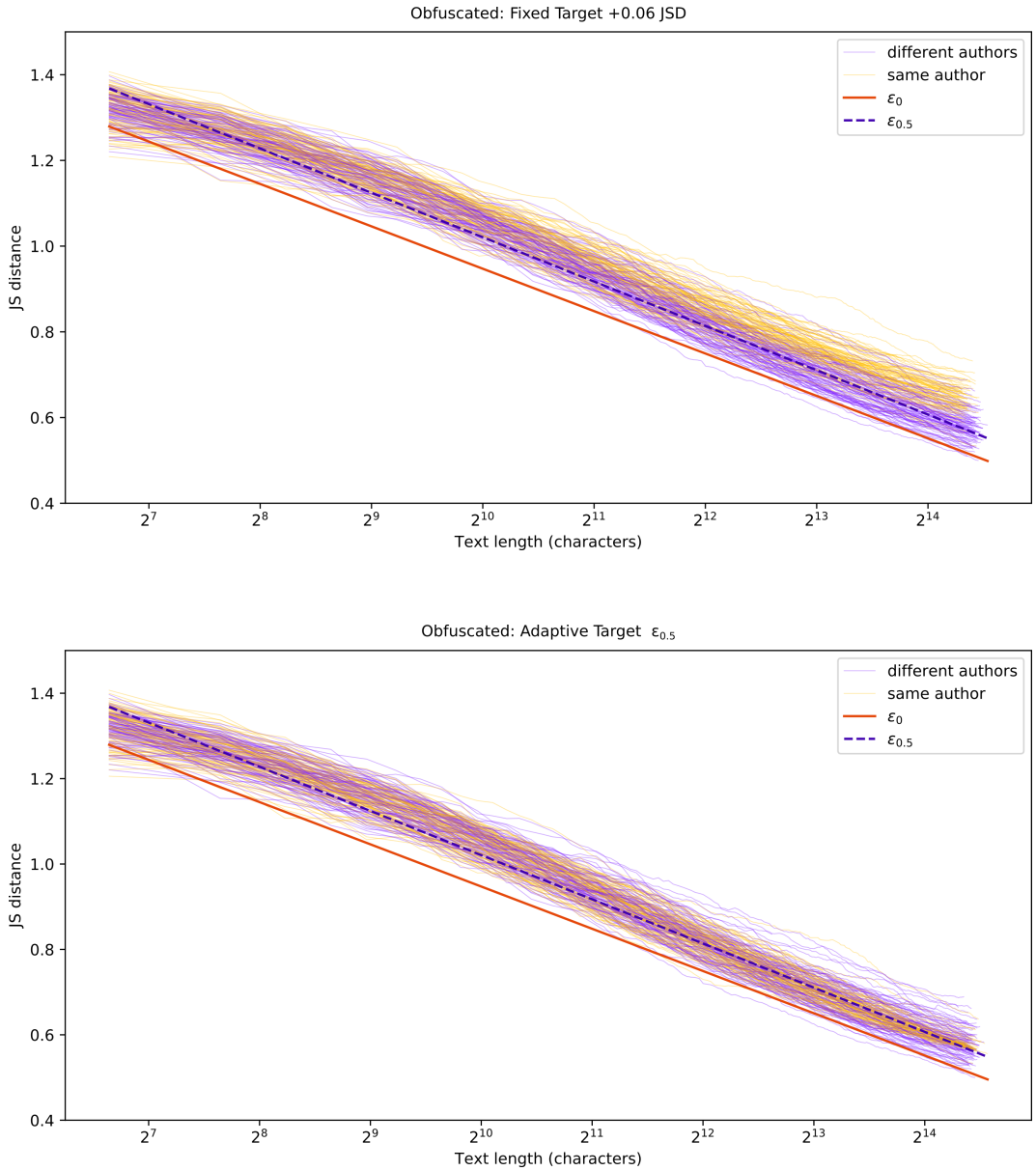

Figure 5.2: Jensen-Shannon distances in our training corpus after obfuscation with a fixed threshold of +0.06 JSD and with an adaptive threshold of $\varepsilon_{0.5}$. With the fixed threshold, we get the strongest result, but overobfuscate many cases. An adaptive threshold helps us obfuscate with only minimal effort.

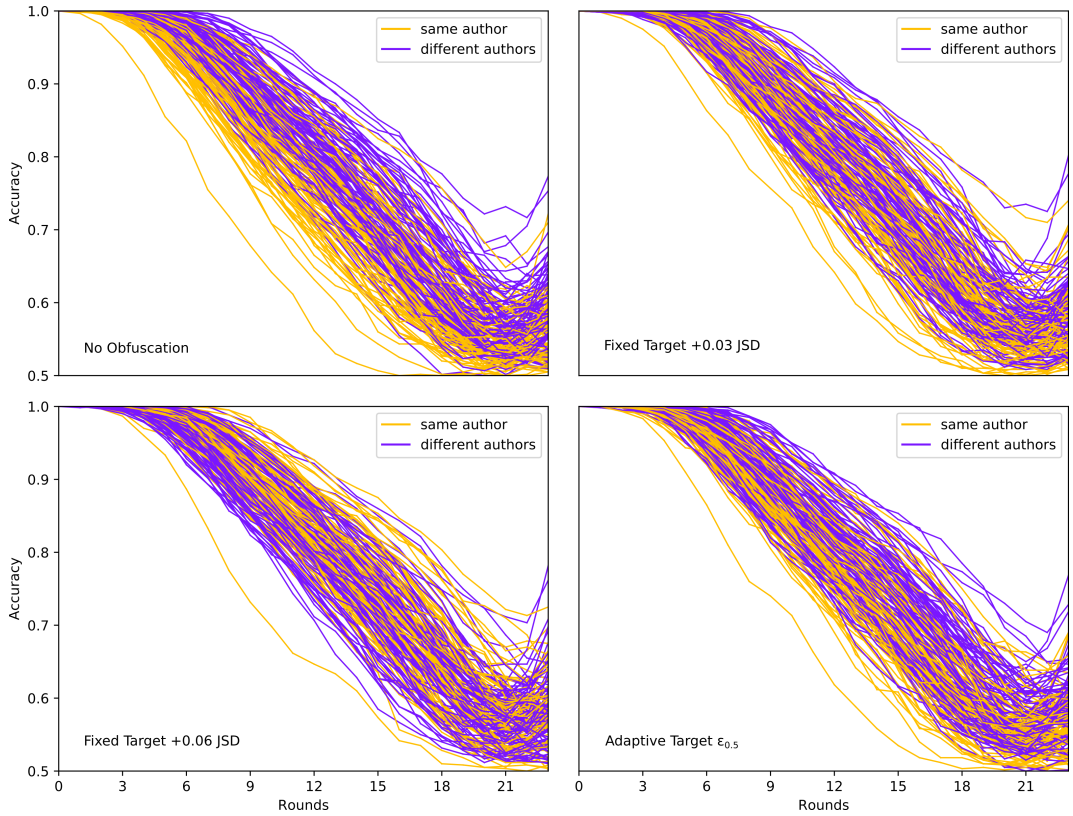

Figure 5.3: Unmasking curves of our training corpus before and after obfuscation with a fixed JSD threshold of +0.03, +0.06, and an adaptive threshold of $\varepsilon_{0.5}$. The overall effect of the adaptive approach on unmasking is lower than a fixed threshold of +0.06, but we can still successfully obfuscate most cases without overobfuscating already undecidable cases. Stronger obfuscation can be achieved by using a higher adaptive threshold.

by +0.06 JSD. This is expected, since +0.03 JSD is not enough to obfuscate all cases, but +0.06 JSD overobfuscates many cases. In sum, almost no *same-authors* cases remain which can be distinguished from *different-authors* cases with high confidence and acceptable precision. Stronger obfuscation can be achieved by using a higher ε threshold.

The ε_0 and $\varepsilon_{0.5}$ threshold lines we fit through our training data will cross the x axis at text lengths of $x \approx 2^{19.5}$ and $x \approx 2^{20}$ characters. Obviously, negative distances do not make any sense and JS_Δ@L curves will display shallower slopes for much longer text pairs. Adaptive ε thresholds are only accurate for short and medium-sized texts. Book-sized texts need different thresholds, which we will not examine here. We can, however, expect those thresholds to be largely constant, making fixed-threshold obfuscation viable for book-sized texts.

Chapter 6

Overview Heuristic Search

The following chapter gives a summary of the most important theoretical basics of heuristic search. The summary is based on the 1984 book *Heuristics: Intelligent Search Strategies for Computer Problem Solving* by Pearl [38].

6.1 Systematic Search

Many solutions to problems cannot (or are extremely hard to) be computed directly. To solve these problems, we can use search algorithms as strategies to navigate through the space S of possible solution candidates with the goal of finding the desired solution. Computer-based strategic problem solving can be modelled by formulating three basic requirements:

1. a **code** or **database** which can represent each candidate object in S,
2. a set of **operators** or **production rules** for transforming the encoding of an object into that of another to transition between objects in S,
3. a **control strategy** to apply operators so as to reach a desired object as quickly and efficiently as possible.

