Authorship Attribution

in a Forensic Setting

Wouter Hajer

Delft University of Technology | Netherlands Forensic Institute

TU Delft

Netherlands Forensic Institute
Ministry of Justice and Security

Authorship Attribution
in a Forensic Setting

by

Wouter Hajer

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday August 27, 2024 at 13:00.

Student number:	4957458	
Project duration:	December 1, 2023 – August 27, 2024	
Thesis committee:	Dr. J. Söhl,	TU Delft, Supervisor
	Dr. ir. T. Nane,	TU Delft, Chair
	A. F. van Luenen	NFI, Daily supervisor

TU Delft

Netherlands Forensic Institute
Ministry of Justice and Security

Preface

Before you is the result of 8 months of thesis research in authorship attribution. I look back at a very enjoyable internship at the Netherlands Forensic Institute (NFI), where I got the opportunity to use the knowledge acquired during my studies to work on a problem with real societal impact. I would like to thank everybody who helped me during my thesis project and will use this preface to thank some of them in particular.

Firstly, I would like to thank Jakob Söhl, my supervisor from the TU Delft, for reaching out about possible projects at the NFI over a year ago and for all his supervision since. Our bi-weekly meetings helped me to take a step back from my work and look at the bigger picture. This was of great value for the overall quality of my thesis. Additionally, I would like to thank Tina Nane for taking place in my graduation committee.

Secondly, I would like to thank Anne Fleur van Luenen, my daily supervisor at the NFI, for all her guidance during my thesis. Our weekly meetings helped me immensely with all aspects of carrying out my thesis project. I also would not have been able to navigate the bureaucratic hurdles of accessing real forensic data without her help. I also would like to thank Rolf Ypma for his valuable input and stimulating discussions during our meetings once every three weeks. Additional thanks go to all my other colleagues at the NFI from which I have learned a lot during lunch conversations, from demos of their projects and their questions and feedback during my demos. I also would like to thank all the other interns with whom I have shared room D2.07 during this project. They made my time at the NFI that much more enjoyable. Lastly, I would like to thank Hans van Halteren for his help with retrieving the abc_nl1 corpus.

Finally, I would like to thank my friends and family for all their support during my studies and in my thesis project. My thanks go to my friends of *Fysisch Incapabel* who I know supported me even though they had to act disappointed that I pursued my Masters in mathematics instead of physics; to Boyd and Koen, for all our lovely games of *mario kart* and *dominion* in between the in-depth discussions about our studies; to my parents and sisters, for all their support during my studies; and to Marthe, for all her love and support.

Wouter Hajer
The Hague, July 2024

Summary

Authorship attribution is the task of determining the unknown author of a text. In forensic authorship attribution, the likelihood that a suspect has written a specific text of unknown origin is computed based on reference texts from both the suspect and a background population. The current method used at the Netherlands Forensic Institute contains a manual and a computational part. In this thesis, we attempted to improve the computational part of this process. We study this problem from three directions.

Firstly, the performance of state-of-the-art computational authorship attribution methods was assessed on Dutch, forensically relevant corpora. The compared methods were support vector machines combined with masking, using either word or character n-grams as features, BERT-based models using a mean pooling strategy to handle long texts and the baseline, which consists of a logistic regression model with the 100 most frequent Dutch words as features. We notice similar performance differences between state-of-the-art methods as in the literature. The best-performing method was a support vector machine without masking using character n-grams as features. In comparison, both the baseline and BERT-based models perform worse on our corpora.

Secondly, a score-based likelihood ratio system was created to modify the computational authorship attribution methods for usage in forensics. This method is based on kernel density estimators and uses cross-calibration to handle the small number of training and calibration texts of the suspect. For most methods, the performance is in line with the previous performances outside the likelihood ratio system, except for the BERT-based methods, which significantly underperform when part of a likelihood ratio system. This is likely caused by the combination of cross-calibration and the randomness in finetuning BERT models.

Additionally, authorship attribution methods should be topic-robust, such that their attribution is not biased by the topic of a text. We introduced two new metrics to measure the topic-robustness of authorship attribution methods, ‘topic impact’ and ‘conversation impact’. These metrics can only be used on specific types of corpora, the topic impact can be computed on topic-controlled corpora and the conversation impact can be computed on conversational corpora. To study whether these metrics both measured the topic-robustness of authorship attribution methods for their respective corpus type, we computed the correlation between the results of the metrics for varying authorship attribution methods. We found a correlation of 0.68. As a result, we cannot conclude that the conversation impact is a perfect metric to measure the topic-robustness of methods using conversational corpora, but it does give a good indication of large differences between methods.

Using this new metric, we found that our best-performing methods suffered from a high conversation impact and, as a result, might be more likely to have a low topic-robustness. If more of the infrequent words were masked, the conversation impact decreased, but so did the performance. A trade-off between high performance and high topic-robustness must be made when a model is chosen for real forensic case work. The conversation impact metric we proposed can help quantify these effects on forensically relevant corpora and therefore assist in making better choices.

Contents

Preface i

Summary ii

Nomenclature v

1 Introduction 1
- 1.1 Structure . . . 2

2 Related Work 3
- 2.1 Linguistics . . . 3
- 2.2 Computational Authorship Attribution . . . 3
  - 2.2.1 The State of Computational Authorship Attribution before 2010 . . . 3
  - 2.2.2 Developments in Authorship Attribution . . . 4
  - 2.2.3 Recent comparative studies . . . 5
- 2.3 Forensic Science . . . 6
- 2.4 Authorship Attribution in Dutch . . . 7

3 Corpora 8
- 3.1 FRIDA . . . 8
- 3.2 RFM . . . 8
- 3.3 abc_nl1 . . . 9

4 Feature-based Authorship Attribution 10
- 4.1 Features . . . 10
  - 4.1.1 Summary statistics . . . 10
  - 4.1.2 Function words . . . 10
  - 4.1.3 Word n-grams . . . 10
  - 4.1.4 Character n-grams . . . 11
  - 4.1.5 POS n-grams . . . 11
  - 4.1.6 Special characters . . . 11
- 4.2 Support vector machines . . . 11
  - 4.2.1 Hard margin SVM . . . 11
  - 4.2.2 Soft margin SVM . . . 12
  - 4.2.3 The SMO algorithm . . . 12
  - 4.2.4 Multiclass SVM . . . 12
  - 4.2.5 Kernels . . . 13
- 4.3 Implementation . . . 13
  - 4.3.1 Feature vector . . . 13
  - 4.3.2 SVM . . . 15
- 4.4 Masking . . . 16
- 4.5 Background vocabulary . . . 16
- 4.6 Baseline . . . 17

5 BERT 18
- 5.1 Model architecture . . . 18
  - 5.1.1 Tokenization . . . 19
  - 5.1.2 Encoding . . . 19
  - 5.1.3 Multi-head attention . . . 20
  - 5.1.4 Feed-forward neural network . . . 20
  - 5.1.5 Pre-training . . . 21
- 5.2 BERT models for authorship attribution . . . 21

5.3 Subverting the token limit . . . 21
5.3.1 Truncation . . . 21
5.3.2 Averaging classifications . . . 21
5.3.3 Mean pooling before classification layer . . . 22
5.4 Dutch BERT models . . . 22

6 Likelihood Ratio Framework **23**
6.1 Common Source problem . . . 23
6.2 Specific Source problem . . . 23
6.3 The CS and SS problem in authorship identification . . . 23
6.4 Likelihood Ratio . . . 24

7 Likelihood Ratio Systems **25**
7.1 Binary scorer . . . 25
7.2 Kernel Density Estimation . . . 25
7.2.1 Cross-calibration . . . 27
7.3 Feature-based likelihood ratio systems . . . 27
7.4 Dataset Balance . . . 28
7.5 Bounding . . . 28
7.5.1 Bounds on the optimal LR system . . . 29

8 Validation **31**
8.1 Metrics in computational authorship attribution . . . 31
8.2 Validation in forensic sciences . . . 32
8.2.1 Accuracy . . . 33
8.2.2 Discriminating power . . . 34
8.2.3 Calibration . . . 37
8.2.4 Graphical representations . . . 37
8.2.5 The choice of V . . . 39

9 Topic and Conversation Impact **41**
9.1 Topic impact . . . 41
9.2 Conversation impact . . . 42
9.3 Practical implementation . . . 44

10 Results and Discussion **45**
10.1 Computational authorship attribution . . . 45
10.1.1 Feature vector and SVM models . . . 45
10.1.2 BERT-based models . . . 46
10.1.3 Comparison . . . 47
10.1.4 Influence of text length, number of texts and number of authors . . . 47
10.2 Forensic authorship attribution . . . 50
10.2.1 Log-likelihood ratio cost . . . 50
10.2.2 Tippett plots . . . 52
10.2.3 ECE plot . . . 54
10.2.4 PAV plots . . . 55
10.3 Topic and conversation impact . . . 56

11 Conclusions and Recommendations **60**
11.1 Conclusions . . . 60
11.2 Future research . . . 61

References **62**

A Proof that the weighted average of the C_{llr} of two sets is the same as the C_{llr} of their union **65**

B Tables of results presented in figures **67**

C Tables of additional experiments results **70**

Nomenclature

Abbreviations

Abbreviation	Definition
AA	Authorship Attribution
BERT	Bidirectional Encoder Representations from Transformers
CAA	Computational Authorship Attribution
CS	Common Source
ECE	Empirical Cross Entropy
ELUB	Empirical Lower and Upper Bound
EU	Expected Utility
FAA	Forensic Authorship Attribution
KDE	Kernel Density Estimation
LR	Likelihood Ratio
MLM	Masked Language Modelling
NFI	Netherlands Forensic Institute
NSP	Next Sentence Prediction
PAV	Pool Adjacent Violators
POS	Part-of-Speech
PDF	Probability Density Function
RFM	Real Forensic Messages
SMO	Sequential Minimal Optimization
SS	Specific Source
SVD	Singular Value Decomposition
SVM	Support Vector Machine

1 Introduction

In 2018, police agents found a dead man in the Dutch town of Heeg [17]. Shortly before his death, a last e-mail was sent from his computer, in which it is explained that after his death €17,000 should be transferred to a client. The police asked the Netherlands Forensic Institute (NFI) to investigate whether this e-mail was written by the victim or one of two other persons who could have accessed this computer. If this e-mail was indeed written by one of the two others, this could be used as evidence that they were involved with his death.

This is an example of a forensic authorship attribution (FAA) case. In such cases, there is a text of unknown origin, here the e-mail, and several known texts, also called reference texts. The reference texts consist of both texts written by the suspect and texts written by the other possible authors, the reference population. In the example, the reference texts could be earlier emails written by the victim and the two possible suspects.

The current method used at the NFI has a dual approach. Firstly, for each possible author, a language expert looks through all their reference texts and observes features that are indicative of their writing style. This could be, for example, the usage of uncommon words or consistent spelling mistakes in certain words. It is important that the experts go blindly into this endeavour: they should not have seen the text of unknown origin, nor should they know which author has written the texts they are reading, to keep bias at a minimum. We will call the notable features found by language experts the manual features.

Secondly, the 100 most frequent Dutch words are also used as features. The occurrences of all both the manual features and the most frequent words are then counted in all texts and normalized by the number of words in the text. The resulting feature vectors of the known texts are then used to train and calibrate a classification model based on logistical regression that can compute the value of evidence of the unknown text. In this thesis, we study methods to improve the quality of these systems. Due to the manual work involved in finding the manual features in the current method, we will focus on the second part, the features consisting of the 100 most frequent words. We will set this method as the baseline method in our research. In real case work, manual features can be added to the feature vector of the models we study to get a complete model.

In recent years, the field of computational authorship attribution (CAA) has seen significant improvements in performance and the current state-of-the-art models are much more complex than using the current baseline [37, 54]. The differences between this field and that of forensic authorship attribution are twofold. Firstly, there is a difference in the used datasets, also called corpora. In computational authorship attribution often large corpora of books, blogs, movie reviews, internet comments or fanfiction are studied, as these are easily accessible on the internet [54]. In forensic cases, the studied texts are often of a more private nature, like chat messages or e-mails, which are often smaller in size and not as easily accessible for most researchers compared to online corpora. Additionally, most research in CAA has been performed on English corpora and to a lesser extent on Spanish, French and German corpora [39, 54], while in forensic cases at the NFI authorship attribution is mostly used on Dutch texts. Due to the difference in studied texts, other methods might perform better on forensically relevant datasets than on the datasets studied in CAA. Therefore, the first aim of this thesis is to study the performance of current state-of-the-art CAA methods on forensically relevant, Dutch corpora and to compare their results with the baseline.

The second difference between computational and forensic authorship attribution is the result of the model. In CAA the models need to attribute a text to one specific author and the performance of models is judged based on metrics based on the percentage of correct attributions. However, in an FAA case, the goal is not just to attribute a text to the correct author, but also to estimate how much more likely that author is to have written the text compared to the reference population of alternative authors. This way, a judge can combine the value of evidence with the value of other pieces of evidence to make a judgement. A so-called likelihood ratio system, which can be based on various authorship attribution methods, can be built to calculate the value of evidence. To the best of our knowledge, this specific area is understudied in terms of publications [20, 25], especially when compared to the adjacent fields of CAA and likelihood ratio systems for the authorship verification task. We want to expand the literature in FAA using the NFI's likelihood ratio framework [32] combined with state-of-the-art authorship attribution models. This leads us to the second aim of this thesis, to study the performance of likelihood ratio systems built using current state-of-the-art CAA methods in comparison with the baseline.