A simple baseline control strategy is random search, which randomly looks at all objects and decides whether they fulfil certain solution criteria. This approach is obviously inefficient and—without a good amount of luck—we might not find a solution at all in the time given to us. In order to do better, we want a control strategy to be **systematic**. A systematic control strategy

1. considers every object in S and
2. does not consider any object more than once.

The first requirement is the **completeness** requirement, which ensures that we 'leave no stone unturned' and therefore miss no opportunities to find the goal.

The second requirement avoids redundant computations. Hill-climbing, for example, is not a systematic method if the considered problem cannot be solved by local optimisation. If hill-climbing finds a local optimum or a large plateau, it will get stuck in it until it is restarted, which violates both requirements of systematic search. Similarly, random search is only systematic if we make sure never to look at the same object twice.

We can represent the space S as a graph in which every object of S is a node. Applying an operator SCS(n) on a node n transitions to a successor node n' and thereby describes an edge in the graph[1]. The new successor node n' is then said to be **generated** and its parent n is said to be **explored**. When *all* successors of a parent node are generated, this node is called **expanded**. The process of generating all successors of a node is called **node expansion**.

Examples of systematic search algorithms for finite search spaces are depth-first and breadth-first search. Depth-first search recurses the full (finite) search space and expands nodes in LIFO order, whereas breadth-first expands them in FIFO order. Both algorithms mean to search S exhaustively, which makes them unsuitable for very large spaces and entirely inapplicable to infinite spaces (although breadth-first search will find finite solutions if every node has a finite number of successors).

Basic depth-first and breadth-first search make decisions about which node to expand next only based on previously seen objects, which is why we call them **blind** or **uninformed**. Algorithms, on the other hand, which consider information about the nature of the goal which helps steer the search in a more favourable direction, are called **informed**. An informed search uses a **heuristic** to expand the most promising nodes first, which is nothing but a rule of thumb for how to reach a goal efficiently.

6.2 State Space Representation

To allow informed search and hopefully avoid exhaustive exploration, we extend the format of the code or database to also include information about the remaining subproblem at any node. We call this additional information **state** and the space of all subproblems spanned by operators from a given position a **state space**. A graph encoding of this space with states as nodes and operators as edges is called a **state space graph**.

As an example of a state, we use the famous travelling salesman problem (TSP). The salesman has travelled through cities A, B, C, and D and has to

[1]We have been using n to denote the order of n-grams, whereas in search theory it describes a node in the search graph. We will continue using n for both in accordance with the literature.

decide how to proceed through cities E and F and then return to A:

$$\underbrace{A \to B \to C \to}_{\text{Solution Base}} \underbrace{D \to \{E, F\} \to A}_{\text{State}}$$

The first part, the solution base, is necessary and sufficient to describe the whole tour as well as the remaining subproblem. The state describes only the latter and contains no information about the previous tour. It is therefore redundant and incomplete. However, storing an explicit state representation has the benefit of simplifying the computation of a heuristic and if we find a different tour with the same state, we can immediately discard the costlier one. This principle is called **pruning by dominance**.

6.3 Best-first Search and A*

With additional knowledge about the problem domain, we can build an informed or **best-first search**. What sets best-first search apart from other search algorithms is that it does not only expand the most promising node at the current position, but the most promising node of *all* nodes that have been generated so far. At any time, we have four disjoint node subsets, which are:

- nodes that have been expanded,
- nodes that have only been explored,
- nodes that have only been generated,
- nodes that have not yet been generated.

Expanded nodes are said to be **closed** and we place them on a list called CLOSED. Explored and generated (yet not expanded) nodes are said to be **open** and we place them on another list called OPEN. Whenever best-first selects a node for expansion, it chooses the most promising node from OPEN and then places it on CLOSED. To determine which node is the most promising one, best-first uses an evaluation function $f(n)$. A very simple evaluation function could estimate the closest distance to a goal node. Best-first search performed in this way is called **greedy**, since it is purely goal-oriented and does not make decisions based on previously incurred costs. Greedy best-first search can therefore choose deceiving and ultimately expensive paths or even end up in loops leading to the same states over and over again. A non-greedy variant of best-first search is the A* algorithm. The evaluation function $f(n)$ used by A* is composed not only of the goal heuristic $h(n)$, but also of a path cost function $g(n)$, which calculates the *actual* path costs incurred for reaching n

from the starting point s on the chosen path. The most promising node on OPEN is the one which minimizes

$$f(n) = g(n) + h(n) \ . \tag{6.1}$$

6.3.1 Admissible Heuristics

The A* algorithm can be proved to be **complete**, **admissible**, and **optimal** if we put some constraints on its heuristic function $h(n)$. As an explanation, an algorithm A_1 is called complete if it always finds a solution when one exists and it is admissible if this solution is guaranteed to be optimal. An algorithm itself is optimal over a class of algorithms if it **dominates** all other algorithms in this class. Dominance of A_1 over A_2 is given if all nodes expanded by A_1 are also expanded by A_2. A_1 **strictly dominates** A_2 if the reverse is not true. In order to guarantee that A* will never miss an optimal solution, we need $h(n)$ to be an **admissible heuristic**.

Let Γ denote the set of all goal nodes in S and let $h^*(n)$ denote the minimum of the *cheapest* path costs $k(n, \gamma)$ from n to any node $\gamma \in \Gamma$. A heuristic $h(n)$ is said to be admissible if it is an *optimistic* estimate of the true remaining costs $h^*(n)$, or mathematically speaking if

$$h(n) \leq h^*(n) \qquad \forall\, n \ . \tag{6.2}$$

If $h(n)$ fulfils the admissibility condition, A* will always find an optimal solution, since it will not expand a goal node γ before having expanded all other more promising nodes first. An admissible heuristic can prune many nodes and speed up the search significantly without risking missing an optimal solution. The pruning power of a heuristic is determined by its informedness. A heuristic $h_2(n)$ is said to be **more informed than** a heuristic $h_1(n)$ if both are admissible and

$$h_2(n) > h_1(n) \qquad \forall\, n \notin \Gamma \ . \tag{6.3}$$

6.3.2 Consistent and Monotone Heuristics

Admissibility guarantees an optimal solution, but it does not guarantee that we never have to reopen a node on CLOSED. Because multiple paths can lead to the same node in the state space graph, the algorithm may find a cheaper path to a node which has already been placed on CLOSED. In this case, the node needs to be reopened and its path has to be updated. This can be avoided if the heuristic is **consistent**. A heuristic function $h(n)$ is called consistent if for every node n and any of its descendants n', the following condition is true:

$$h(n) \leq k(n, n') + h(n') \qquad \forall\, n, n' \ . \tag{6.4}$$

Furthermore, a heuristic function $h(n)$ is said to be **monotone** if for a node n and any of its *direct* successors n', the following is true:

$$h(n) \leq c(n, n') + h(n') \qquad \forall\, n, n' \,|\, n' \in \mathrm{SCS}(n)\ , \tag{6.5}$$

with $c(n, n')$ denoting the operator costs for generating n' from n. It can be proved that consistency and monotonicity are equivalent properties. Monotonicity corresponds to the fulfilment of the triangle inequality by a heuristic and restricts the changes in $h(n)$ to be at most the size of $c(n, n')$. This implies non-decreasing f values:

$$f(n') \geq f(n) \qquad \forall\, n, n'\ . \tag{6.6}$$

Consistency and monotonicity guarantee that once a node has been expanded, the costs $g(n)$ for reaching it are optimal. As a result, no node on CLOSED must ever be reopened. Since $h(\gamma) = 0$ and $k(n, \gamma) = h^*(n)$, it can be shown that a consistent heuristic is also always admissible:

$$h(n) \leq k(n, \gamma) + h(\gamma) \Leftrightarrow h(n) \leq h^*(n) \qquad \forall\, n, \quad \forall\, \gamma \in \Gamma\ . \tag{6.7}$$

Chapter 7

Heuristic Search for Obfuscation

Deleting selected n-grams from a text is an easy and efficient method for obfuscating it, but it produces syntactically incorrect outputs and offers only first-order safety at best. In order to produce better and more inconspicuous obfuscations which follow at least syntactic rules, we want to rewrite parts of the text using automatic paraphrasing. The main difficulty lies in the fact that not only do we need to change a text, we also want to achieve a specific effect (i.e. obfuscation), which requires us to guide the paraphrasing mechanism in a certain direction. As we have seen before in Section 4.2.3, it is possible to change large portions of a text without having much of an effect on authorship verifiers, and we want to avoid these kinds of modifications. Moreover, we accept that we will not be able to fully rewrite a text without producing at least semantically incorrect paraphrases, and therefore want to perform as few text operations as possible.

With these three constraints: (1) achieving obfuscation, (2) producing syntactically correct outputs, and (3) changing as little of a text as possible, we cannot employ an uninformed text transformation process any more that simply terminates when a certain condition is satisfied. Instead, we have a very complex optimization problem. It is complex, because the problem's value range spans all possible paraphrases (or in general: modifications) of the source text, which is an exponentially large or, in fact, infinite amount of possible values, which we can never examine exhaustively. For that reason, it makes sense to formulate the obfuscation task as a search problem, which we can solve heuristically with the A* algorithm. As operators to span the search space, we define various kinds of text transformations or paraphrases. At any point during the search, a specific text is a state in the search space and a goal state is a text that is obfuscated according to our metric. The task of the search strategy is to find a path from the initial state to such a goal state with minimal costs. The cost calculation is derived from the problem description. A

minimal-cost path is therefore a path that has as little a negative impact on a text's syntax (and semantics) as possible. In practice, this usually implies performing as few operations as possible.

7.1 Developing an Obfuscation Heuristic

Before we start discussing actual operator implementations, we need to develop a useful heuristic function to guide the A* search algorithm. It does not matter which operation we perform on a text if we cannot estimate how valuable it is for reaching the obfuscation goal. For optimal results, we need a strong and robust heuristic function even if we continued using only n-gram deletion operations. We have been using JSD measures for our previous (non-search-based) obfuscation approach and established JS_Δ as a useful authorship metric, so we want to use it as a basis for the heuristic as well. Designing a goal check for terminating the search upon reaching a goal state is the easiest task. For this, we will use the adaptive JS_Δ@L metric at a predetermined ε threshold.

7.1.1 Naive Approach: Relative JS Distance

A naive approach to building a heuristic is to use the relative distance between a text pair's original JS_Δ and the adaptive target value, so $h(n)$ is the JS_Δ we still need to gain and $g(n)$ is the sum of the costs of all applied operators. This heuristic can definitely be admissible, but the obvious shortcoming of it is that there is no relation between path costs $g(n)$ and heuristic costs $h(n)$. The stepwise operator costs $c(n, n') = g(n') - g(n)$ of an operation are on a different scale than the heuristic and may not even be correlated with it at all. This relative JS_Δ heuristic is also bounded by $\sqrt{2}$, whereas path costs are not, so they will eventually dominate the heuristic, diminishing its influence on the total evaluation function $f(n)$. But even if step costs are carefully adjusted (we leave it open how that may be done), $h(n')$ will not necessarily reflect a step's path cost increase by $c(n, n')$ sufficiently in accordance with its contribution to reaching the goal. A unit of text operations is not the same as and not even immediately related to a unit of JS_Δ, so we may easily misguide the search.

7.1.2 Heuristic Based on Normalized Path Costs

A more useful approach is to estimate the remaining costs based on the JS_Δ gain we have achieved by previous text operations, which means to develop a heuristic that extrapolates path costs in proportion to JS_Δ change:

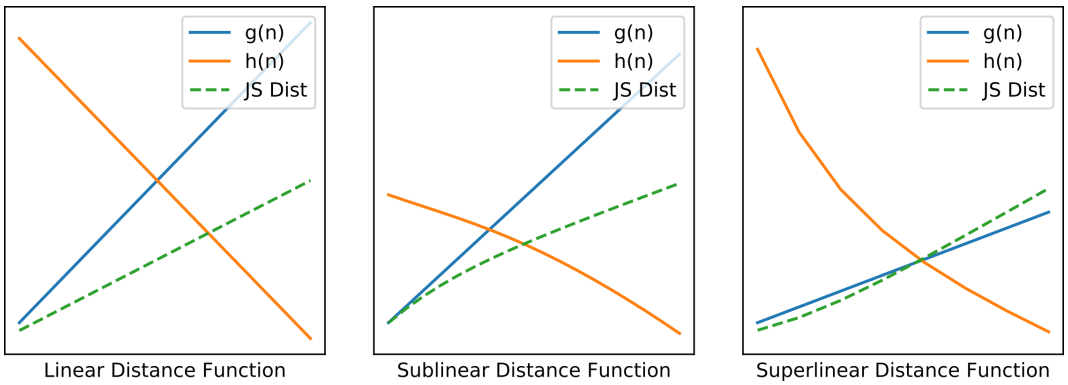

Figure 7.1: Schematic of the heuristic function $h(n)$ with constant step costs $c(n, n')$ and linear, sublinear, and superlinear cumulative JS_Δ gain functions.

$$h(n) = (\varepsilon - \mathrm{JS}_{\Delta n}) \cdot \frac{g(n)}{\mathrm{JS}_{\Delta n} - \mathrm{JS}_{\Delta 0}} \ . \tag{7.1}$$

ε denotes the adaptive JS_Δ@L target and $\mathrm{JS}_{\Delta 0}$, $\mathrm{JS}_{\Delta n}$ the JS_Δ values of the initial and the current state. We define

$$h_{prior}(n) = \varepsilon - \mathrm{JS}_{\Delta n} \tag{7.2}$$

as the **prior heuristic** and the cost to gain ratio

$$g_{norm}(n) = \frac{g(n)}{\mathrm{JS}_{\Delta n} - \mathrm{JS}_{\Delta 0}} \tag{7.3}$$

as the **normalized path costs**. In the product $h_{prior}(n) \cdot g_{norm}(n)$, normalized path costs determine the initial slope of the heuristic and the prior heuristic guarantees convergence towards zero as we approach a goal node $\gamma \in \Gamma$. We will use this product as the actual heuristic function $h(n)$ for the A* algorithm.

7.1.3 Consistency and Admissibility Properties

Because the proposed heuristic function $h(n)$ uses the proportion of path costs $g(n)$ to cumulative JS_Δ gain, it is naturally sensitive to variation in either quantity. As long as both are linear functions, the proportion will also be linear (schematically shown in Figure 7.1), resulting in an admissible, consistent and maximally informed heuristic function $h(n) = h^*(n)$. Of course, this is a rather unrealistic scenario. In reality, the proportion of path costs to cumulative JS_Δ gain will not be constant. If we only set fixed step costs $c(n, n')$ and thus have a linear cost function $g(n)$, then with a sublinear cumulative JS_Δ gain function, the heuristic will still be admissible, because it will initially

underestimate and then monotonically decrease from there on. On the other hand, a superlinear cumulative gain function will produce neither a consistent nor admissible heuristic. This heuristic still converges, but we will always overestimate at first as is clearly visible in Figure 7.1.

More generally, we can show (in)consistency for arbitrary differentiable cumulative cost and gain functions. If we recall the monotonicity condition $h(n) \leq c(n, n') + h(n')$ (which is equivalent to the consistency condition), rewrite it as

$$-h(n') + h(n) \leq g(n') - g(n) \ , \tag{7.4}$$

and insert the actual heuristic function, we come to the following inequality:

$$-\frac{\varepsilon - \mathrm{JS}_{\Delta n'}}{\mathrm{JS}_{\Delta n'} - \mathrm{JS}_{\Delta 0}} \cdot g(n') + \frac{\varepsilon - \mathrm{JS}_{\Delta n}}{\mathrm{JS}_{\Delta n} - \mathrm{JS}_{\Delta 0}} \cdot g(n) \leq g(n') - g(n) \ . \tag{7.5}$$

We define $\bar{g}(n) = \mathrm{JS}_{\Delta n} - \mathrm{JS}_{\Delta 0}$ as the gain function of a node and can write

$$\frac{-h_{prior}(n')}{\bar{g}(n')} \cdot g(n') - \frac{-h_{prior}(n)}{\bar{g}(n)} \cdot g(n) \leq g(n') - g(n) \ . \tag{7.6}$$

We know $h_{prior}(n)$ to be monotonically decreasing, inverse to $\bar{g}(n)$, and converging towards zero as we approach the goal. If cost and gain functions are equivalent up to scale, $g(n)$ and $\bar{g}(n)$ cancel each other out (up to scale), the slope of their quotient becomes zero and the inequality turns into an exact equality. If they are not equivalent, but $g(n)$ dominates $\bar{g}(n)$, the inequality will still be true. With 'dominate' we mean that $g(n)$ grows faster than $\bar{g}(n)$ and the gradient of their quotient is always positive. This obviously only holds given that both functions are differentiable at n. If $\bar{g}(n)$ grows faster, the sign of the quotient's gradient flips (as can be proved by the quotient rule), which breaks the inequality and violates the consistency condition.

Unfortunately, we will not see differentiable gain (or cost) functions in reality very often. Because texts are naturally noisy, the n-gram ranking function R_{KL} is only a loose guideline, and side effects are naturally unpredictable, actual stepwise JS_{Δ} gain will be very noisy as well. If we accidentally introduce a very bad side-effect n-gram, the cumulative gain function is not even guaranteed to be monotonic. Step costs $c(n, n')$ will never be negative, but also not constant when using multiple operators (and perhaps context-dependent cost calculation). That makes $g(n)$ monotonic, but not differentiable, so the heuristic function will not be fully consistent and may even overestimate. Figure 7.2 shows the influence of random noise on the heuristic function $h(n)$ with either the gain or cost function being subject to it. The heuristic is obviously sensitive to noise

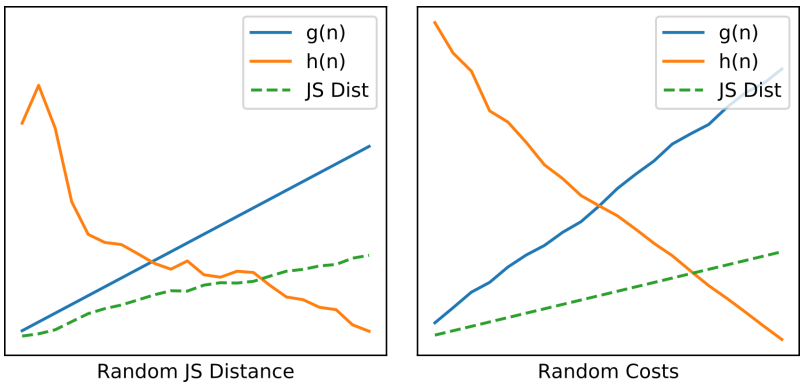

Figure 7.2: Schematic of the heuristic function $h(n)$ with random stepwise JS_Δ gain and random step costs $c(n, n')$.

in both functions. The strength largely depends on the index of dispersion of the noise and the random gain case is arguably worse, because the gain curve does not behave monotonically. It is easy to see that here neither heuristic is consistent or admissible, although, in another case, admissibility alone may be given. If (by accident), we start with a high gain, we will not overestimate, but still see large jump discontinuities, which break at least the consistency and monotonicity condition.