A risk in the field of FAA is the impact of the topic of a text on its attribution. Suppose a hypothetical case where our reference set consists of texts about football written by Alice and texts about painting written by Bob. Suppose a model attributes new texts to either Alice or Bob based on the frequency of all words used in the text. In Alice's known texts, words like team, field, game, and referee might frequently be used, while Bob might use words like canvas, brush, pigment, and colour. If Alice now writes a text about painting, the model might attribute this text to Bob, because the text contains painting-related words which Alice did not use in her texts about football. The same effect can happen when the model is tested on a text written by Bob about football.

In FAA, the topic of the unknown text can be significantly different from the topics of the reference texts. An example of this is cases where we want to use messages from a suspect's regular phone to identify the author of text messages sent from a burner phone, a second phone only used for criminal purposes. The topics discussed on the burner phone are likely different from topics discussed in text messages on the regular phone. As a result, it is desirable to use models that do not suffer attribute texts based on topic, which we call topic-robust models.

Recently, some studies have tested the influence of topic on authorship attribution in corpora where several authors have each written texts about the same varying topics [1, 52], also called topic-controlled corpora. However, to the best of our knowledge, a corpus that is both topic-controlled and forensically relevant has not yet been created. To still be able to test the topic-robustness of state-of-the-art CAA methods on forensically relevant datasets we propose the usage of conversations as a proxy for the topic. If two people are having a conversation, with for example chat messages or e-mails, they are likely writing texts about the same topic. Therefore we might be able to use both halves of a conversation as two texts written by different authors about the same topics. To validate this idea we want to define metrics to measure the impact of topic on a topic-controlled corpus and the impact of the usage of conversations as a proxy for topic in a conversation corpus. If the results of various models are correlated, this might support the possibility of measuring the topic-robustness of models on forensically relevant conversation corpora. We summarize this in our third and final research aim: to define metrics to calculate both the topic and conversation impact and study their correlation.

1.1. Structure

We will start by studying the related work and identifying research gaps in Chapter 2. In Chapter 3 we introduce all datasets used in this thesis. We will proceed with the theory behind two state-of-the-art authorship attribution methods, feature vectors with support vector machines (Chapter 4) and BERT-based models (Chapter 5). Then we will explain the theory behind likelihood ratios (Chapter 6) and how to use a computational authorship attribution method in this likelihood ratio framework (Chapter 7). Having covered all this theory, we will continue by showing how to validate both computational and forensic authorship attribution methods (Chapter 8). As our last theory chapter, we will define the metrics 'topic impact' and 'conversation impact' in Chapter 9. We continue by presenting and discussing our results in Chapter 10. Lastly, we will draw our conclusion from these results and make recommendations for future research in Chapter 11.

2 Related Work

As authorship attribution is a multi-disciplinary field, research in this area is performed from a variety of backgrounds and with different aims. Three main categories of research done in authorship attribution are studies with either a linguistic, a forensic or a computer science focus. In this chapter, we will describe the previous work done in these subfields, with a focus on both computational and forensic authorship attribution, as those fields are closely related to this thesis. Furthermore, we will give an overview of the previous research performed on Dutch texts. As this chapter is meant to give an overview, we will not go into depth into many of the mentioned techniques. For the techniques we use in this thesis, we will reference the section in which they will be explained in depth.

2.1. Linguistics

The origin of authorship attribution is located in the field of linguistics, with studies discussing the authorship of famous anonymous texts or books. The example that is often [50] referred to as the first study in authorship attribution is the study by Mosteller and Wallace [36] into the origin of the Federalist papers. Much of this research performed before computers could handle large sets of features was done by finding specific features by hand. This could be specific words or sentences that only one of the authors uses or summary statistics like the average length of words or sentences. This feature could then be quantified and counted in both the known and unknown texts to try to determine the author of the unknown text.

2.2. Computational Authorship Attribution

The field of computational authorship attribution is the largest of the three subfields. We will therefore divide the previous research in this subfield into three parts. We will first illustrate the state of computational authorship attribution before 2010, as around that time three influential works were published that reviewed the state of the field, by Juola [24], Koppel et al [29] and Stamatatos [50]. From this baseline, we will describe all major publications and newly proposed methods of the following years. Lastly, we will cover the results of two recent comparative studies in which the performance of a variety of methods is compared on several of the most studied corpora [37, 54].

2.2.1. The State of Computational Authorship Attribution before 2010

In his book, Juola [24] extensively covers the history of authorship attribution and describes the results of the so-called Ad-hoc Authorship Attribution Contest he organised in 2004. During this contest corpora from a variety of languages were included, specifically English, French, Serbian, Latin and Dutch. Of the top three methods in the contest, the first one used a support vector machine (SVM) with "unstable words", words with commonly used substitutes, as features. The other two methods were a character n-gram method with k-nearest neighbour classification and a profile-based method using features including word n-grams, character n-grams and summary statistics. In Chapter 4 we give an in-depth discussion of the varying features mentioned here and the theory behind SVMs.

Koppel et al [29] compares five sets of features with five machine learning techniques for classification on three English corpora. The three corpora consist respectively of e-mails, literature and blogs. In this comparison, they find the best results using a feature set containing the 1000 character 3-grams/words with the highest “information gain” from the 10.000 most frequent 3-grams/words in the training corpus. Furthermore, when combining these feature sets the performance is improved even more. On their corpora, two classification methods show the best results, namely Bayesian multi-class regression and a support vector machine with a linear kernel.

Stamatatos [50] did not evaluate and compare several methods quantitatively, but gives an extensive qualitative overview of many previous publications. The features used in previous research are subdivided into four categories: lexical, character, syntactic and semantic. The methods are described as part of two categories, profile-based and instance-based approaches. In a profile-based approach, all training documents of a single author are concatenated into a single document, of which summary statistics are used to determine if it is related to an unknown text. On the other hand, in an instance-based approach, the individual training texts are used to train a classifier, which then classifies the unknown texts. Furthermore, he describes an important issue at hand, namely how to discriminate between three factors that impact texts: authorship, genre and topic. In many of the corpora used at the time, there is a correlation between author and topic, as people tend to discuss similar topics between different texts. In 2009 a new English corpus was published in which topic and genre were controlled [18], leading to many studies in the subfield of cross-domain authorship attribution. The most influential of those are included in the following section.

2.2.2. Developments in Authorship Attribution

Sapkota et al [46] studied the performance of different types of character 3-grams. They divided these n-grams into three main categories: affix n-grams, word n-grams and punctuation n-grams, with further subdivision in these categories. They use these n-gram types as feature sets to show that on a single-domain corpus affix n-grams perform significantly better than other types of n-grams, while on a cross-domain corpus, both affix and punctuation n-grams get the best results. That the relative usefulness of punctuation n-grams increases on a cross-domain might be a sign that these are features of an author’s style that are better conserved across topics.

In a 2018 paper Stamatatos [52] proposes a new method to tackle cross-domain authorship called masking. In this technique words that occur less frequently are replaced by asterisks and all digits are replaced by hashtags. Upon the modified dataset they utilize two different methods, character 3-grams with a support vector machine and prediction by partial matching (PPM) [53]. They show that the use of character 3-grams in combination with a support vector machine and masking works significantly better than the same system without masking in cross-topic authorship attribution. This result is the strongest when only 100 - 200 of the most frequent words are not masked. For PPM no significant increase was seen in accuracy when using masking and the overall performance was lower than that of the character 3-gram methods. Four different methods for masking were proposed, where words outside of the most frequent words were replaced by either one asterisk per letter, one asterisk per word, one asterisk per letter while leaving the exterior letters intact or one asterisk per letter while leaving the last two letters intact. The performance of these methods was comparable, with the method replacing each letter with an asterisk outperforming the other methods slightly. We discuss masking and our implementation of it further in Chapter 4.4.

During the yearly PAN/CLEF conference shared tasks on digital text forensics and stylometry are done. Authorship Attribution tasks have been included during some of these years, most recently in 2019 [27]. During this task, the focus was on cross-fanfiction, a variant of cross-domain, authorship attribution. In this task, the texts in the training set are about a different fandom (e.g. Harry Potter, Lord of the Rings) than the texts in the validation set. Additionally, some of the texts in the validation sets were from authors outside the author set and have to be classified as unknown. All good performing strategies in this task were n-gram and SVM-based. The best performance was by Muttenthaler et al [39] using an SVM with as features an ensemble of word and character n-grams, disregarding the 50% least frequent features. Additionally, they used dimension reduction by singular value decomposition. We study the implementation of the method by Muttenthaler et al in-depth in Chapter 4.3.

Transformers

The use of transformers in authorship attribution was first proposed by Fabien in 2020 [16]. This paper introduces the method BERTAA, which is based on the pre-trained language model BERT [14]. We discuss the inner workings of BERT models and the implementation of BERTAA more in-depth in Chapter 5.

Additionally, Fabien explores ways of combining BERTAA with stylometric features and n-grams [16]. This is done by building two individual logistic regression models, one using an array of stylometric features and the other using the hundred most frequent n-grams. Then another logistic regression model is fitted to combine the outputs of the three models. The addition of these features did not lead to significant improvements in the macro-accuracy of the model [16].

In 2021, Barlas and Stamatatos proposed using transfer learning in cross-domain authorship attribution [6]. Four different transformers are trained on an authorship attribution task: BERT, GPT-2, ELMo and ULMFiT. The authorship is attributed using a multi-headed classifier, meaning that an individual classifier is trained for each author. During the validation phase the classification heads of all the authors are connected to the model and the author is decided as the one with the lowest cross-entropy loss for the validation text. In a later comparative study [54], this method is referred to as per-Author Language Model (pALM), which is the abbreviation we will use as well. A similar multi-headed classifier approach was first proposed for the authorship verification task by Bagnall in 2016 [4] using a recurrent neural network (RNN) to represent the texts. Authorship verification is a task related to authorship attribution in which it is determined whether two texts are written by the same author, or by two different authors. The implementation using pre-trained transformers obtains a higher performance than a modified version of the method using RNNs, especially using BERT and ELMo [6].

Altakrori et al [1] proposed a new subtask for authorship attribution called the topic confusion task. For this experiment, a dataset is used in which all authors have written ten articles about each of four different topics. The authors are then split into two groups and the training set of group A covers topic 1, while the training set of group B covers topic 2. For the validation set this is reversed, so topic 2 for group A and topic 1 for group B. The remaining two topics are used for hyperparameter tuning before the model is tested on the validation set. The authors assume that this approach filters methods quite well in whether they classify based on the topic or the author's style. We study this so-called topic confusion task more deeply in Chapter 9.

Altakrori et al [1] use two types of methods: a feature set with an SVM and methods using a BERT-based transformer. The feature sets used in the SVM models are stylometrics, POS n-grams, character n-grams and word n-grams. Additionally, they look at the masking method [52] and combinations of the feature sets. For the BERT-based models, they look at both BERT and RoBERTa. These were implemented with either just a classification layer, similar to BERTAA but with the weights of the transformer frozen during training, or the pALM method [6].

Their results show that the n-gram-based methods outperform the BERT methods significantly. Of the n-gram-based methods masking with character or word n-grams worked best. This performance was increased slightly when combined with the features from stylometrics and POS n-grams. From the transformer-based models, the pALM method with RoBERTa worked better than the other implementations. For this model, the same group error was similar to the n-gram methods. However, the cross-group error was significantly higher, seemingly suggesting that the influence of the topic is larger on transformer-based methods [1].

2.2.3. Recent comparative studies

Two recent studies comparing performances of several different methods on a larger set of corpora have been done by Murauer et al [37] and Tyo et al [54].

Murauer et al (2021)

Murauer et al [37] compare five feature-based methods that use an SVM as a classifier with a transformer-based method using three different pre-trained language models. The five feature-based methods are character 3-grams [51], universal part-of-speech tagging [8], DT-grams [38], and two methods with a document embedding using either character 3-grams or words as tokens [19]. The pre-trained language models used were BERT [14], and two later variants of BERT, namely DistilBERT [45] and RoBERTa [33]. Here the pre-trained language models are used without any further modifications, similar to BERTAA [16].

Murauer et al evaluate the F_1-score (Chapter 8.1) of these models on six categories of corpora, three of which are cross-domain. The corpora that are not cross-domain are subdivided into standard corpora, small corpora, with only ten training texts per author, and corpora with several languages, where each author uses only one language. The cross-domain corpora are subdivided into cross-topic, cross-genre, or cross-language. Murauer et al's results show that character 3-grams outperform the other methods on all corpora except the cross-language corpora [37]. Especially for small corpora and cross-topic corpora the models using character 3-grams get a significantly higher F_1-score than the other methods. Of the pre-trained language models, RoBERTa performs the best, with a slight edge over BERT and DistilBERT. RoBERTa also had the overall best performance on the cross-language corpus [37].

Tyo et al (2022)

Tyo et al [54] compare four different methods based on macro-accuracy (Chapter 8.1): the n-gram method by Muttenthaler et al that won at PAN/CLEF in 2019 [39], prediction by partial matching (PPM) [53], BERTAA [16] and pALM [5]. Tyo finds that n-gram performs the best on 5 out of the 7 datasets, with BERTAA performing the best on the other two datasets. Tyo et al note that the corpora on which BERTAA outperforms n-grams have the largest training sets per author of the corpora. Due to large computational costs, the other two methods could only be used on smaller datasets. On the tested datasets PPM performs similarly to BERTAA, while pALM performs worse than the other methods.