In a practical scenario, we can directly control the cost but probably not the already more problematic gain function, so we will have to deal with problems regarding overestimation and local optima. Generally, the first few steps of a search path are the most problematic, because with hardly any prior information, the heuristic has to extrapolate based on only a few data points, but is still expected to give an accurate estimate of the remaining costs. Hence, an early heuristic is particularly susceptible to noise and can only give a very coarse cost estimate. With more cumulative cost and gain information available, the heuristic will stabilize towards the mean cost-gain proportion and eventually converge towards zero.

In Figure 7.3, we show the heuristic in action. Step costs are constant, because we have only been using a single operator so far (which is n-gram removal) and do not consider context information such as the current state for step cost calculation. The noise function in grey shows the relative JS_Δ gain per step. As we suspected correctly, stepwise gain is noisy with high variance, causing $h(n)$ to produce a rather unstable cost estimate curve at first and then stabilize and converge towards the real remaining path costs. With only a single successor state being produced at each search state, inadmissibility or inconsistency is not problematic—with only one successor node to choose from, we effectively ignore the heuristic—, but one can imagine this noisy

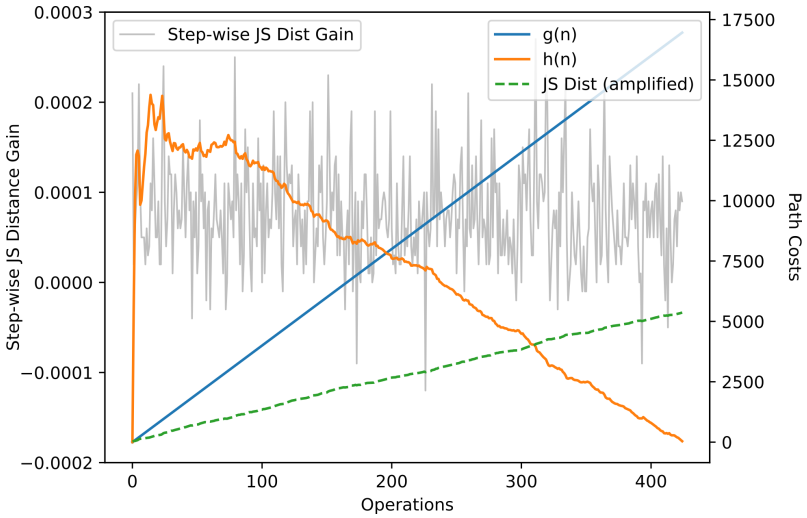

Figure 7.3: JS_Δ gain per step (grey) and $h(n)$, $g(n)$ curves when applying n-gram removal as the only operator to *Tales of Unrest* by Joseph Conrad. Cost estimation is very unreliable at first and converges towards the real remaining costs with increasing search depth. Cumulative JS_Δ gain has been amplified by a constant factor of $170,000$ for better visual representation on the heuristic and cost scale.

curve to pose a challenge later when we produce multiple successor states. In this example, we get a considerably smoother curve only after a depth of 50–100. With just two successors per operation, the fully expanded search space would be of the size of 2^{50}–2^{100} nodes already, so exploration for compensating for an inaccurate heuristic is not an option. Fortunately, JS_Δ is bounded by $\sqrt{2}$, so globally, cumulative gain cannot be superlinear and indeed, the cumulative JS_Δ gain function appears to be approximately linear (if we smooth out noise sufficiently). This suggests that the heuristic is at least asymptotically admissible and our main concern will be inconsistencies.

7.2 Developing Operators

With a heuristic in place, we can start developing more and better operators. In the following section we present a comprehensive but not exhaustive list of possible text operations that can be used to change an n-gram's occurrences or the word(s) containing it in specific places. A few operators are inspired by Bräutigam's [6, 48] thesis on generating acrostics via heuristic search.

The list is loosely ordered by increasing complexity of the operator and the sensibleness and soundness of its result. There is no formal method to measure how good an operator is, but to some extent, all operators can be assessed by

the two dimensions *strength* and *quality*. Strong operators are operators that are very effective and universally applicable throughout the search. Not all operators can always be applied (e.g. an operator that changes punctuation can only be applied on n-grams containing punctuation). With quality, we determine how well-formed or semantically similar the resulting text is. An operator will rarely be both strong and of high quality, so we must usually weigh strength and quality as a trade-off. There are operators which score well in both dimensions, but they tend to be very complex and costly in terms of runtime.

7.2.1 Asyntactic Operators

These operators are very strong and easy to implement, but tend to produce syntactic errors and generally very low-quality text results.

n-gram removal This is the operator we have been using so far. It is the strongest operator we can build if we neglect side effects, but also one of the lowest-quality operators overall. It will almost always produce syntactic errors, inhibit readability and can easily be spotted by automatic spell correction. The operator can be expanded to delete whole words, but without context, that procedure does not provide much more quality and rather worse comprehensibility.

Character flips Flipping positions of two neighbouring characters can also be applied universally, but suffers from the same problems as n-gram removal. We can trade strength for quality by only flipping characters which are rather inconspicuous or are close together on the keyboard. That would make this operator effectively a *typo* operator.

Character maps We can map specific characters to other replacement characters, so this is basically a synonym operator for characters. For example, we can replace exclamation marks by periods, commas by semicolons etc. If we restrict it to punctuation, this operator will be quite hard to notice, but its applicability is rather low. Extending it to also replace look-alike characters (e.g. lowercase L with uppercase I), makes the operator stronger, but also extremely easy to spot and revert by spell correction.

Common misspellings If not applied too often, application of a substitution map of correct words to common misspellings can be mildly effective and inconspicuous.

7.2.2 Syntactic Operators

These operators are still rather easy to implement, but either more restricted or they require a little more linguistic knowledge to produce at least syntactically correct results. No semantic or advanced grammatical knowledge is required for these operators.

Contractions and expansions In the English language, personal pronouns can often be contracted with a directly following auxiliary verb. 'I will' becomes 'I'll' and 'you are' becomes 'you're'. An operator can quite safely replace occurrences of these with their contraction. This operator is rather weak in terms of applicability, but the result will be syntactically correct and will not even change the meaning of a sentence. To make it a bit stronger, we can also replace already existing contractions with their expansions, but this way we cannot always guarantee correct results without further grammatical analysis. A contraction 'he's' is ambiguous and may have been either 'he is' or 'he has'.

Number words to digits Replacing number words with digits is quite an easy and inconspicuous operator, but it may be advisable to refrain from replacing too many single-digit numbers as it is usually considered bad style and can appear suspicious. The operator could also be applied vice versa, but with doubtable effectiveness. Especially larger numbers will rarely be indicative of writing style and therefore hardly ever be selected for replacement.

Abbreviations and acronyms Especially in technical writing, many words have common (and not so common) abbreviations and acronyms and quite a few of them can automatically be substituted without semantic knowledge. For instance, 'doctor' can be replaced by 'Dr.' and 'with regard to' by 'w.r.t.'. This mapping is not always perfect. A correct abbreviation for 'doctor' could also be 'Ph.D.' in some cases, but many replacements can be expected to be quite safe. Existing abbreviations could also be expanded, but then we need to be a lot more careful, because abbreviations are often extremely ambiguous.

Function word and adverb synonyms Many function words and adverbs are easy candidates for replacement. Their meaning is not influenced by context and they have no grammatical flexion. 'However', 'but', and 'yesterday' can probably be replaced by 'Nonetheless', 'yet', and 'the day before' without breaking anything. Some words are more problematic, such as 'since', which can either be a synonym for 'because' or describe time, yet many more unambiguous words remain.

Context-less synonyms This operator may also be called 'false synonyms' and it uses a thesaurus to select word replacements. As long as we do not

mix different parts of speech, the result should be grammatically sound in many cases with no guarantee for semantic correctness, however. The sentence 'She arrived late at work' could become something like 'She arrived late at endeavour', which is grammatically correct, but does not make a lot of sense. But applied in small pieces, this operator can be quite strong at reasonable quality.