Tyo et al also looked at the cross-domain and cross-topic performance of the methods, on the same corpora that Murauer et al used. Here n-gram significantly outperforms all other methods, while the remaining methods perform similarly to each other.

Comparison

Although these studies both include four of the same corpora, we cannot directly compare the results of both papers. This is due to the difference in performance measures, where one uses F_1-score and the other macro-accuracy. However, we can study the general trend in both papers, namely that methods using n-grams with an SVM still outperform other methods on most corpora, especially with a small number of training texts per author. Of the other methods, the ones using BERT-based methods perform the best, but it seems that current methods still need a significant amount of training texts per author to outperform character n-grams-based models. We thus conclude that currently SVMs with n-grams as features are state-of-the-art on almost all corpora, but BERT-based methods are the closest of the other methods and are state-of-the-art on some of the corpora with the most text per author. As a result, we will focus on these two methods in the rest of this thesis.

2.3. Forensic Science

Although the previously described research has significantly furthered the accuracy of authorship attribution methods, these results are not directly transferable to the forensic sciences. This is due to two major reasons: the datasets used in the studies and the measures used to score the performance of the methods. Datasets that are commonly used in research consist of for example news articles, blogs, movie reviews, fanfiction, books and forum posts [54]. On the other hand, this thesis research focuses on conversations, spoken or text messages. This results in individual messages with a much smaller length than those used in most authorship attribution research. Additionally, classification in computational authorship attribution studies is performed without a measure for the strength of the evidence that the appointed author has written the text. This makes it impossible for judges to use these results in their verdicts.

In his 2022 book, Grant [20] describes the history and current state of forensic authorship attribution, while comparing it to other forensic sciences. In the forensic sciences, there has been a push to quantify the strength of evidence in likelihood ratios. A likelihood ratio, also called a Bayes factor, is a measure of the strength of evidence, which we study in depth in Chapter 6.

The usage of likelihood ratios in authorship attribution generalizes the way evidence is measured with a variety of other forensic disciplines, as it is also common practice in fields like DNA analysis and automatic speaker recognition. Some research has been published focusing on determining likelihood ratios for this field [23, 25]. Still, no standardized procedure that is accepted in British courts has arisen from this [20]. At the Netherlands Forensic Institute, likelihood ratios are used to quantify the evidence of authorship attribution in court cases, but no methodology of their method has previously been published.

Juola [25] proposes a procedure in which the authorship of a questioned document by a certain suspect is determined. This procedure is based on collecting reference material of the suspected author and 10 random authors and comparing these texts to the questioned document on five feature sets. These feature sets consist of words used, word length, character 4-grams, most common words and punctuation. For each of these categories and each author, a distance between the questioned and known documents is determined based on the cosine distance. The distances of the 11 authors are then ranked per category, with the lowest distance being given 1 point and the highest 11. By summing these scores for the suspect a total score is determined, which is compared to reference data to estimate a likelihood ratio. In our view, this method is an important first publication focusing on likelihood ratios in the field of authorship attribution but also leaves some room for improvement. For example, the five feature sets are dependent on each other and the method of transforming distances to a ranked sum leaves information unused.

More studies have been done into likelihood ratios for the related authorship verification, for example by Ishihara [23] and Sergidou [48]. Due to the difference in tasks, these methods are not directly transferable. In a previous Master's thesis at the Netherlands Forensic Institute Scheijen [47] focused mainly on the verification task, but also included some research into likelihood ratios for the attribution task. In this study, kernel density estimation was used to transform a feature set of frequent words into likelihood ratios. This is also called a feature-based likelihood ratio system. Due to unstable covariance matrices for larger feature sets, the number of frequent words used was limited to only 20 words. As a result, the log-likelihood ratio cost of her results was quite high, around 0.6. This was an important contribution to the field of forensic authorship attribution, as it is the only feature-based likelihood ratio system for authorship attribution that has currently been proposed.

2.4. Authorship Attribution in Dutch

Most research in authorship attribution has been done on corpora of English text. Some papers have previously been published focusing specifically on the Dutch language [3, 26, 31, 34, 56]. As the last of these studies was published in 2011 they are not up to date concerning recent state-of-the-art methods. The corpora studied in the majority of these studies consisted of essays written by Dutch [3, 26, 56] or Flemish [34] students. The remaining study used a corpus of Dutch newsgroup messages, a type of forum [31]. The most recent paper focused on Dutch authorship attribution, by Luyckx and Daelemans [34], concluded that the features that performed the best were character n-grams. This corresponds to the results of studies on English corpora around that time [24, 50]. This suggests that the performance of this method might be transferable to the Dutch language. The previously mentioned thesis by Scheijen [47] also covers a Dutch corpus, consisting of transcribed speech, created at the Netherlands Forensic Institute, FRIDA [55]. As the focus of her research was on the verification task, no comparison of several methods on the attribution task was performed. The features used for authorship attribution were the 20 most frequently used words.

3
Corpora

In this chapter, we will describe the three corpora, also called datasets, used in this thesis research, FRIDA, RFM and abc_nl1.

3.1. FRIDA

FRIDA [55] is a dataset created at the Netherlands Forensic Institute consisting of phone calls between individuals. The participants were paid to have 8 phone conversations with a duration of 5 minutes. From the participants, pairs were made such that each participant always conversed with the same other participant, which we will call their conversation partner, or simply partner. The original goal of this dataset was the usage for validating automatic speaker recognition models. Later, all texts were transcribed by hand, leading to a dataset that can be used in authorship attribution.

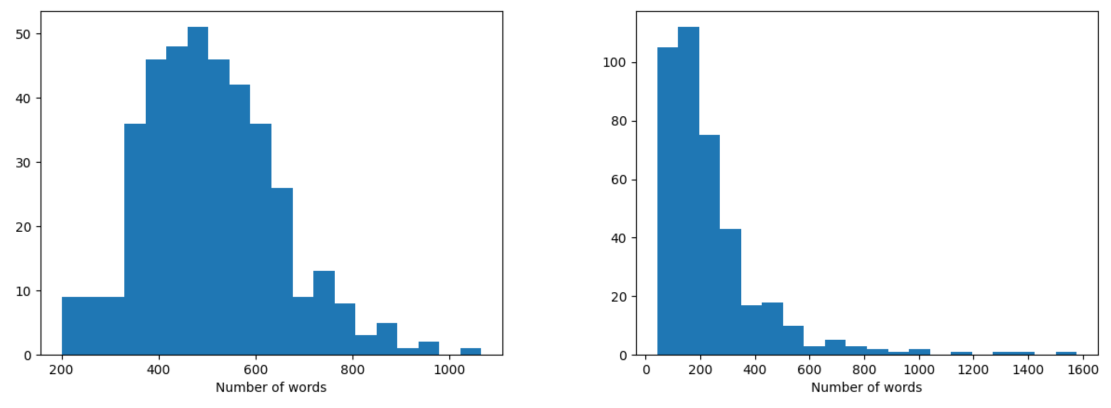
The transcription results in 8 conversation halves per speaker, who we will also call the author. In total FRIDA contains 224 authors, but in our study, we use just the first 50 of these authors. This number is chosen as it represents a common size of a background population in forensic authorship attribution cases. The average number of words per text in FRIDA is 508.6. In figure 3.1a a histogram of the number of words per text in FRIDA is shown.

3.2. RFM

Real Forensic Messages (RFM) is, as the name suggests, a dataset created from real chat messages used in forensic case work. We cannot go into depth about the creation or the contents of this dataset, but we will give statistics about the average text length, the number of authors and the number of texts used per author.

We look at messages between 25 author pairs, giving us 50 authors in total. We divide the time into blocks of two hours and merge all messages of one author per time chunk into one conversation half. We then randomly selected 8 conversation halves from all conversation blocks in which both the author and their partner sent at least 10 messages. This gives us a mean word count of 238.7 per conversation half, while the median is 185. In Figure 3.1b a histogram of the number of words per text of RFM is shown. We will also investigate the impact of text length and the number of texts per author on RFM, by creating altered versions of RFM with different parameters. When using these altered versions we will mention their specific properties.

In the histograms containing the number of words in FRIDA and RFM, we notice a significantly different distribution between the two datasets. This is caused by the way they are created. As each conversation in FRIDA is exactly 5 minutes long, the number of words per half of the conversations is much more normally distributed compared to the number of words per text in RFM. In RFM, a text consists of all messages sent by a single author in a space of 2 hours if at least 10 messages were sent in this period it can contain significant outliers. We notice that the longest text consists of more than 1500 words in a single text, which is more than six times the average text length. This leads to the right-skewed distribution of text lengths we see in Figure 3.1b.



(a) FRIDA **(b)** RFM

Figure 3.1: Histogram of the number of words per text in the datasets FRIDA and RFM.

3.3. abc_nl1

The abc_nl1 corpus [3] was created in 1999 at the Radboud University to be used for authorship attribution research. 8 students, 4 in their first year and 4 in their final year were paid to write three argumentative, three descriptive and three fictive texts. These topics were fixed, such that each author wrote about the same topics. Due to issues with accessing the original dataset, we used a reduced version of this dataset with only 6 texts per participant[1]. The remaining texts cover the following topics: argumentative texts about the television show ‘Big Brother’ and the health risks of smoking, descriptive texts about football and a book review and fiction texts about Little Red Riding Hood and a chivalric romance. A second change in the reduced version of this dataset is that all text is written in lowercase. We assume that this change has little impact on the attribution of texts, as the texts in this dataset generally consist of correct Dutch. This means that capitalization is only used at the start of sentences and for proper nouns. The start of sentences is also signified by preceding punctuation marks, so information about the frequency of using certain words at the start of sentences can still be captured. Secondly, if all authors had capitalized all proper nouns, which we expect in essays, no information about authorship was present in the capitalization.

We utilize the abc_nl1 corpus solely for our third research aim, the study of the relationship between the impact of topic and conversation on authorship attribution. As abc_nl1 is a topic-controlled corpus, it can be used to study the topic impact of various authorship attribution methods. We do not use abc_nl1 when computing our other results as it is not a forensically relevant corpus.

[1]This dataset can be found as problem M at: `https://www.mathcs.duq.edu/~juola/authorship_materials2`

4 Feature-based Authorship Attribution

In this chapter, we will cover necessary background knowledge from authorship attribution using feature vectors. We will include an overview of commonly used features like character, word and part-of-speech n-grams. We will also cover techniques to classify the resulting feature vectors, with a focus on Support Vector Machines. Then we will describe the feature-based authorship attribution model we used in this thesis.

4.1. Features

Many different features can be used to attribute texts to authors. In this section, we will give a short overview of those most commonly used in authorship attribution [50].

4.1.1. Summary statistics

The earliest authorship attribution methods were based on summary statistics. These features included for example the average character length of words or the average number of words per sentence. In calculating these averages, these methods disregard a lot of information and are therefore not suited to differentiate between large numbers of authors.

4.1.2. Function words

A different method is to use a curated list of function words as features. Function words are words with little lexical meaning, like articles, pronouns and auxiliary verbs. Using function words as features has the advantage that little to no topic is captured in these words, meaning that the topic of a text cannot influence its attribution. A drawback is that a curated list has to be created and this list is language-dependent, meaning that it cannot be easily used on texts in a different language. Alternatively, the most common words in a corpus could be assumed to be function words and require less curation.

4.1.3. Word n-grams

Another common way of extracting features from a text is the use of n-grams. An n-gram is a set of $n \in \mathbb{N}$ consecutive tokens taken from a list. In the context of authorship attribution, these tokens could for example be characters or words. For example, the Dutch sentence “Hij gaat naar school.” contains the word uni-, bi- and tri-grams shown in the table below.

Word unigrams	[“Hij”, “gaat”, “naar”, “school”]
Word bigrams	[“Hij gaat”, “gaat naar”, “naar school”]
Word trigrams	[“Hij gaat naar”, “gaat naar school”]

Here all words are counted instead of just the function words. Furthermore, by looking at combinations of words, uncommon sentence structures used by a specific author can be found.

4.1.4. Character n-grams

Similarly, to word n-grams, we can look at combinations of characters. If we look back at the example sentence, it contains the following character uni-, bi- and tri-grams:

Character unigrams	[“H”, “i”, “j”, “ ”, “g”, “a”, “a”, “t”, ... ,“l”, “.”]
Character bigrams	[“Hi”, “ij”, “j ”, “ g”, “ga”, “aa”, “ar”, ... , “ol”, “l.”]
Character trigrams	[“Hij”, “ij ”, “j g”, “ ga”, “gaa”, “aar”, ... , “ool”, “ol.]

Note that we include spaces in the character n-grams. Furthermore, a choice can be made to include or exclude punctuation marks and whether to view all letters as lowercase letters or to keep this distinction. Methods often include punctuation marks if they use character n-grams while excluding them when using word n-grams [39]. Because of the high number of possible word and character n-grams, especially for high values of n, the number of features can become quite large. This can be beneficial, as much information is still present, but also increases computational time.

4.1.5. POS n-grams

Part-of-speech (POS) n-grams look at the sentence structure used in texts. All words are first given a POS tag which annotates the grammatical function of each word in the sentence, like article, noun or verb. The sentence “Hij gaat naar school.” is tagged as [Pronoun, Verb, Adposition, Noun]. POS bi- or tri-grams can highlight an author’s commonly used sentence structures, which might be used with different words in different sentences.

4.1.6. Special characters

Some methods also specifically include special characters like punctuation symbols or emojis as features. This information can also be processed as a part of either word or character n-grams, but including it as a specific feature can put additional emphasis on these features.

4.2. Support vector machines

There exist several methods to use the feature vectors to classify new texts. A simple method would be to normalize the elements and calculate the average distance to the feature vectors of the training texts for each author [30]. As the authorship attribution task is a multiclass classification problem, many machine-learning classification methods can be used, including logistical regression, naive Bayes, support vector machines, decision trees and random forests [24, 29, 47, 50]. As support vector machines (SVMs) generally outperform other methods using feature vectors [47, 52, 54], we focus on implementing and improving methods using SVMs. In this section, we will give a theoretical background of the SVM, which was first proposed by Boser, Guyon and Vapnik in 1992 [9]. We start from a linearly separable binary classification problem, also called the hard margin SVM and will expand this to the non-separable setting, also called soft margin SVM, and to the multiclass setting.

4.2.1. Hard margin SVM

Suppose we have a dataset $S = (x_1, y_1), ..., (x_{N_s}, y_{N_s})$, with $x_i \in \mathbb{R}^{n_s}$ and $y_i \in \{-1, 1\}$. We first assume that this dataset is linearly separable, so there exists a hyperplane $\beta^\top x + \beta_0 = 0$ with parameters $\beta \in \mathbb{R}^{n_s}, \beta_0 \in \mathbb{R}$ such that

$$\begin{aligned} \beta^\top x_i + \beta_0 > 0,\ \forall y_i = 1, i \in 1, ..., N_s \\ \beta^\top x_i + \beta_0 < 0,\ \forall y_i = -1, i \in 1, ..., N_s. \end{aligned} \tag{4.1}$$

Often more than one hyperplane separates the two classes. We define the optimal separating hyperplane as the hyperplane for which the minimal distance, the margin M, between the dataset and the hyperplane is the largest. As the inequality in the Conditions (4.1) is strict, we know that $M > 0$. Let x_i be a point from the dataset and x a point on the hyperplane. The minimal distance between the point x_i and the hyperplane can be found by calculating the length of the projection of $x_i - x$ on to vector perpendicular to the hyperplane, which is β. So:

$$d(\beta, \beta_0, x_i) = ||\text{proj}_\beta(x_i - x)|| = ||\frac{(x_i - x)^\top \beta}{\beta^\top \beta}\beta|| = |x_i^\top \beta - x^\top \beta|\frac{||\beta||}{||\beta||^2} = \frac{|\beta^\top x_i + \beta_0|}{||\beta||} \tag{4.2}$$

To be able to express the size of M in terms of the length of β, we now rewrite the Conditions (4.1). This new condition holds for fewer hyperplanes, but we can rescale every hyperplane that satisfies Conditions (4.1) with a factor $\frac{1}{M}$ to satisfy Condition (4.3).

$$y_i(\beta^\top x_i + \beta_0) \geq 1, \forall i \in 1, ..., N_s. \tag{4.3}$$

If we substitute (4.3) into Equation (4.2) and use that $|y_i| = 1$ we get that for all $x_i \in S$

$$d(\beta, \beta_0, x_i) \geq \frac{1}{||\beta||}. \tag{4.4}$$

So by minimizing $||\beta||$ while satisfying Condition (4.3) we can find the separating hyperplane with the largest distance to all points in S. To simplify later derivations we change the optimization objective to minimizing $\frac{1}{2}||\beta||^2$, which does not change the solution. To summarize, we can formulate the hard margin support vector machine as the following optimization problem:

$$\begin{gathered}\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 \\ \text{subject to: } y_i(x_i^\top \beta + \beta_0) \geq 1, \forall i \in 1, ..., N_s.\end{gathered} \tag{4.5}$$

The optimal solution can be found using the SMO algorithm (Chapter 4.2.3). We note that for the optimal solution, there exists at least three points for which Condition (4.3) is satisfied with equality. These points are also called support vectors and have given this classifier the name support vector machine.

4.2.2. Soft margin SVM

In general, we do not know whether a hyperplane that separates S exists or not. To still be able to use SVMs in the case where this hyperplane does not exist, we introduce slack variables $\xi_i, \ i \in 1, ..., N_s$ and slightly change both the objective function and the corresponding conditions. We first modify Condition (4.3) into

$$y_i(\beta^\top x_i + \beta_0) \geq 1 - \xi_i, \ \xi_i \geq 0, \forall i \in 1, ..., N_s. \tag{4.6}$$

This means that there no longer exists a hard margin around the hyperplane that separates the two classes. Furthermore, data points may be located on the wrong side of the hyperplane. To still be able to find a hyperplane that separates the classes reasonably well we need to update the optimization objective with a cost for large values of ξ_i, to limit the number of points that cross the margin. This finally leads us to the following optimization problem:

$$\begin{gathered}\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N_s} \xi_i \\ \text{subject to: } y_i(\beta^\top x_i + \beta_0) \geq 1 - \xi_i, \xi_i \geq 0, \forall i \in 1, ..., N_s.\end{gathered} \tag{4.7}$$

Here C is a regularization parameter that can be chosen, with a default value of 1.

4.2.3. The SMO algorithm

Both the hard and soft margin SVM optimization problem can be solved by first using Lagrangian multipliers to find the Wolfe dual of this problem [21]. This dual is a quadratic programming problem that can be solved in various ways. Currently, the most used algorithm uses sequential minimal optimization (SMO), proposed by Platt in 1998 [41]. In SMO, all but two variables are assumed fixed and the optimization problem is solved for the remaining two variables. This is done iteratively until no improvement can be found. The SMO algorithm guarantees that it converges to the global optimum [41].

4.2.4. Multiclass SVM

We have described implementing a support vector machine as a binary classifier between two classes. For authorship attribution, we need to be able to classify a text between several authors. To do this, two common heuristical multiclass strategies exist for SVMs, One-vs-One classifying and One-vs-Rest classifying.

One-vs-One classifier

As the name suggests, a One-vs-One classifier handles two classes per instance. To do this, for every pair of classes an individual SVM is built using the training data of those classes, leading to $\frac{n(n-1)}{2}$ classifiers as we have n different classes, one for each author. Given a new test sample, every classifier decides which of the two classes is the most likely. In the end, the text is attributed to the class with the most votes from the individual classifiers.

One-vs-Rest classifier

One-vs-Rest classifier functions by building n classifiers, where each classifier is an SVM that compares the training data from one class to that of new class containing the training data from all other classes. With each classifier, a score is calculated for validation texts using the distance to the decision boundary. This score is negative if the validation sample is on the other side of the decision boundary compared to the training samples of the class for which the specific classifier was built. The text is then attributed to the class with the highest score.

In our implementation, we experimented with both strategies. In these experiments (Appendix C) we noticed that the One-vs-Rest classifier performed much better, leading to us continuing with this strategy. We note that the computational load of the One-vs-Rest classifier is also lower for large numbers of authors compared to the One-vs-One strategy, due to the quadratic relation between the number of classes and the number of classifiers in the One-vs-One strategy.

4.2.5. Kernels

To find non-linear hyperplanes that separate classes, one can expand the SVM with the kernel trick, to increase the dimensionality of the feature space with for example a polynomial or Gaussian kernel. In Authorship Attribution this is not commonly used, and in our experiments this also led to a deteriorated performance (Appendix C). This is likely due to the low number of training texts compared to the dimensions of the feature space. As a result, a high dimensional kernel assures that optimal separating hyperplanes can be found, but this also increases the risk of overfitting on the training data. With our limited amount of 7 training texts per author, it is not unsurprising that utilizing the kernel trick introduces significant overfitting problems.

4.3. Implementation

In the following sections, we will cover our implementation of an SVM, which is a combination of the implementation by Muttenthaler, Lucas and Amann [39] with the idea of masking from Stamatatos [51, 52].

4.3.1. Feature vector

The implementation by Muttenthaler, Lucas and Amann [39] works by averaging the results of an SVM with character n-grams as features and an SVM with word n-grams as features. As both SVM have the same architecture we will explain them at the same time, while highlighting the small differences. The model starts by creating the feature vector for all texts. For the word n-gram model, all word 1 and 2 grams are counted, while for the character n-gram model, all character 2, 3, 4 and 5 grams are counted. In the word n-gram model capitalization and punctuation are ignored, while in the character n-gram model both are included.

The feature vector is then reduced to include only the 30% most frequent n-grams over all the training texts. This is done to negate the influence of n-grams that occur infrequently. The reason that these infrequent features can be an issue is that an n-gram that occurs in only one training text would give a dimension that separates that text from all other texts. As almost any text has at least one unique character 5-gram, not excluding infrequent features could result in overfitting on the training set. This is a slight modification of Muttenthaler et al, where the 50% most frequent features were included. The remaining features are first divided by the total number of words or characters in the text and then scaled by dividing the count by the maximal count of that feature over all texts. This is to ensure all features have a value between 0 and 1 and therefore prevent that more frequent n-grams have a higher weight than less frequent n-grams.

All the permutations to the feature vectors are also performed on the validation texts, using the transformations as computed on the training texts. As a result, on the validation texts, it is for example not necessary that the value for each feature is less or equal to 1, as the frequency of a feature on a validation set might be higher than the highest frequency of that feature in the training set.

In general, the number of unique features m_f is much larger than the number of training texts N_s, $m_f \gg N_s$. Let M be the training matrix, where each row is the transposed feature vector of a training text, leading to the dimensions of M being N_s by m_f. Using the singular value decomposition (SVD) of M we can reduce the dimensionality of the feature vectors to N_s. We will show that by doing this we can lower the computational cost without impacting the performance of the SVM.

For every real-valued matrix with more columns than rows, there exists a singular value decomposition $M = U\Sigma V^\top$, with Σ a diagonal matrix of dimensions $N_s \times N_s$, U an orthogonal matrix, meaning $UU^\top = U^\top U = I$, of dimensions $N_s \times N_s$ and V a semi-orthogonal matrix of dimensions $m_f \times N_s$, with $V^\top V = I$.

We can then compute a lower dimension version of M, $\tilde{M} = MV$ of dimensions $N_s \times N_s$. The transposed feature vectors of $x_i^\top$ are the rows of M. So $\tilde{x_i} = V^\top x_i$ are column vectors of $\tilde{M}$, with length N_s. We now want to show that the optimal SVM on the data $(x_1, y_1), ..., (x_{N_s}, y_{N_s})$ gives the same results as creating the optimal SVM with the reduced feature vectors $\tilde{x_1}, ..., \tilde{x}_{N_s}$. To do this we will show that Formulation (4.7) can be reformulated as

$$\begin{gathered}\min_{\tilde{\beta}, \beta_0} \frac{1}{2}||\tilde{\beta}||^2 + C\sum_{i=1}^{N_s} \xi_i \\ \text{subject to: } y_i(\tilde{\beta}^\top \tilde{x_i} + \beta_0) \geq 1 - \xi_i, \xi_i \geq 0, \forall i \in 1, ..., N_s.\end{gathered} \tag{4.8}$$

For this proof, we first show that the optimal solution for Formulation (4.8) has a corresponding solution for Formulation (4.7) with the same objective value. Then we will show that the optimal Formulation (4.7) has a corresponding solution for Formulation (4.8) with the same objective value. From these two parts, we can then conclude that the corresponding solution of an optimal solution is also optimal and both problems have the same objective value for their optimal solution.

We start by showing (4.8) to (4.7). Suppose $\tilde{\beta}$ is the optimal solution to Formulation (4.8). We need to show that we can find β such that $||\tilde{\beta}|| = ||\beta||$ and $\tilde{\beta}^\top \tilde{x_i} = \beta^\top x_i$. Take $\beta = V\tilde{\beta}$. We then find

$$\tilde{\beta}^\top \tilde{x_i} = \tilde{\beta}^\top V^\top x_i = (V\tilde{\beta})^\top x_i = \beta^\top x_i, \quad \forall I \in 1, ..., N_s \tag{4.9}$$

and

$$||\beta||^2 = ||V\tilde{\beta}||^2 = (V\tilde{\beta})^\top V\tilde{\beta} = \tilde{\beta}^\top V^\top V\tilde{\beta} = \tilde{\beta}^\top \tilde{\beta} = ||\tilde{\beta}||^2. \tag{4.10}$$

This shows that $\beta = V\tilde{\beta}$ satisfies the conditions of Formulation (4.7) and has the same objective value. Note that we have kept the values of β_0 and ξ_i the same between both formulations.

We then show (4.7) to (4.8), for which the proof is significantly more elaborate. Suppose β is an optimal solution to Formulation (4.7).