Context-less hypernyms and hyponyms This is another, more specific 'false synonym' variant. Hypernyms are more general categories of other more specific words. For example, 'bird' is a hypernym for 'parrot'. Hyponyms on the other hand, are more specific words for a more general category. Conservatively applied, replacement by a hypernym tends to be rather safe if we do not mix parts of speech. On the other hand, replacement by hyponyms frequently leads to very confusing synonyms and may not necessarily be recommended, but is still an option. The operator can be implemented easily by extracting hypernyms from WordNet [36] synsets.

Context-less homonyms Commonly confused homonyms can be used as another false synonym operator. Homonyms are words that sound alike or are used in a similar context, but have different meaning, such as 'bear' and 'bare', 'their' and 'they're', or 'if' and 'whether'.

7.2.3 Context-based Operators

These operators live on a middle ground between purely syntactic and grammatical operators. They can be applied without grammatical or semantic knowledge, but are based on language models or other contextual information.

Run-on and split sentences This operator is similar to character maps. We can replace periods with commas to produce run-on sentences. Similarly, we can replace commas with periods to create split sentences. The difference between this and the character map operator is that we can only replace punctuation at the end of sentences or the beginning of clauses and must also change capitalization of the following sentence or clause.

Context-dependent replacements Word replacements and deletions based on context can be performed using word-level n-gram language models. The strength of this operator depends on the size of the used n-gram collection and the quality depends much on context size. Word five-grams are decent at determining if a word in question can be removed or replaced in the given context, but are rarely applicable. A bi-gram operator is stronger at the cost of quality. We implemented this operator using Netspeak [47], which utilizes the Google n-gram corpus, a web-scale word n-gram corpus with $n \in [1, 5]$.

Tautologies A tautology is produced by saying the same thing twice or by adding other 'useless' information. For disrupting word transition n-grams, we can insert new words between two existing words. This can either be done in a general fashion using Netspeak or by inserting related adjectives before nouns, such as 'frozen ice' or 'hot fire'.

Tautological phrases A tautological phrase with no meaning in itself and no influence on its context may be added between clauses. Such phrases are 'This is true', or 'As we all know...', etc. Their occurrence may be annoying or distracting, but does rarely change the meaning of a sentence.

7.2.4 Grammatical Operators

Operators in this category require more advanced grammatical knowledge of a sentence, up to some basic semantic understanding. The first two of them do not strictly require full grammatical parsing of a sentence, but benefit hugely from it.

Transition signals Many sentences or subclauses can be started with transition signals or introductory words without altering their meaning too much. These are 'Furthermore', 'whereas', 'however', 'indeed', 'anyway', 'notably', 'likewise', and many more. Care has to be taken only as to whether these are signalling continuation, contrast, or a new idea; otherwise we could completely negate the meaning of a clause. A correct choice needs at least grammatical parsing, but as a very rough approximation, contrasting signals could simply be omitted.

Tag questions The tag question operator is similar to introductory words and tautological phrases. Many sentences can be ended with semantically redundant tag questions, such as 'isn't it?', 'don't they?', or 'has he?' without altering meaning. While these phrases could just be appended without contextual analysis, appending the correct tag requires grammatical inference of the acting pronoun, used verb, and negation.

Pronoun substitution If the grammatical structure of a sentence is known, nouns and proper nouns can be substituted with referencing pronouns or vice versa. For example, 'The waiter gave us the bill' could be rewritten as 'He gave us the bill'.

Neural edits Specialized deep neural networks can be used for paraphrasing whole sentences [15] as mentioned shortly in Chapter 2. Full sentence paraphrasing is quite uncontrollable and may be too general of an operator,

but the supervised neural editor developed by Guu et al. [16] appears to be very promising for performing targeted edits of individual words. It allows a supervising system to mark specific words which the editor then tries to rephrase appropriately.

7.3 Design and Algorithmic Considerations

For running search-based obfuscation on texts, we developed an efficient C++ framework, which implements the A* algorithm for authorship obfuscation and allows to apply arbitrary text operators for transforming a text towards an adaptive obfuscation goal. When designing such a framework, special care must be taken to optimize execution time and memory consumption in order to make search on longer texts feasible and also enable application of more expensive operators. With an ASCII input text of 23,000 characters (or bytes), memory consumption will be at 23 MB after generating 1,000 successors in a naive way, i.e. by copying the string. Imagining that overall, we will generate tens of thousands if not hundreds of thousands of states (nodes), we will end up with hundreds of megabytes of memory consumed for representing them. This is not so problematic if the text were the only information we need to store, but reparsing a text into n-grams or words every time an operator is applied, demands an infeasible amount of computation time. If n-gram parsing takes 6 ms on an average CPU, we can generate a maximum of 167 nodes per second. This is only the bare successor state generation, but in practice we also need to run the actual operator, re-calculate a state's JS_Δ (for which we have to iterate all n-grams), and maintain the Open and Closed list. Overall, an actual state generation of 20 nodes/s or less seems more realistic. A simple way to increase performance is to pre-calculate n-grams and to store them in a lookup table together with the text. This reduces computation time, but causes memory consumption to increase immensely (we can easily fill gigabytes in no time) and copying the lookup table for every state is still overly costly.

To overcome these limitations, we represent n-grams as limited-precision integers (32 bit for ASCII trigrams to preserve alignment) and store a single pre-calculated profile of relative n-gram frequencies as an ordered tree map with access complexity $\mathcal{O}(\log n)$ together with the original text. Every state maintains a second ordered n-gram map containing only the updated n-grams and a differential string storing a series of incremental updates to the original. When accessing a state, the edit history is reiterated and both the current n-gram profile and the text are produced on-the-fly. The state's JS_Δ can be recalculated by iterating both the original n-gram map and the one having the updates simultaneously, hence the ordered tree data structure. We achieve

Table 7.1: Implemented search operators and their assigned fixed step costs in our heuristic obfuscation framework prototype. The context-dependent replacement operator has been split into actual word replacement and deletion.

Operator Name	Cost Value
n-gram removal	40
Character flips	30
Context-less synonyms	10
Context-less hypernyms	6
Context-dependent replacement	4
Character maps	3
Context-dependent deletion	2

both a performance gain and a decrease in memory consumption compared to reparsing the whole text and storing it with the state every time. If the update history of a state becomes too long, we can apply it and restart from a new base. As a further performance optimization, we can recalculate the JS_Δ only on the changed n-grams, but this calculation is highly inaccurate and needs to be corrected every five iterations. The OPEN list is maintained as a priority queue in the form of a min heap of pointers to nodes with access complexity $\mathcal{O}(1)$ for the lowest-cost node. We also maintain an $\mathcal{O}(1)$ hash map of the same pointers for random access to any node. The CLOSED list only needs constant access time and no ordering, so a hash map is the suitable data structure.

With the described architecture, our framework can generate up to 20,000 nodes/s on an average medium- to low-end desktop PC (Intel Core i5-6400, 2.7 GHz) with a single 'cheap' operator. In practice with multiple complex operators, we generate between 400 and 2,000 new nodes/s and move up to 100 nodes/s to CLOSED. Average memory consumption per state is around 10–15 KiB, including intermittently cached values and other overhead. For the first working prototype, we implemented the following seven operators: n-gram removal, character flips, character maps, context-less synonyms, context-less hypernyms, context-dependent replacement and context-dependent removal (both using Netspeak). Each operator was assigned a fixed cost value according to how we judge its overall impact on text quality.

7.3.1 Search Space Challenges

A major challenge we faced was generation and intelligent selection of successor nodes. We have seen before (Figure 7.3) the amount of noise that influences the gain function and the actual gain magnitude we are working with. With

stepwise JS_Δ gain varying heavily between (slightly less than) 0 and 0.0003, the search front becomes quite homogeneous and it is difficult to find the next best candidate for expansion with certainty. The gain magnitude itself is less important as the heuristic function will scale it up accordingly to match the magnitude of the path costs, but the variance imposes a serious challenge by not guaranteeing consistency or even admissibility as we established in previous sections.

The main challenge that we need to overcome is finding a sensible middle ground between finding a non-optimal solution too quickly (which can happen with inadmissible heuristics) or not finding a solution at all. We can easily turn the A* search into a depth-first search by generating only a single successor node at each step or by ensuring that expanding the next node is always cheaper than exploring neighbours. We can do the latter by setting operator costs to extremely low values or even zero. An obvious disadvantage of this method is that depth-first search will always find a solution when sufficiently many operations have been performed, but it will almost never be the optimal solution we are looking for, i.e. a solution which requires only very few and high-quality edits. On the other hand, by making expansion of new nodes too expensive, the search will deteriorate into breadth-first search and probably never finish before running out of time or memory. This pathological case will occur when the search runs into a local minimum of the heuristic cost estimate at which all successors are seemingly worse and farther away from the goal than all neighbours.

7.3.2 Partial Node Expansion

The most direct way to expand a node is to generate a successor with each applicable operator for each occurrence of the selected top-ranked n-gram. This approach is not ideal, since it generates a lot of successors very quickly and only leads to faster exponential search space explosion. The actual n-gram context may vary slightly, so side effects and context-dependent operator applicability may vary slightly as well, but in the worst case, this strategy will only produce a large number of similar states with identical costs, almost identical gain, and the same applicable operators to generate further successors. Expanding all these similar states will not get us quicker to our goal. A first countermeasure to make sure that the search does not plateau too early is to ensure that we do not generate too many of these similar successors. Fortunately, we are not reliant on a perfectly optimal solution, so only selecting one or two occurrences of an n-gram for expansion becomes justifiable.