We first note that as $x_i^\top$ is a row of M, it is equal to the ith row of $U\Sigma V^\top$, which is equal to $U_i\Sigma V^\top$, where U_i is the ith row of U. Then as $\Sigma^\top = \Sigma$ we have $x_i = V\Sigma U_i^\top$, which leads to

$$\tilde{x_i} = V^\top x_i = V^\top V\Sigma U_i^\top = \Sigma U_i^\top. \tag{4.11}$$

From this we find

$$x_i = V\Sigma U_i^\top = V\tilde{x_i} \left(= VV^\top x_i\right). \tag{4.12}$$

Secondly, we establish that $VV^\top$ is an orthogonal projection. This can be seen from

$$(VV^\top)^2 = VV^\top VV^\top = V(V^\top V)V^\top = VIV^\top = VV^\top \tag{4.13}$$

and

$$(VV^\top)^\top = VV^\top. \tag{4.14}$$

We can now use the Cauchy-Schwarz inequality to find

$$||VV^\top \beta||^2 = \beta^\top VV^\top VV^\top \beta = \beta^\top VV^\top \beta = \langle \beta, VV^\top \beta \rangle \leq ||\beta|| \cdot ||VV^\top \beta||. \tag{4.15}$$

By dividing both sides by $||VV^\top \beta||$ we get

$$||VV^\top \beta|| \leq ||\beta||. \tag{4.16}$$

We additionally have that, using Equation (4.12),

$$(VV^\top \beta)^\top x_i = \beta^\top VV^\top x_i = \beta^\top x_i. \tag{4.17}$$

So we see that $\beta^* = VV^\top \beta$ satisfies the conditions of Formulation (4.7) and has a lower or equal objective value than β. Therefore, as β was an optimal solution to Formulation (4.7), β^* must also be an optimal solution to Formulation (4.7). Thus we have,

$$||\beta^*|| = ||\beta|| \tag{4.18}$$

It now suffices to show that we can find $\tilde{\beta}$ such that $||\beta|| = ||\tilde{\beta}||$ and $(\beta)^\top x_i = \tilde{\beta}^\top \tilde{x_i}$. We can choose $\tilde{\beta} = V^\top \beta$. Using this combined with the previous result that $x_i = V\tilde{x_i}$, we get

$$\beta^\top x_i = \beta^\top V \tilde{x_i} = (V^\top \beta)^\top \tilde{x_i} = \tilde{\beta}^\top \tilde{x_i}, \quad \forall I \in 1, ..., N_s \tag{4.19}$$

We note that for β^* we have

$$VV^\top \beta^* = VV^\top VV^\top \beta = VV^\top \beta = \beta^*. \tag{4.20}$$

Finally, we find

$$||\tilde{\beta}||^2 = ||V^\top \beta||^2 = \beta^\top VV^\top \beta = \beta^\top VV^\top VV^\top \beta = (\beta^*)^\top \beta^* = ||\beta^*||^2 = ||\beta||^2. \tag{4.21}$$

This shows that $\tilde{\beta} = V^\top \beta$ satisfies the conditions of Formulation (4.8) and has the same objective value. As we have now shown this both ways, we can conclude that if we find an optimal solution $\tilde{\beta}$ for Formulation (4.8), $\beta = V\tilde{\beta}$ must also be optimal for Formulation (4.7) and vice versa. Therefore, we can reduce the feature vectors to length N_s, while keeping the same classification results.

Similarly, feature vectors created for texts in the validation set can also be reduced in dimension using $\tilde{x_v} = V^\top x_v$ for x_v a feature vector of a text in the validation set and V as computed using the training data. These feature vectors can then be classified using the SVM created on the reduced space. This reduction of feature space significantly speeds up the computation of the SVMs and as it does not alter the attribution this speed increase comes free of cost. For our problem, this reduces the dimension from the order of 10,000 features to 350 features.

4.3.2. SVM

The feature vectors are then used to train a support vector machine. Here the support vector classifier of scikit learn is used, which uses the libsvm [12] implementation of the SMO algorithm. The default value of $C = 1$ is used for the regularization parameter in the support vector classifier. The results of the character and word model are combined by averaging the resulting softmax probabilities for each class of both models and then attributing them to the class with the highest probability. We will study both the attribution performance of the combined model and the individual word and character models in the results (Chapter 10).

4.4. Masking

To reduce the impact of the topic of a text on its attribution the idea of masking can be used. This idea, of which the usage in authorship attribution was first proposed by Stamatatos in 2017 [51, 52], uses a list of the most frequent words to mask infrequent words. The idea is simple, very general words like “is”, “he”” and “going” are used more often than topic-related words like “football” and “paint”. Therefore, if we only look at the N_f most frequent words this might hide the topic of a text and leave us with the usages of general words, in which style still might be present. Stamatatos [51, 52] replaces words outside of the list of most frequent words with either one or more asterisks. In single masking, a single asterisk is used to mask these words while in multiple masking the same number of asterisks as the number of letters in the word is used. The idea of masking is quite similar to the use of function words as a feature but has the added value that we can still use character n-grams as features.

In our implementation of masking, we use the hashtag sign # instead of the asterisk *, as the asterisk is already used to signify certain speech properties in FRIDA. Furthermore, we noticed no significant performance difference between single masking and multiple masking, as can be seen in Appendix C. Therefore we used single masking for computational efficiency.

Lastly, we did some experiments using masking in combination with BERT-based models (Chapter 5). This led to a worse performance (Appendix C) and, as a result, we dropped this analysis from our results.

Below an example sentence from FRIDA is given, together with how the sentence is modified when masking is performed with $N_f \in \{5000, 1000, 200, 100\}$. In this thesis, we sometimes use $N_f = \infty$ to specify that no masking is being performed and all texts are kept the same.

$N_f = \infty$	Uh over vliegtuigen dingen allemaal dat soort dingen
$N_f = 5000$	Uh over vliegtuigen dingen allemaal dat soort dingen
$N_f = 1000$	Uh over # dingen allemaal dat soort dingen
$N_f = 200$	Uh over # dingen allemaal dat # dingen
$N_f = 100$	Uh over # # # dat # #

4.5. Background vocabulary

Masking uses a list of the most frequent words as a background vocabulary. Here an important choice is whether a generic Dutch frequency list is used or a new frequency list is made for this corpus specifically. We chose to create corpus-specific frequency lists for the FRIDA and RFM corpora and used a generic Dutch frequency list for abc_nl1. This has two reasons: the usage of slang and the size of the corpora.

Most existing frequency lists of Dutch are created on standard Dutch, meaning that no slang is included. In real forensic datasets, a lot of slang is present and useful for authorship attribution. Consequently, information is lost by excluding these words. As abc_nl1 consists of formal essays, it does not include slang.

The second reason for not using a corpus-specific frequency list for abc_nl1 is its size. It consists of 48 texts with an average length of 911 words, which is not big enough to be representative. The fixed topics strengthen this effect. For example, the 81st most common word in abc_nl1 is “Roodkapje” (Little Red Riding Hood), which is not a common word outside of fairy tales. The frequently list we use for abc_nl1 is the SUBTLEX-NL [28] frequency list [1], created from dutch movie subtitles, as it is the most commonly used Dutch frequency list available to us. For the other two datasets, the number of authors is significantly larger, ensuring a more representative number of texts. Furthermore, the number of authors is larger than the set of authors we use for authorship attribution. This ensures that the vocabulary of the authors taken into account is diluted by the vocabulary of all other authors in the set.

For FRIDA we created a frequency list by counting the frequency in all texts in the dataset. For RFM we did the same with a representative sample from the full dataset from which RFM was created.

[1]The list can be downloaded at `https://osf.io/3d8cx/`

4.6. Baseline

To compare our models with the current baseline, we also implemented the non-manual part of what is currently used at the NFI. To simplify our code base we implemented the baseline as a modified version of the previously introduced SVM model. This version uses a logistic regression classifier, word n-grams as features and masking with $N_f = 100$. Two additional modifications are made. Firstly, we only look at the word 1-grams, and exclude the word two grams. Secondly, we no longer cut off the 70 % least frequent features, such that all the 100 most frequent words are used as features.

5 BERT

The deep learning architecture **transformers** was first introduced by researchers from Google in 2017 [57]. Before transformers, architectures like recurrent neural networks (RNNs) and convolutional neural networks (CNNs) struggled with capturing long-range dependencies and context in data effectively. Due to the use of a softmax-based attention mechanism, transformer-based models can selectively focus on different parts of the input data. This attention-based approach handles sequences more effectively, making it particularly suitable for natural language processing tasks that require understanding context and relationships between words. Several models have been based on this improvement, e.g. GPT [42] and BERT [14]. Due to its good performance on authorship attribution tasks compared to other transformer-based models [6] we will focus on BERT and explain its architecture to illustrate the working of a transformer-based model. Then, we will cover how BERT can be utilized for the authorship attribution task and introduce the two BERT-based models pre-trained on Dutch data that are used in our research.

5.1. Model architecture

Bidirectional Encoder Representations from Transformers (BERT) uses a modified version of the encoder of the full transformer architecture. In Figure 5.1 the architecture of BERT is illustrated.

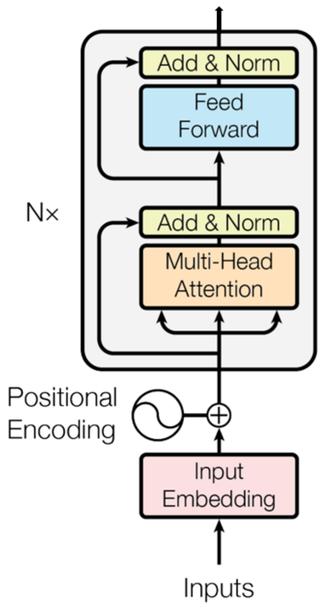


Figure 5.1: Illustration of the encoding part of the transformer architecture used in BERT, modified from Vaswani et al [57].

To use a deep learning model on text, the text first has to be converted to numerical vectors. This is done in two steps. First, the text is cut into smaller parts, tokens, from a specific vocabulary. Subsequently, the encoding of that token is calculated using the token, its segment and its position. The resulting encoding matrix is repetitively put through a multi-head attention layer and a feed-forward neural network layer to get the final hidden state. In the following subsections we will cover these steps in depth.

5.1.1. Tokenization

To tokenize inputted text BERT uses WordPiece embeddings [60]. A vocabulary of 30,000 wordpieces is created starting from individual characters by combining them into commonly used parts of words and words. As a result, the meaning of most words can be captured in tokens while keeping the vocabulary as small as possible. This vocabulary is then used to tokenize texts into a vector of integers, as shown below on the example sentence “She is metalworking in Delft.”

Text		She	is	metalworking		in	Delft		.	
Wordpieces	[CLS]	she	is	metal	##working	in	del	##ft	.	[SEP]
Tokens	101	2016	2003	3384	21398	1999	3972	6199	1012	102

We notice that short words like “she”, “is” and “in” are included in the vocabulary and therefore have their own token. Lesser used or longer words like “Delft” and “metalworking” are split into two tokens, were the WordPiece of the second token starts with ## to signify it is not at the start of a word, so “##working” and “working” correspond to different tokens. Due to this method of splitting words into chunks, the vocabulary can be kept lower, while still having the meaning of longer words being encoded in their chunks. For example, the words “woodworking” and “overworking” are both also outside the vocabulary and thus tokenized using the “##working” token. As this piece has the same meaning in all three words it allows us to convey a similar message with a smaller vocabulary size.

There are two types of WordPiece tokenization models, a cased version that includes capitalization and an uncased version that ignores capitalization. As a result of this, both cased and uncased BERT models have been released. The example above uses uncased tokenization. When tokenizing a sentence in BERT, a [CLS] (classifier) and [SEP] (separation) token are always added to the start and end of the token sequence, respectively. Other special tokens are the [UNK] token for unknown elements, e.g. Chinese characters, the [MASK] token used to mask other tokens during pre-training and the [PAD] (padding) token used to extend all sequences to length 512, which is the maximal input length BERT can handle.

5.1.2. Encoding

To be able to perform computations on the token sequences we want to transform their values from integers to vectors of dimension d_{model} consisting of floats. As we do this for each of the 512 tokens our resulting encoding matrix E has dimensions $512 \times d_{\text{model}}$. Here each row corresponds to the encoding of a single token. The encoding of each token is computed as the sum of the token encoding (E^τ), the positional encoding (E^p) and the sentence encoding (E^s).

The token encoding is computed using a $30,000 \times d_{\text{model}}$ matrix I, where each row corresponds to the token encoding of an individual token in the vocabulary. Let τ be a sequence of tokens representing a text, also called a token vector, then the token encoding of τ, $E^\tau = \begin{bmatrix} I_{\tau_1}^\top & I_{\tau_2}^\top & \dots & I_{\tau_{512}}^\top \end{bmatrix}^\top$. Here I_{τ_1} corresponds to the τ_1^{th} row of matrix I.

The positional encoding matrix E^p is calculated independently of the values of the tokens and is therefore constant. It is calculated with the following equations:

$$\begin{aligned} E^p_{i,2j} &= \sin\left(i/10000^{2j/d_{\text{model}}}\right) \\ E^p_{i,2j+1} &= \cos\left(i/10000^{2j/d_{\text{model}}}\right) \end{aligned} \tag{5.1}$$

Lastly, a sentence encoding matrix is added. The sentence encoding is only relevant for the next sentence prediction task, one of the two tasks used in the pre-training of BERT. It consists of two types of rows of dimension d_{model}, E^s_A for the first sentence and E^s_B for the second sentence. We then have

$E^s = \begin{bmatrix} E_A^s \\ \dots \\ E_A^s \\ E_B^s \\ \dots \\ E_B^s \end{bmatrix}$, where the change of the row vector happens at the end of the first sentence and is signified by the [SEP] token. In our application, we use only E_A^s, leading to a matrix E^s with constant rows.

Finally, we can calculate the complete encoding matrix E as follows,

$$E = E^\tau + E^p + E^s. \tag{5.2}$$

5.1.3. Multi-head attention

We now use this encoding matrix E to calculate the resulting hidden states. It first passes through a multi-head attention layer, consisting of h attention heads, where we choose h to be a divisor of d_{model}. For each attention head i, $i \in 1, ..., h$, a Query, Key and Value matrix is calculated using weight matrices: $Q_i = EW_i^Q$, $K_i = EW_i^K$ and $V_i = EW_i^V$. The dimensions of the weight matrices are $d_{\text{model}} \times d_k$, with $d_k = d_{\text{model}}/h$. Therefore the resulting Q, K and V matrices have dimensions $512 \times d_k$. From this, we calculate the attention matrix

$$\text{Attention}_i\left(Q_i, K_i, V_i\right) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i. \tag{5.3}$$

Here the softmax function is used on the rows of the matrix $\frac{Q_i K_i^T}{\sqrt{d_k}}$.