A problem that remains to be solved is operator applicability itself. High-quality operators can be weak, but we want to apply them whenever possible.

We have seen that even by not selecting the single highest-impact n-gram, we can obfuscate with passable effort. Due to this observation, we can increase an operator's strength by not only selecting the top-ranked n-gram but a small number of different top n-grams. This way, we have multiple high-impact n-grams with different contexts to work with and increase the chances of applying a weak but high-quality operator. Additionally, we have opened alternative paths for the search itself. We have seen that JS_Δ gain is not a monotonic function and always selecting the single highest-impact n-gram does not guarantee an overall lowest-cost path. By selecting alternative n-grams, we allow the search to find a better path than the path of locally steepest ascent. Clearly, we have counteracted our previous search space reduction measure with this. To find a balance, we need to reduce the selection again before adding successors to the OPEN list. This can be done by random or cost-based subset selection.

We applied each operator on two occurrences of the top ten n-grams and then randomly reduced these (up to) 20 successors to six, which gives a total maximum of 42 successors per node. On average, this number will be much less, since not all seven operators can generate six successors every time.

7.3.3 Hybrid Search

Despite using only partial node expansion, we will still generate hundreds of thousands if not millions of nodes very quickly, fill up the OPEN list, and eventually run out of memory without finding a solution. Inconsistencies in the heuristic and the inherent gain similarity of most nodes make it hard to find a narrow low-cost path through the search space. Luckily, the inherent node similarity also enables us to regard a found intermediate solution as 'good enough'. It is plausible that, after a while, exploring more neighbouring nodes will not produce much better results, so we are allowed to clear OPEN and restart from a small subset of only the most promising nodes. This hybrid approach resembles a global depth-first search with best-first as extended node expansion. We chose to restart the search after 40,000 nodes on OPEN (approximately 600 MB, not counting nodes on CLOSED) and keep only the 10 most promising ones. We found this to be a good compromise between execution time and result quality. The average time to find a solution for a text of 23,000 characters with this approach is about 8.5 minutes on an Intel Core i5-6400 CPU with 2.7 GHz clock speed. The exact time varies with different cost configurations for the strongest operators.

Table 7.2: Performance values of adaptive local obfuscation in comparison with adaptive A*-based obfuscation on our test corpus. Mean and median values were calculated on cases which needed at least zero, one, and ten operations. We can reduce the median number of operations by up to 22 % and the median path costs by up to 70 %. Path costs for local obfuscation are the n-gram removal operator cost times the number of operations. Contrary to local n-gram removal, heuristic obfuscation does not reduce the text lengths any more.

		Mean			Median		
	Min. Ops	Local	A*	Gain	Local	A*	Gain
Operations	0	202	176	−13 %	185	176	−5 %
	1	259	225	−13 %	262	212	−19 %
	10	267	232	−13 %	275	215	−22 %
Length Ratio	0	0.97	1.01	+3.4 %	0.97	1.01	+3.4 %
	1	0.96	1.01	+4.4 %	0.96	1.01	+4.4 %
	10	0.96	1.01	+4.5 %	0.96	1.01	+4.5 %
Path Costs	0	8,113	2,380	−71 %	7,440	2,380	−68 %
	1	10,384	3,049	−71 %	10,500	3,278	−69 %
	10	10,711	3,146	−71 %	11,040	3,365	−70 %

7.4 Results Analysis

To distinguish between obfuscation by heuristic search and our previous uninformed proof-of-concept approach, we will call the latter *local obfuscation* from here on. The term *local* refers to this approach performing obfuscation based on local optimization along the KLD gradient.

By using heuristic search for obfuscation, we can successfully reduce the number of operations needed for achieving a desired obfuscation effect compared to local obfuscation. Despite the lower number of necessary operations, the effect on unmasking (as shown in Figure 7.4) remains unchanged and most *same-author* cases become undecidable after obfuscating them up to the adaptive $\varepsilon_{0.5}$ threshold. Some cases can still be distinguished from *different-authors* cases as a result of the target threshold being calculated on the training corpus. This can be corrected by choosing a higher threshold (e.g. $\varepsilon_{0.75}$) and is not an inherent effect of search-based obfuscation.

Table 7.2 lists performance measures on our test corpus for text operations required for reaching a goal node (which corresponds to the length of the solution path minus one), total path costs, and the text length ratio between the original and the obfuscated text. Path costs of local obfuscation are simply

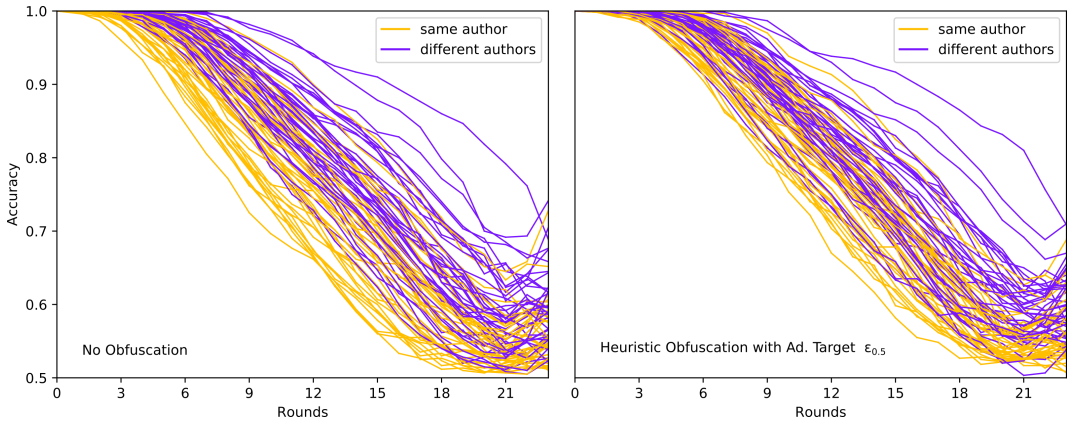

Figure 7.4: Unmasking curves of our test corpus before and after heuristic obfuscation up to the adaptive $\varepsilon_{0.5}$ threshold. The effect is comparable to adaptive local obfuscation. Stronger obfuscation can be achieved by choosing a higher threshold.

the product of the n-gram removal operator's costs and the number of applied operations. Since both obfuscators use adaptive goals, it makes sense to filter out cases which need no or only very few operations before calculating these performance measures. Therefore, Table 7.2 lists means and medians for all cases and only for cases requiring at least one and ten operations. We could reduce the median number of operations per text pair by up to 22 % and the mean by 13 %. Median path costs could be reduced by 70,%, but this is not at all surprising, considering that local obfuscation can only apply the most expensive n-gram removal operator. We can also see that heuristic obfuscation does not reduce the text lengths any more.

The actual text quality is harder to quantify. A cost decrease by 70 % suggests better results, but we must consider that we assigned operator costs mostly arbitrarily and none of the implemented operators is a very sophisticated grammatical one. To give a rough idea what a heuristically obfuscated text looks like, we show an excerpt from *A Filbert Is a Nut* by Rick Raphael. For comparison, this is the original text:

> `With a furtive glance around him, he clapped the other half of the clay sphere over the filled hemisphere and then stood up. The patients lined up at the door, waiting for the walk back across the green hills to the main hospital. The attendants made a quick count and then unlocked the door. The group shuffled out into the warm, afternoon sunlight and the door closed behind them. Miss Abercrombie gazed around the cluttered room and picked up her chart book of patient progress.`

```
Moving slowly down the line of benches, she made short,
precise notes on the day's work accomplished by each
patient.  [...]
```

and this is its obfuscated version (differences are highlighted):

```
With a furtive glance around him, he clapped the other
half of the clay sphere over the filled hemisphere
and then stood up.  The patients lined up at the
door, waiting for the walk back across the site hills
to the main hospital.  The attendants made a quick
investigation and then unlocked the door.  The group
shuffled out into the warm, daylight sunlight and the
door closed behind them.  Miss Abercrombie gazed around
the cluttered room and picked up her chart forward
of patient progress.  Moving slowly down the line of
bens, she made parcel, precise notes on the day's work
accomplishedb y aehc patient.  [...]
```

Overall, the text looks much better than a text with missing n-grams, but we can clearly see that the context-less synonym operator is lacking basic grammatical and semantic knowledge. However, as a prototypical implementation of heuristic obfuscation, the results are still satisfying. Synonym dictionaries can be improved and smarter operators can be implemented. What we also see is where the search failed to find suitable operators and resorted back to the more expensive and disruptive character flip and n-gram removal operators. We expect to see this whenever there are too few applicable high-quality operators. The obvious solution for counteracting this fall-back behaviour is to implement more and better operators. But we can also improve by allowing more successor states or choosing more different n-grams at the cost of execution time and memory. However, it is worth noting that only spending more time on finding a solution with low- and medium-quality operators, instead of adding more high-quality operators, imposes a risk.

We can demonstrate this with another experiment for which we adjusted the selection strategy to not only pick the top ten n-grams with highest rank, but to choose ten random n-grams from the positive half of the n-gram distribution. We left the number of occurrences for each of the ten n-grams at a value of two and the maximum allowed number of successors per operator at six. By randomly selecting also less effective n-grams, we limited the impact of the selection locally, but created more opportunities for the search to apply one of the weaker operators. As a result, nodes will be expanded more often up

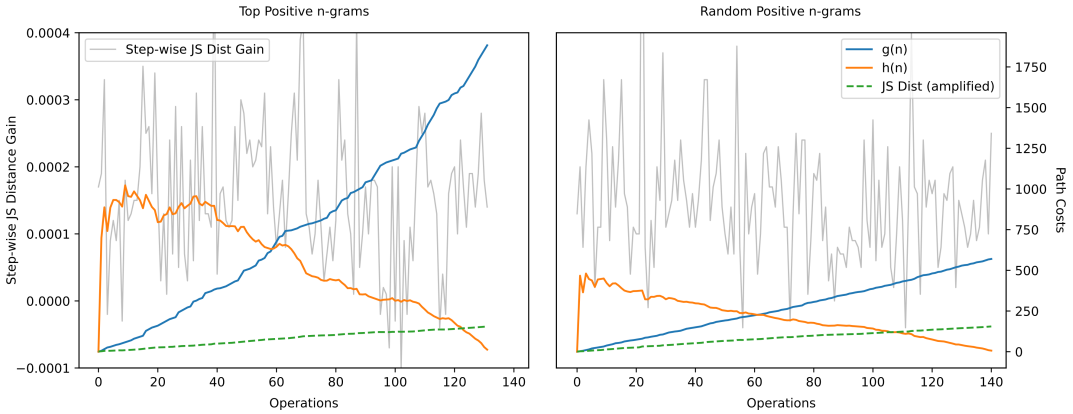

Figure 7.5: Heuristic cost estimate, actual path costs and JS_Δ gain for A*-based obfuscation of *A Filbert Is a Nut* using different n-gram selection strategies. The chart to the left shows costs and gain when selecting only the top ten n-grams and the one to the right when selecting ten random (positive) n-grams. The JS_Δ curves have been amplified by a factor of 7,500 for better visual representation.

to their actual maximum expansion cap of 42 successors, which leads to more successors overall and therefore opens more possible paths.

The experiment showed no notable increase in average required text operations, but average execution time per text pair rose from 8.5 minutes to over 35 minutes due to many of the selected n-grams having lower impact and the overall number of successful operator applications being higher. Despite the almost identical number of needed operations to obfuscate a text, we saw an average cost decrease from 2,380 to 740, which means that weaker low-cost operators were indeed applied more often. Shown at the same example text as above, the result is rather surprising:

> With a furtive glance around him, he clapped the other half of the clay sphere over the filled hemisphere and then stood up! The patients lined up at the door, waiting for the walk back across the green hills to the main hospital! The attendants made a quick investigating and then unlocked the door. The group shuffled out into the warm, afternoon sunlight and the door closed behind them, Miss Abercrombie gazed around the cluttered room and picked up her chart book of patient progress! Moving slowly down the line of benches, she made short, precise notes on the day's work accomplished by each patient. [...]

The A* search managed to obfuscate the text almost entirely by only applying the cheap but situational character map operator, which swaps out punctuation characters for other punctuation characters. We can show a more extreme example with *Tales of Unrest* by Joseph Conrad. This is the original text:

> They thronged the narrow length of our schooner's decks with their ornamented and barbarous crowd, with the variegated colours of checkered sarongs, red turbans, white jackets, embroideries; with the gleam of scabbards, gold rings, charms, armlets, lance blades, and jewelled handles of their weapons. They had an independent bearing, resolute eyes, a restrained manner; and we seem yet to hear their soft voices speaking of battles, travels, and escapes; boasting with composure, joking quietly; sometimes in well-bred murmurs extolling their own valour, our generosity; or celebrating with loyal enthusiasm the virtues of their ruler. [...]

The obfuscated text reads like this:

> They thronged the narrow length of our schooner's decks with their ornamented and barbarous crowd; with the variegated colours of checkered sarongs, red turbans; individual jackets, embroideries; with the gleam of scabbards. gold rings; charms, armlets, lance blades, and jewelled handles of their weapons. They had an independent bearing; resolute eyes, a restrained variety; and we seem yet to hear their soft voices speaking of battles, travels, and escapes; boasting with composure; joking quietly; sometimes in well-bred murmurs extolling their own valour; our generosity; or celebrating with loyal enthusiasm the virtues of their ruler! [...]

Except for some minor flow issues and two creative synonym replacements, the obfuscation looks close to ideal at first, but its problems are obvious: excessive repetition of a single operator becomes suspicious very quickly and obfuscation solely based on punctuation has no effect on purely word-based authorship verifiers like our unmasking implementation. Both problems can be tackled by increasing the costs of these cheap operators after repeated application,

but then we need more and stronger alternative operators again to find a better solution.

In summary, we can say that obfuscation by heuristic search can significantly reduce the number of text operations required for preventing authorship detection. Moreover, it can also produce better texts, but relies heavily on good choices for search parameters and both strong and high-quality operators.

Chapter 8

Conclusion

We developed an authorship verification scheme based on Koppel's unmasking, which allows us to decide the authorship of texts with only a few thousand characters in length with very high precision and we used this verification scheme as an adversarial baseline for our research into authorship obfuscation. By attacking the Kullback-Leibler divergence or Jensen-Shannon distance of a pair of texts written by the same author, we could show that only few text operations are necessary for rendering unmasking unable to determine the pair's shared authorship. The success of these obfuscative modifications supports our previous findings in the field of authorship obfuscation and endorses the usefulness of the Kullback-Leibler divergence as a basic model for authorship. However, we also identified potential pitfalls of this model and demonstrated that if operations are applied non-uniformly on a text, sufficient obfuscation is not guaranteed, despite the global KLD (or JSD) being increased.

To further lower the amount of operations needed for obfuscating a text, we proposed a text length–dependent method for determining an adaptive obfuscation target, which is the minimal Jensen-Shannon distance two texts must have for a certain level of obfuscation safety. To reach such an obfuscation target with as few text operations as possible and to improve the resulting text quality, we designed an obfuscation framework which uses heuristic search to control the applied obfuscative modifications. Using this framework, we could reduce the number of operations even more and also apply less disruptive and quality-deteriorating modifications.

As an overall result, we conclude that unmasking and—based on the tests on compression models and on other verifiers from PAN challenges during our previous work—possibly many other state-of-the-art verification systems can be defeated by very few targeted edits, which can, if performed cautiously, be subtle and not trivial to detect. Further, we could demonstrate prototypically how heuristic search can aid us in executing these edits automatically.

8.1 Future Work

We have shown that we can improve the quality of an obfuscated text compared to our previous proof-of-concept approach by using heuristic search. However, without better operators, we could not improve the quality of an obfuscation sufficiently to make it completely unrecognisable. Therefore, the most obvious improvement would be the implementation of grammatical operators, which can edit a text in a much less disruptive way. Also incrementally increasing costs of repeatedly applied operators can help preventing recognizable patterns in the text.

Once an obfuscated text looks inconspicuous enough to not be identified immediately by spell and grammar checkers or cursory human readers, heuristic obfuscation needs to be evaluated against statistical de-obfuscation attacks. Based on this evaluation, both the search strategy and operators may need to be adjusted in order to prevent such attacks. Using a more random n-gram selection (similar to the one we demonstrated towards the end of Section 7.4), may help avert de-obfuscation.

In addition, gathering additional data about the nature and variance of adaptive obfuscation thresholds on a larger number of text pairs and with more variety within the texts, would be useful to reduce the chances of accidental under- or overobfuscation. We have seen that adaptive thresholds learned on our training corpus were approximately but not totally accurate on the test corpus. Making these learned thresholds more robust and descriptive for a wider range of texts would overall increase obfuscation safety without having to resort to repeatedly testing against a (large) set of different classifiers to determine if an applied obfuscation is sufficient. We can also refine the definition of adaptive thresholds itself. For example, instead of defining a threshold geometrically, we could define it in terms of the type II error probability when deciding authorship based on the Jensen-Shannon distance.

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):7:1–7:29, 2008.

[2] Douglas Bagnall. Author identification using multi-headed recurrent neural networks. In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8–11, 2015.* CEUR-WS.org, 2015.

[3] Oleg Bakhteev and Andrey Khazov. Author masking using sequence-to-sequence models. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11–14, 2017.* CEUR-WS.org, 2017.

[4] Janek Bevendorff. Authorship verification and obfuscation using distributional features. Bachelor's thesis, Bauhaus-Universität Weimar, Fakultät Medien, Medieninformatik, September 2016.

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

[6] Christof Bräutigam. Einsatz heuristischer Suchverfahren zur Erzeugung eines Akrostichons. Master's thesis, Bauhaus-Universität Weimar, Fakultät Medien, Medieninformatik, September 2012.

[7] Michael Brennan, Sadia Afroz, and Rachel Greenstadt. Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity. *ACM Trans. Inf. Syst. Secur.*, 15(3):12:1–12:22, 2012.

[8] Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, Jennifer C. Lai, and Robert L. Mercer. An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18(1):31–40, 1992.