The resulting attention matrices are then concatenated into a $512 \times d_{\text{model}}$ matrix and multiplied by another weight matrix W_o of dimensions $d_{\text{model}} \times d_{\text{model}}$ to get

$$O = \text{concat}\Big(\text{Attention}_1\left(Q_1, K_1, V_1\right), ..., \text{Attention}_h\left(Q_h, K_h, V_h\right)\Big) W_o. \tag{5.4}$$

Since d_k is chosen such that $d_k = d_{\text{model}}/h$ concatenating the outputs of the h attention heads results in a matrix with dimensions $512 \times d_{\text{model}}$, which is the same as the encoding matrix E. Lastly, the original matrix E is added to O and the result is normalized using row-based z-score normalization.

5.1.4. Feed-forward neural network

After passing through the attention layer each row of the resulting matrix is passed through a feed-forward neural network with one hidden layer of dimension d_{hidden} and a ReLu ($f(x) = \max(0, x)$) activation function. This corresponds to the output being computed as

$$F_i = \max(0, O_i W_1 + b_1) W_2 + b_2, \forall i \in 1, ..., 512. \tag{5.5}$$

With O_i being the columns of O and F_i the columns of the resulting matrix F. The W_1 and W_2 are weight matrices of dimensions $d_{\text{model}} \times d_{\text{hidden}}$ and $d_{\text{hidden}} \times d_{\text{model}}$, respectively. b_1 and b_2 are bias vectors of size d_{hidden} and d_{model}. In practice d_{hidden} is chosen to be equal to $4 \cdot d_{\text{model}}$.

After passing through the feed-forward layer the matrix O is added to F and again z-score normalization is performed on the rows. The combination of passing the matrix iteratively through a multi-head attention layer and a feed-forward neural network layer is repeated N times. For each iteration different weight matrices are used. The resulting output matrix 512 by d_{model} matrix F after N iterations is called the final hidden state. The first row of the final hidden state, corresponding with the [CLS] token, is often used for inference about the whole sentence.

We note that except for the multi-head attention layer, all matrix manipulations are performed purely on the individual rows, meaning that only the information of one specific token is used. Only during the multi-head attention layer, the context of the adjacent rows is used during matrix multiplication.

Two different models were proposed in the original paper [14], a base version with model dimension $d_{\text{model}} = 768$ and number of iterations $N = 12$ and a large version with $d_{\text{model}} = 1024$ and $N = 24$. For both versions $h = 8$ is used. Due to computational restrictions, we will use the base versions of all BERT-based models. In model benchmark tests it was shown that the large version slightly outperforms the base version [14], but in practice, the base versions are used more often due to the combination of satisfactory performance and memory restrictions.

5.1.5. Pre-training

The original BERT model was pre-trained on two tasks, masked language modelling (MLM) and next sentence prediction (NSP). In the masked language modelling task 15% of all tokens are randomly replaced with a [MASK] token. The model must then predict the original token at the location of all [MASK] tokens. This is done using a classification head on the final hidden state at the [MASK] token which maps to the entire vocabulary. A classification head is a dense neural network layer, in this case from d_{model} nodes to 30,000 nodes. The resulting scores are converted into probabilities using a softmax function. From these scores, the cross-entropy loss is computed and backpropagated through the BERT model.

In the NSP task, the model is given two sentences and it must predict whether or not they are consecutive sentences from the same text. During the pre-training on this task, a classification head converts the final hidden state of the [CLS] token to a vector of size two, from which probabilities are again computed using a softmax function and the cross-entropy loss is backpropagated.

5.2. BERT models for authorship attribution

In previous research, transformer-based models have been utilized for two types of authorship attribution methods. The first method consists of fitting a classification head onto the final hidden state [16], and the second method uses a multiheaded classifier [5, 6]. Overview studies have shown better performance for the first method [37, 54] and therefore we chose to focus on this method.

This method, called BERTAA [16], adds a classification head on top of the BERT model, that maps the vector of size $d_{\text{model}} = 768$ from the final hidden state of [CLS] token to a vector r of size n, with n being the number of authors. This model is then finetuned on the training set. During finetuning, training texts written by the n authors are iteratively passed through the model and the loss is calculated using cross-entropy loss. The softmax probability of the correct classification is first calculated to compute the cross-entropy loss. Suppose a training text t is written by author a_j, $j \in 1, ..., n$. The probability of attributing the text to author a_i, where $i \in 1, ..., n$, is defined as

$$p_{a_i} = \frac{e^{r_i}}{\sum_{k=1}^{n} e^{r_k}}. \tag{5.6}$$

This is called the softmax function and guarantees that all probabilities are non-negative and their sum is equal to 1.

The cross-entropy loss is defined as

$$L(t, p) = -\log\left(p_{a_t}\right). \tag{5.7}$$

This results in a loss of 0 if the probability for the true author is 1 and this loss increases as the probability decreases. The same cross-entropy loss is used during the pre-training of BERT models on the NSP and MLM tasks. This loss is then backpropagated through the entire BERTAA model to update all weight matrices.

5.3. Subverting the token limit

As the input of BERT models is limited to 512 tokens, long messages cannot be parsed as one message. To avoid this limitation when classifying long texts, we have developed two new strategies. We will first introduce the original method used in BERTAA [16] before introducing our proposals.

5.3.1. Truncation

The original method truncates the tokenized message to satisfy the limit of 512 tokens, so only the first 512 tokens of the message are considered. While this method is simple in both implementation and computation it disregards a significant amount of information from longer texts. We will refer to this method as the **truncation** method.

5.3.2. Averaging classifications

In our first proposal, the text is subdivided into message parts of 512 tokens. To ensure no context is missed, the parts are chosen such that they overlap. Let τ be the vector of tokens for a message t, with $|\tau| > 512$. We first strip the [CLS] and [SEP] tokens from this vector to create τ^*. We now subdivide

τ^* into two non-overlapping vectors of 255 tokens, $\tau_1^*, ..., \tau_m^*$, $m = \text{ceil}(|\tau|/255)$. To the last vector, τ_m^* padding tokens, [PAD], are added such that it is of length 255. Now we create a set of token vectors T^{**} as follows: $\tau_i^{**} = \text{concat}([\text{CLS}], \tau_i^*, \tau_{i+1}^*, [\text{SEP}])$, for $i \in 1, ..., m-1$. T^{**} is now a set of $m-1$ vectors of tokens of length 512, with 255 tokens overlapping with the previous vector and 255 with the next vector. After subdividing the texts we use our classification model on each of the vectors, and calculate the softmax probabilities for each author, as in Equation (5.6). We then take the average of the probabilities for each vector and attribute the text to the author with the highest average probability. We summarize this computation in Equation (5.8).

$$a_a = \underset{a_j \in A}{\text{arg max}} \frac{1}{m-1} \sum_{i=1}^{m-1} \frac{e^{r_j^i}}{\sum_{k=1}^{n} e^{r_k^i}} \tag{5.8}$$

Here a_a is the author to which the text is attributed and r_j^i is the value from the classification layer of the BERT model for token vector τ_i^{**} and author a_j.

During finetuning with this method, the cross-entropy loss is also computed using these average probabilities. We will refer to this method as the **averaging** method.

5.3.3. Mean pooling before classification layer

In the second method we propose, we use the same subdivisions into token vectors as in our first proposal. Instead of averaging the resulting probabilities we now average the final hidden states at the [CLS] positions of the message parts. A layer that averages these vectors is also called a mean pooling layer. The resulting vector is then the input of the classification head and the computations continue similar to the original method. We will refer to this method as the **mean pooling** method.

A drawback of the overlapping message chunks in both the averaging and the mean pooling methods is that the first and last tokens of the message are considered only once, while all other tokens are considered twice. As a result, they have a lesser impact on the overall classification compared to the middle sections.

5.4. Dutch BERT models

After the introduction of BERT models, several Dutch BERT-based models were also created by pre-training this architecture on Dutch corpora. In this thesis, we have studied the performance of the two most used Dutch BERT-based models, BERTje [59] and RobBERT [13].

BERTje uses the same architecture as BERT but was pre-trained using Dutch corpora. The training corpus included Wikipedia articles, news articles and historical fiction [59]. The pre-training procedure was also altered slightly from the original BERT paper. The NSP task was replaced with a sentence ordering prediction (SOP) task, where the model classifies whether the order of two consecutive sentences is correct or inversed. Furthermore, to increase the difficulty of the MLM task all WordPieces of the same word were masked if one of the pieces was masked.

RobBERT is pre-trained with the same strategy as RoBERTa [33], an english BERT-based model pre-trained on just the MLM task. For pre-training RobBERT the Dutch part of the OSCAR corpus was used, consisting of crawled web data [13].

6 Likelihood Ratio Framework

In the forensic sciences, evidence is quantified using likelihood ratios (LRs). In this chapter, we will cover the theoretical bases behind the likelihood ratio. Furthermore, we will give a short introduction to the common source and the specific source problems using authorship identification as our leading example.

6.1. Common Source problem

Suppose threats are made to someone from two different social media accounts. These two messages could be sent by either one person using both accounts or by two different people. This is an example of the Common Source (CS) problem in forensic identification. The problem does not focus on who sent the threats, but whether or not they were sent by the same person. To generalize this problem, we formulate two hypotheses, the prosecution hypothesis (H_p) and the defence hypothesis (H_d) [40]. Suppose we have two sets of unknown source evidence, e_{u_1} and e_{u_2}, then

H_p : The two sets of unknown source evidence both originate from the same unknown source.

H_d : The two sets of unknown source evidence originate from two different unknown sources.

6.2. Specific Source problem

The Specific Source (SS) problem is defined with two slightly different hypotheses. We return to the example of online threats to illustrate these hypotheses. Suppose that one social media account has made a threat to a person and it is suspected that the author of the threat is a_s. Then, we can examine previous social media messages which are known to be written by a_s and social media messages from the background population to determine if a_s wrote the threats. This is called a specific source problem, as we want to decide whether a piece of unknown source evidence originates from a specific suspected source or another source in the background population. We have one set of unknown source evidence e_u, control material e_s known to come from the suspected source and control material e_b known to come from various sources in the background population. The hypotheses are now defined as follows [40]:

H_p : The unknown source evidence e_u and the specific source evidence e_s both originate from the specific source.

H_d : The unknown source evidence e_u does not originate from the specific source, but from some other source in the alternative source population.

6.3. The CS and SS problem in authorship identification

In the field of authorship identification, the general common source problem of determining whether two texts have been written by the same author or by a different author is called the Authorship Verification (AV) task. The specific source problem of finding the specific author of an unknown text is called the

Authorship Attribution (AA) task. In forensic literature, the AV task [22, 23, 44, 47, 48] has been studied more extensively than the AA task [20]. Due to the high relevancy in court cases and the lack of research, this thesis will focus on the AA problem.

6.4. Likelihood Ratio

We want to define the influence of the set of scientific evidence E on the odds of hypothesis H_p against H_d. The set E is defined as $E = \{e_u, e_s, e_b\}$ for the specific source problem. In court cases often additional non-scientific evidence is considered, for example, motive, opportunity, eyewitness accounts and alibi [15]. The set of all initial, non-scientific evidence is called I. Prior odds, meaning the conviction of the court about the non-scientific evidence, are defined as

$$\frac{P(H_p|I)}{P(H_d|I)}. \tag{6.1}$$

We want to update this value by including the evidence E. To do this we need to calculate the posterior odds from the prior odds. the posterior odds are defined as

$$\frac{P(H_p|E,I)}{P(H_d|E,I)}. \tag{6.2}$$

To update the prior odds, we use Bayes' Theorem, a well-known result following directly from the definition of conditional probability $\left(P(A|B) = \frac{P(A\cap B)}{P(B)}\right)$. It states:

Theorem 6.4.1 (Bayes' theorem)**.** *Let A and B be events where P(B)>0. Then*

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using this theorem on both the top and the bottom of the posterior odds, and then applying the definition of conditional and Bayes' theorem again, leads us to:

$$\begin{aligned}\frac{P(H_p|E,I)}{P(H_d|E,I)} &= \frac{\frac{P(E,I|H_p)P(H_p)}{P(E,I)}}{\frac{P(E,I|H_d)P(H_d)}{P(E,I)}} = \frac{P(E,I|H_p)P(H_p)}{P(E,I|H_d)P(H_d)} \\ &= \frac{P(E|I,H_p)P(I|H_p)P(H_p)}{P(E|I,H_d)P(I|H_d)P(H_d)} = \frac{P(E|I,H_p)\frac{P(I|H_p)P(H_p)}{P(I)}}{P(E|I,H_d)\frac{P(I|H_d)P(H_d)}{P(I)}} \\ &= \frac{P(E|I,H_p)}{P(E|I,H_d)} \cdot \frac{P(H_p|I)}{P(H_d|I)}\end{aligned} \tag{6.3}$$

We call the factor $\frac{P(E|I,H_p)}{P(E|I,H_d)}$ the Likelihood Ratio (LR). The LR is calculated by forensic scientists and presented to the court as the value of the evidence E. The court can then take the value of the evidence into account to update their prior beliefs. In practice, the non-scientific evidence I is often left out of the formula of the likelihood ratio, reducing it to

$$\text{LR} = \frac{P(E|H_p)}{P(E|H_d)}. \tag{6.4}$$

7 Likelihood Ratio Systems

To be able to use authorship attribution in court cases we need a model that outputs likelihood ratios when given validation samples. To transform classification models into likelihood ratio systems we go through three steps. First, we must alter the multiclass classification models into binary scorers as the likelihood ratio framework has only two hypotheses, H_p and H_d. Kernel density estimation (KDE) is then used to calibrate a function that transforms the scores into likelihood ratios. Lastly, we have to bound the resulting LRs to prevent over- or underestimating them in the tail regions of the KDEs. During the construction of our likelihood ratio system we follow a part of the guideline for the construction of score-based LR systems by Leegwater et al [32]. Our three steps correspond to steps 4 and 5 in their paper.

7.1. Binary scorer

We perform two steps to convert a multiclass classification model into a binary scoring model. We use the same types of models, but with a small modification to the dataset and how we handle the resulting score. Firstly, we set one of the authors as the suspect. The texts written by this author, and thus with true hypothesis H_p are given label 1, all other texts, those with true hypothesis H_d, are given label 0. The binary scorer is then trained to assign a number, corresponding to the likelihood of that sample being written by the suspect. The binary version of the SVM and baseline models return a single score, called the decision value. We can directly use this value for the calibration step. For the BERT models, the classification head returns two values, one corresponding to the H_p class and one to the H_d class. To convert these values to a single score we calculate the logit of the softmax of the results of the classification head, meaning

$$s = \log\left(\frac{p_1}{1-p_1}\right), \quad p_1 = \frac{e^{r_1}}{e^{r_1}+e^{r_2}}. \tag{7.1}$$

Here s is the resulting score, and r_1 and r_2 are the results from the classification head for the classes H_p and H_d. s corresponds to the score the H_p class should be given to ensure the same resulting softmax probability p_1 if the score of the H_d class is set to 0. This ensures that the score s corresponds to the relative differences between the scores of the two classes.

7.2. Kernel Density Estimation

To convert the scores from the binary scorer into likelihood ratios we use Kernel Density Estimators (KDEs). A KDE estimates a probability density function (PDF) from samples drawn from a distribution. This is done by creating a standard PDF, the kernel function, around each sample and taking the average of the individual PDFs as the overall PDF. Mathematically, we can write this as

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right). \tag{7.2}$$

Here K is the kernel function, $x_1, ..., x_n$ are the samples drawn from a distribution and h is the bandwidth used for K. $\hat{f}(x)$ is the resulting estimate of the PDF. We will use the most used kernel for likelihood ratio systems, the Gaussian kernel. For a Gaussian kernel, the kernel function is equal to the PDF of a standard normal:

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{7.3}$$

If we write out $K\left(\frac{x-x_i}{h}\right)$ we get a scaled version of a normal distribution with $\mu = x_i$ and $\sigma = h$. So we fit a normal distribution with a standard deviation of h around each sample and take the average of all these normal distributions to find our KDE.

In kernel density estimation an important choice is the value of the bandwidth h. A too small value of h leads to overfitting on the individual samples, resulting in a PDF with a peak at the score of each sample. On the other hand, a too large value of h results in a too general model, that does not fit the data. The optimal value of h can only be computed in the theoretical case where the underlying density $f(x)$ is known. In practice, Silverman's rule of thumb (7.4) is often used [49]. This estimator is modified from the optimal value of h for samples from an underlying Gaussian distribution. It is given by

$$\hat{h} = 0.9 \min\left(\hat{\sigma}, \frac{\text{IQR}}{1.34}\right) N_c^{-1/5}. \tag{7.4}$$

Here $\hat{\sigma}$ is the estimated standard deviation of the samples and the IQR is the interquartile range, the difference between the 75th and 25th percentile of the scores. N_c corresponds to the number of samples in the calibration set.

In the likelihood ratio framework, KDEs can be used to construct two PDFs, $\hat{f}_p$ for the samples under H_p and $\hat{f}_d$ for the samples under H_d. These PDFs represent the probability that a sample under that hypothesis is given a score s by the binary scoring model. Therefore, the score s of a new sample can be used to estimate the corresponding likelihood ratio:

$$LR(s) = \frac{P(E|H_p)}{P(E|H_d)} = \frac{\hat{f}_p(s)}{\hat{f}_d(s)} \tag{7.5}$$

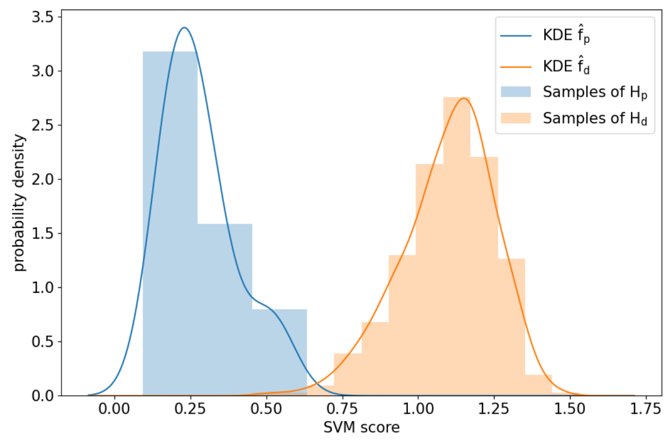


Figure 7.1: Plot containing two histograms of the density of calibration samples under H_p and H_d, in blue and orange respectively, against their score given by a binary SVM scorer. Also included are the corresponding KDE fits in the same colours. The score has a scale from negative infinity to infinity. Made on a subset of FRIDA consisting of 50 authors with each 7 texts in the training-and-calibration set, using 7-fold cross-calibration.

An example of the KDEs fitted on texts of both classes can be seen in Figure 7.1. The blue histogram and line correspond to the samples under H_p and the KDE fit $\hat{f}_p$. Similarly, the orange histogram corresponds to samples under H_d, with the orange line being the resulting fit $\hat{f}_d$. This example comes from a problem with 50 authors and 7 calibration texts per author. As a result, the fit for H_p is done on

7 samples, against 343 samples under H_d. We also notice this in the bandwidth of the respective KDE fits, which is smaller for $\hat{f}_d$ due to the factor $N_c^{-1/5}$ in Equation (7.4).

7.2.1. Cross-calibration

We cannot calibrate the KDEs on texts from the training dataset, as the binary scorer has already seen these texts during training and therefore scores them differently than unseen texts. Thus we split our dataset not into two but three parts, a training set, a calibration set and a validation set. The training set is used to train the binary scorer and then the scores computed for samples in the calibration set are used to fit the KDEs. Lastly, the system is validated by estimating the LRs on the validation set and comparing their values with the ground truth (Chapter 8).

A common limitation in authorship attribution cases is the available number of texts, especially for samples under H_p. To use all available texts to their maximal potential, we use k-fold cross-calibration. In k-fold cross-calibration, we have a joint training and calibration set split in k folds. We then train k different models, leaving one of the k folds out of the training set to be used as the calibration set. Each iteration returns a set of scores created by the model on the respective calibration set. We then combine all these calibration scores and compute the KDEs under both hypotheses. Lastly, we train a new model on the entire training and calibration set and use that model together with the KDEs to calculate the resulting likelihood ratios on the validation set. In Figure 7.2 the principle of cross-calibration is illustrated with $k = 4$.

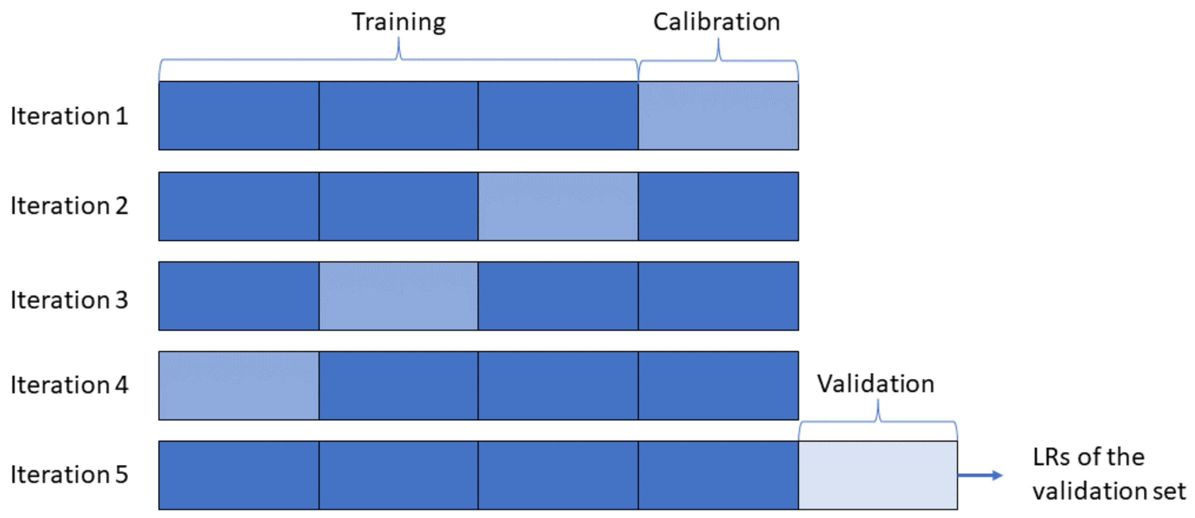


Figure 7.2: Illustration showing the concept of 4-fold cross-calibration. The scores on the calibration set in iterations 1-4 are used to create the KDEs that calculate the LRs on the validation set.

The advantage of cross-calibration is that we can use more data to fit the KDEs and train the model that is used on the validation set. The biggest drawback is that the models used to calculate the calibration scores are not the same models as the model used to calculate the scores of the texts in the validation set. We assume that due to the calibration models being trained on a significant subset of the set on which the validation model is trained the difference in scores is small. In practice, using cross-calibration meant that we could use 7 samples under H_p for both training and calibration, compared to 5 and 2 respectively under normal calibration. This increase in training and calibration texts results in a significantly better performance. A second drawback of using k-fold cross-calibration is that it increases the computational load significantly, by approximately a factor k.

7.3. Feature-based likelihood ratio systems

The likelihood ratio system introduced in this chapter is a score-based method. This means that text samples are first given a one-dimensional score, from which the system estimates a likelihood ratio. Next to score-based methods, also feature-based methods exist, where a kernel density estimator is fitted over all the feature dimensions, leading to a multidimensional KDE. This could for example be done with the features used in the SVM models or with the hidden state of dimension 512 resulting from the classifier token in BERT-based models. Using the working of KDEs and the fact that we only have 7 calibration samples under H_p we will motivate our decision to focus on score-based likelihood ratio systems.

Creating a well-calibrated one-dimensional KDE on 7 samples is challenging, but in more than 100 dimensions this is next to impossible, due to the low density of samples in the multidimensional space. So, to build a feature-based model we would have to restrict ourselves to only a few features, losing a lot of information in the process. As our binary scoring models convert all the features into a single score, they can still use parts of the information present in all features. As they are also better suited for dealing with the low number of calibration texts under H_p in authorship attribution tasks, we chose to focus on score-based likelihood ratio systems.

7.4. Dataset Balance

With the creation of the binary problem, we have also created an unbalanced training dataset. As the texts from all authors except for the suspect have true hypothesis H_d, we have significantly more texts under H_d than under H_p. For the SVM models, this unbalance did not result in problems. While these models did classify almost all samples to be more likely to be not from the suspect, the following calibration step using KDEs solves this issue. This can also be seen in Figure 7.1, where we notice that all calibration samples (the histogram) get assigned a score greater than 0, on a scale from negative infinity to infinity. This means that the SVM would classify every calibration text as more likely to be written by someone else than the suspect if we were to use it for classification. However, a clear separation between the scores for samples under H_p and samples under H_d is seen, resulting in a well-performing LR system after calibration with KDEs.

The BERT models reached a local optimum where all texts were classified as under H_d and this did not improve with further finetuning. To solve this we altered the weight of the samples under H_p, increasing the loss for miss-classification of those samples. We chose to give text under H_p a weight of $n-1$, with n the number of authors, such that the total weight of training texts is $7(n-1)$ for both classes. A drawback of this method is that the 7 texts of the suspected author have significantly more impact on the model than all other texts. Especially for large sets of authors, this can lead to significant overfitting on the training texts in this class. In practice, we noticed that due to this overfitting on texts under H_p we had to restrict the number of epochs we trained our model in comparison to the computational authorship attribution task.

7.5. Bounding

Calculating likelihood ratios using kernel density estimation can result in extremely large or small LRs when in the tail distribution of the scores under respectively H_d and H_p. As there are no calibration samples from one of the distributions in these regions, it is difficult to calibrate these extremely large LRs. To combat this problem we want to find both a lower and upper bound on the values an LR can take for a specific problem.

Several methods for calculating such bounds exist, with varying levels of complexity. A simple rule of thumb states: "Likelihood ratios should never be larger than the size of the training dataset under H_d, or smaller than 1 divided by the size of the training data set under H_p" [58]. Note that the training dataset in this quote corresponds to what we call the calibration dataset. Their justification for this bound is that a conservative estimate for the probability that the score of a new sample under H_p is smaller than the scores of all previous samples under H_p is $\frac{1}{N_p}$, with N_p being the number of samples under H_p. While this approach is pragmatic, it lacks a theoretical basis and ignores much information present in the LRs calculated on the calibration set [58]. To combat this Vergeer et al [58] proposed a method called Empirical Lower and Upper Bound (ELUB).