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

tools in translated text. In *Proceedings of Sixth IEEE International Conference on Semantic Computing, ICSC 2012, Palermo, Italy, September 19–21, 2012*, pages 121–125. IEEE Computer Society, 2012.

[10] Daniel Castro-Castro, Reynier Ortega Bueno, and Rafael Muñoz. Author masking by sentence transformation. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11–14, 2017*. CEUR-WS.org, 2017.

[11] John G. Cleary and Ian H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Trans. Communications*, 32(4): 396–402, 1984.

[12] Dominik Maria Endres and Johannes E. Schindelin. A new metric for probability distributions. *IEEE Trans. Information Theory*, 49(7):1858–1860, 2003.

[13] Tim Gollub, Benno Stein, and Steven Burrows. Ousting ivory tower research: towards a web framework for providing experiments as a service. In *Proceedings of the 35th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2012, Portland, OR, USA, August 12–16, 2012*, pages 1125–1126. ACM, 2012.

[14] Irving John Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3 and 4):237–264, 1953.

[15] David Grangier and Michael Auli. Quickedit: Editing text & translations via simple delete actions. ArXiv.org, 2017.

[16] Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. Generating sentences by editing prototypes. ArXiv.org, 2017.

[17] Matthias Hagen, Martin Potthast, and Benno Stein. Overview of the author obfuscation task at PAN 2017: Safety evaluation revisited. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11–14, 2017*. CEUR-WS.org, 2017.

[18] Oren Halvani, Christian Winter, and Lukas Graner. On the usefulness of compression models for authorship verification. In *Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, August 29 - September 01, 2017*, pages 54:1–54:10. ACM, 2017.

[19] Oren Halvani, Christian Winter, and Lukas Graner. Authorship verification based on compression-models. ArXiv.org, 2017.

[20] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Systems Science and Cybernetics*, 4(2):100–107, 1968.

[21] Farkhund Iqbal, Rachid Hadjidj, Benjamin C.M. Fung, and Mourad Debbabi. A novel approach of mining write-prints for authorship attribution in e-mail forensics. *Digital Investigation*, 5:S42–S51, 2008.

[22] Patrick Juola. Detecting stylistic deception. In *Proceedings of the Workshop on Computational Approaches to Deception Detection, EACL 2012*, pages 91–96. Association for Computational Linguistics, 2012.

[23] Patrick Juola and Efstathios Stamatatos. Overview of the author identification task at PAN 2013. In *Working Notes for CLEF 2013 Conference, Valencia, Spain, September 23–26, 2013.* CEUR-WS.org, 2013.

[24] Patrick Juola and Darren Vescovi. Empirical evaluation of authorship obfuscation using JGAAP. In *Proceedings of the 3rd ACM Workshop on Security and Artificial Intelligence, AISec 2010, Chicago, Illinois, USA, October 8, 2010*, pages 14–18. ACM, 2010.

[25] Patrick Juola and Darren Vescovi. Analyzing stylometric approaches to author obfuscation. In *IFIP Advances in Information and Communication Technology*, pages 115–125. Springer, 2011.

[26] Gary Kacmarcik and Michael Gamon. Obfuscating document stylometry to preserve author anonymity. In *Proceedings of ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Sydney, Australia, 17–21 July 2006*. The Association for Computer Linguistics, 2006.

[27] Alexei Kaltchenko. Algorithms for estimating information distance with application to bioinformatics and linguistics. In *Canadian Conference on Electrical and Computer Engineering 2004*, pages 2255–2258. IEEE, 2004.

[28] Yashwant Keswani, Harsh Trivedi, Parth Mehta, and Prasenjit Majumder. Author masking through translation. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5–8 September, 2016*, pages 890–894. CEUR-WS.org, 2016.

[29] Dmitry V. Khmelev and William John Teahan. A repetition based measure for verification of text collections and for text categorization. In *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 – August 1, 2003, Toronto, Canada*, pages 104–110. ACM, 2003.

[30] Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Proceedings of 1995 International Conference on Acoustics, Speech, and Signal Processing, ICASSP 1995, Detroit, Michigan, USA, May 08–12, 1995*, pages 181–184. IEEE Computer Society, 1995.

[31] Moshe Koppel and Jonathan Schler. Authorship verification as a one-class classification problem. In *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4–8, 2004.*, page 62, ACM, 2004.

[32] Muharram Mansoorizadeh, Taher Rahgooy, Mohammad Aminian, and Mehdy Eskandari. Author obfuscation using wordnet and language models. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5–8 September, 2016*, pages 939–946. CEUR-WS.org, 2016.

[33] 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 *Proceedings of Privacy Enhancing Technologies - 12th International Symposium, PETS 2012, Vigo, Spain, July 11–13, 2012*, pages 299–318. Springer, 2012.

[34] 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*, 2013.

[35] Tsvetomila Mihaylova, Georgi Karadjov, Yasen Kiprov, Georgi Georgiev, Ivan Koychev, and Preslav Nakov. SU@PAN'2016: Author obfuscation. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5–8 September, 2016*, pages 956–969. CEUR-WS.org, 2016.

[36] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244, 1990.

[37] Arvind Narayanan, Hristo S. Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, and Dawn Song. On the feasibility of internet-scale author identification. In *IEEE Symposium on Security and Privacy, SP 2012, 21–23 May 2012, San Francisco, California, USA*, pages 300–314. IEEE Computer Society, 2012.

[38] Judea Pearl. *Heuristics: Intelligent search strategies for computer problem solving*. Addison-Wesley series in artificial intelligence. Addison-Wesley, 1984.

[39] Lisa Pearl and Mark Steyvers. Detecting authorship deception: A supervised machine learning approach using author writeprints. *Literary and linguistic computing*, 27(2):183–196, 2012.

[40] Anselmo Peñas and Álvaro Rodrigo. A simple measure to assess non-response. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 19–24 June, 2011, Portland, Oregon, USA*, pages 1415–1424. The Association for Computer Linguistics, 2011.

[41] Martin Potthast, Tim Gollub, Francisco M. Rangel Pardo, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. Improving the reproducibility of PAN's shared tasks: Plagiarism detection, author identification, and author profiling. In *Proceedings of Information Access Evaluation. Multilinguality, Multimodality, and Interaction - 5th International Conference of the CLEF Initiative, CLEF 2014, Sheffield, UK, September 15–18, 2014.* Springer, 2014.

[42] Martin Potthast, Matthias Hagen, and Benno Stein. Author obfuscation: Attacking the state of the art in authorship verification. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5–8 September, 2016*, pages 716–749. CEUR-WS.org, 2016.

[43] Josyula R. Rao and Pankaj Rohatgi. Can pseudonymity really guarantee privacy? In *9th USENIX Security Symposium, Denver, Colorado, USA, August 14–17, 2000*. USENIX Association, 2000.

[44] D. Sculley and Carla E. Brodley. Compression and machine learning: A new perspective on feature space vectors. In *Proceedings of 2006 Data Compression Conference (DCC 2006), 28–30 March 2006, Snowbird, UT, USA*, pages 332–332. IEEE Computer Society, 2006.

[45] 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 *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15–18, 2014*, pages 877–897. CEUR-WS.org, 2014.

[46] 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 *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8–11, 2015*. CEUR-WS.org, 2015.

[47] Benno Stein, Martin Potthast, and Martin Trenkmann. Retrieving customary web language to assist writers. In *Advances in Information Retrieval, 32nd European Conference on IR Research, ECIR 2010, Milton Keynes, UK, March 28–31, 2010. Proceedings*, pages 631–635. Springer, 2010.

[48] Benno Stein, Matthias Hagen, and Christof Bräutigam. Generating acrostics via paraphrasing and heuristic search. In *Proceedings of COLING 2014, 25th International Conference on Computational Linguistics, Technical Papers, August 23–29, 2014, Dublin, Ireland*, pages 2018–2029. ACL, 2014.

[49] William J. Teahan and David J. Harper. Using compression-based language models for text categorization. In *Language modeling for information retrieval*, pages 141–165. Springer, 2003.

[50] Hoi Le Thi, Reihaneh Safavi-Naini, and Asadullah Al Galib. Secure obfuscation of authoring style. In *Information Security Theory and Practice - 9th IFIP WG 11.2 International Conference, WISTP 2015 Heraklion, Crete, Greece, August 24–25, 2015 Proceedings*, pages 88–103. Springer, 2015.

[51] John W. Tukey. *Exploratory data analysis*. Addison-Wesley series in behavioral science : Quantitative methods. Addison-Wesley, 1977.

[52] Wei Xu, Alan Ritter, Bill Dolan, Ralph Grishman, and Colin Cherry. Paraphrasing for style. In *Proceedings of COLING 2012, 24th International Conference on Computational Linguistics, Technical Papers, 8–15 December 2012, Mumbai, India*, pages 2899–2914. Indian Institute of Technology Bombay, 2012.

[53] Ying Zhao, Justin Zobel, and Phil Vines. Using relative entropy for authorship attribution. In *Information Retrieval Technology, Third Asia Information Retrieval Symposium, AIRS 2006, Singapore, October 16–18, 2006, Proceedings*, pages 92–105. Springer, 2006.

[54] 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.