The ELUB method is based in decision theory and tries to find the region of LRs in which the LRs given by the proposed system result in a higher expected utility than the reference system. The reference system is a likelihood ratio system that always returns an LR of 1, independent of the evidence. A suspect is only convicted if the utility of conviction is larger than that of acquittal. From this, it can be reasoned that there must be a threshold LR_{th} such that when the likelihood ratio is larger than this number the suspect is convicted. Using this reasoning, Vergeer et al [58] generalize the utilities of the four possible outcomes to the following values:

$$\begin{aligned}
\text{Guilty } (H_p) \text{ and convicted } (LR > LR_{th}) &: U = 0 \\
\text{Guilty } (H_p) \text{ and acquitted } (LR \leq LR_{th}) &: U = -1 \\
\text{Innocent } (H_p) \text{ and convicted } (LR > LR_{th}) &: U = -LR_{th}\frac{P(H_p)}{P(H_d)} \\
\text{Innocent}(H_d) \text{ and not acquitted } (LR \leq LR_{th}) &: U = 0.
\end{aligned}$$

The expected utility (EU) of a well-calibrated system is then equal to

$$\begin{aligned}
\text{EU} &= \mathbb{E}\left[U\right] = \mathbb{E}\left[U|H_p\right]P(H_p) + \mathbb{E}\left[U|H_d\right]P(H_d) \\
&= -P(LR \leq LR_{th}|H_p)P(H_p) - LR_{th}\frac{P(H_p)}{P(H_d)}P(LR > LR_{th}|H_d)P(H_d) \\
&= -P(LR \leq LR_{th}|H_p)P(H_p) - LR_{th}P(H_p)P(LR > LR_{th}|H_d) \\
&= -P(H_p)\Big(P(LR \leq LR_{th}|H_p) + LR_{th}P(LR > LR_{th}|H_d)\Big)
\end{aligned} \tag{7.6}$$

If we compute this for both the reference system and another LR system we find the EU ratio:

$$\text{EU ratio} = \frac{\text{EU(reference)}}{\text{EU(LR system)}} = \frac{P(1 \leq LR_{th}|H_p) + LR_{th}P(1 > LR_{th}|H_d)}{P(LR \leq LR_{th}|H_p) + LR_{th}P(LR > LR_{th}|H_d)} \tag{7.7}$$

If this ratio is greater than 1 the negative utility of using the reference system is greater than that of using the LR system. So we should only use the LR system in the region of LR_{th} where the EU ratio is greater than 1. The upper and lower bounds on the LRs are found by locating the threshold LRs (LR_{th}) for which the EU ratio is equal to 1.

The probabilities $P(LR \leq LR_{th}|H_p)$ and $P(LR > LR_{th}|H_d)$ are estimated from the LRs and their truth values in the calibration set. To estimate the bounds conservatively, two samples of misleading evidence are added to the set of calibration LRs: one corresponding to a sample under H_p with an LR of 0 and one with truth value H_d and an LR of infinity. As a consequence, for large LR_{th} we always have

$$P(LR > LR_{th}|H_d) > 0, \text{ while } P(1 > LR_{th}|H_d) = 0,\ \forall\, LR_{th} > 1.$$

From this, it follows that the EU ratio goes to 0 if the LR_{th} goes to 0. If the LR system is beneficial for some LR_{th} the EU ratio must be greater than 1 for those LR_{th}. Now the lowest LR_{th} where the EU ratio is lower than 1, gives us the upper bound on our LR system. Similarly, a lower bound is guaranteed due to the misleading sample under H_p with an LR of 0. Adding these misleading evidence points keeps us from over- or underestimating LRs due to small datasets that by chance consist of better-separated data compared to the true population.

Now, to ensure that the use of the likelihood ratio system is always beneficial over that of the reference system, all LRs greater than the upper bound are set to the upper bound and, similarly, all LRs smaller than the lower bound are set to the lower bound. This ensures that we only assign LRs in the region where this is beneficial given a conservative estimate.

7.5.1. Bounds on the optimal LR system

From the formula for the EU ratio, we can calculate the lower and upper bound of an optimal LR system, which gives each sample under H_p an LR of ∞ and under H_d an LR of 0. We can use these values to compare the results of our LR systems with the maximal reachable values when using an ELUB bounder with one misleading sample per class.

We start by calculating the upper bound. For $LR_{th} \geq 1$, we have that $P(1 > LR_{th}|H_d) = 0$ and $P(1 \leq LR_{th}|H_p) = 1$. Furthermore, for our optimal LR system with one misleading sample added, we have, if we set the number of calibration samples under H_d as N_d, that

$$P(LR \leq LR_{th}|H_p) = \frac{1}{N_p + 1} \text{ and } P(LR > LR_{th}|H_d) = \frac{1}{N_d + 1}, \tag{7.8}$$

as only for the added misleading samples we have $LR \leq LR_{th}$ under H_p or $LR > LR_{th}$ under H_d.

The EU ratio then becomes

$$\frac{\text{EU(reference)}}{\text{EU(LR system)}} = \frac{1}{\frac{1}{N_p+1} + \frac{LR_{th}}{N_d+1}}. \tag{7.9}$$

We note that this ratio is indeed larger than one for $LR_{th} = 1$ and then decreases for larger LR_{th}. We now look at where the expected utility of both systems is equal, so where the EU ratio is equal to 1. If we set this equation equal to 1 and factor out LR_{th} we get

$$LR_{th} = \frac{1 + N_d}{1/N_p + 1}. \tag{7.10}$$

In our standard case, we study 50 authors, of which one is marked as the suspect, with each 7 training and calibration texts. Therefore we have $N_p = 7$ and $N_d = 343$. If we plug this into Equation (7.10) we get

$$LR_{th} = 301. \tag{7.11}$$

So for an optimal LR system with our sample size, the ELUB bounder sets all LRs larger than 301 to the upper bound of 301. For LR systems that are not optimal, this upper bound can only be equal or lower, as no system can decrease the probabilities in the denominator of the EU ratio.

We can do something similar to get a lower bound on the LR of an optimal system. We now look at LR_{th} smaller than 1, which results in $P(1 > LR_{th}|H_d) = 1$ and $P(1 \leq LR_{th}|H_p) = 0$. As the denominator remains the same as in the upper bound case we have

$$\frac{\text{EU(reference)}}{\text{EU(LR system)}} = \frac{LR_{th}}{\frac{1}{N_p+1} + \frac{LR_{th}}{N_d+1}}. \tag{7.12}$$

If we again set this to be equal to 1 and factor our LR_{th}, we get

$$LR_{th} = \frac{1 + 1/N_d}{N_p + 1}. \tag{7.13}$$

We can plug $N_p = 7$ and $N_d = 343$ into this equation to get the following lower bound on the LR of an optimal system with our sample sizes.

$$LR_{th} \approx 0.125 \tag{7.14}$$

We will use these bounds on the optimal LR system to calculate a lower bound on the log-likelihood-ratio cost, a metric for LR systems which we will introduce in Chapter 8.

8 Validation

There are several ways to measure the performance of an authorship attribution method. In this chapter, we will give an overview of these metrics. We will introduce precision, recall, macro-accuracy and F_1-score, which are often used as metrics in computational authorship attribution. After this, we will cover the performance metrics used to evaluate likelihood ratio systems in forensic authorship attribution, including the log-likelihood-ratio cost. We will also introduce several graphical methods which are used to interpret the results of likelihood ratio systems.

8.1. Metrics in computational authorship attribution

To enable us to define metrics mathematically, we will first describe the computational authorship attribution problem in terms of mathematical variables. Assume we have a set $A = \{a_1, a_2, ..., a_n\}$ consisting of n authors and sets T, V consisting of texts written by the authors in A, respectively the training and validation set. Let m be a model for multiclass classification that is trained on the texts in set T. We then define M as a $n \times n$ matrix, where element M_{ij} is defined as the number of texts in V written by author a_i attributed to author a_j by m. For an author $a_i, i \in \{1, 2, ..., n\}$ the **precision** of model m is defined as the ratio of all samples with true author a_i that are attributed to author a_i. In formula form, we write this as

$$\text{precision}(m, a_i, V) = \frac{M_{ii}}{\sum_{j=1}^{n} M_{ij}}. \tag{8.1}$$

Similarly, the **recall** is defined as the ratio of all samples that are classified as being written by author a_i of which the true author is a_i, mathematically

$$\text{recall}(m, a_i, V) = \frac{M_{ii}}{\sum_{j=1}^{n} M_{ji}}. \tag{8.2}$$

From the precision and recall per author, we can calculate the average recall and precision of the entire model. This can be done using either the micro-average or the macro-average. When using the **micro-average** we give the same weight to every text in V, resulting in

$$\text{precision}_{\text{micro}}(m, V) = \frac{\sum_{i=1}^{n} M_{ii}}{\sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij}}, \tag{8.3}$$

$$\text{recall}_{\text{micro}}(m, V) = \frac{\sum_{i=1}^{n} M_{ii}}{\sum_{i=1}^{n}\sum_{j=1}^{n} M_{ji}}. \tag{8.4}$$

We note that the terms on the right-hand side of these equations are equal, resulting in

$$\text{precision}_{\text{micro}}(m, V) = \text{recall}_{\text{micro}}(m, V), \tag{8.5}$$

which is always true in multiclass classification. Intuitively, this can be explained by the fact that a false positive for one class must always correspond to a false negative for another class. As a result, the total number of false positives is equal to the total number of false negatives. The micro-averaged precision is therefore also called the micro-accuracy or simply the accuracy of the model.

When taking the **macro-average** each class is given the same weight, resulting in

$$\text{precision}_{\text{macro}}(m, V) = \frac{1}{n}\sum_{i=1}^{n}\text{precision}(m, a_i, V) = \frac{1}{n}\sum_{i=1}^{n}\frac{M_{ii}}{\sum_{j=1}^{n} M_{ij}}, \tag{8.6}$$

$$\text{recall}_{\text{macro}}(m, V) = \frac{1}{n}\sum_{i=1}^{n}\text{recall}(m, a_i, V) = \frac{1}{n}\sum_{i=1}^{n}\frac{M_{ii}}{\sum_{j=1}^{n} M_{ji}}. \tag{8.7}$$

The macro-averaged precision and recall reduce to their micro-averaged versions whenever the value in the denominator is constant. For the precision, a constant denominator means that each author has written an equal number of the texts in V, while for recall it means to each author an equal number of texts is attributed. In literature, the macro-averaged precision is also sometimes called the **macro-accuracy** [37].

We also define the F_β-score, which is a weighted average of the precision and the recall

$$F_\beta(m, V) = (1+\beta^2)\frac{\text{precision}(m, V)\cdot\text{recall}(m, V)}{\beta^2\cdot\text{precision}(m, V) + \text{recall}(m, V)}. \tag{8.8}$$

We can again calculate either a micro-averaged or a macro-averaged F_β-score. Here the micro-averaged F_β-score is calculated from $\text{precision}_{\text{micro}}(m, V)$ and $\text{recall}_{\text{micro}}(m, V)$. Due to their equality, we can reduce this Equation to

$$\begin{aligned} F_{\beta,\text{micro}}(m, V) &= (1+\beta^2)\frac{\text{precision}_{\text{micro}}(m, V)\cdot\text{recall}_{\text{micro}}(m, V)}{\beta^2\cdot\text{precision}_{\text{micro}}(m, V) + \text{recall}_{\text{micro}}(m, V)} \\ &= (1+\beta^2)\frac{(\text{precision}_{\text{micro}}(m, V))^2}{(1+\beta^2)\text{precision}_{\text{micro}}(m, V)} = \text{precision}_{\text{micro}}(m, V). \end{aligned} \tag{8.9}$$

So the micro F_β reduces to the micro-averaged precision for multiclass classification problems. Therefore it is more interesting to look at the macro-averaged F_β-score. Specifically, we look at the **macro $\mathbf{F_1}$-score**, as F_1 is the harmonic mean between the recall and the precision, giving both equal importance. We get

$$F_{1,macro}(m, V) = \frac{1}{n}\sum_{i=1}^{n}F_1(m, a_i, V) = \frac{2}{n}\sum_{i=1}^{n}\frac{\text{precision}(m, a_i, V)\cdot\text{recall}(m, a_i, V)}{\text{precision}(m, a_i, V) + \text{recall}(m, a_i, V)}. \tag{8.10}$$

We have now introduced a range of performance metrics used in multiclass classification problems like authorship attribution. Of these macro-accuracy and the macro F_1-score are most often used in literature [37, 54]. We will use the macro F_1-score as our primary metric to compare the performance of multiclass classification models. We choose this metric over the other options as it can properly handle imbalanced datasets. For example, for micro F_1-score, good performance for authors with many texts in the test set can skew the overall accuracy. Furthermore, macro F_1-score considers both precision and recall, which makes it a preferable metric over macro-accuracy.

In this thesis we will often refer to the macro F_1-score as simply the F_1-score for readability. When we use the term F_1-score this always refers to the macro-averaged version.

8.2. Validation in forensic sciences

Likelihood ratio systems return a likelihood ratio (LR) instead of a classification, such that the strength of the evidence can be weighted in court. As a result, when we quantify the performance of a likelihood ratio system we want to take the value of the LRs into account. For example, given a text not written by the suspect, we prefer a model that returns an LR of 2 over one that returns an LR of 300, even though both models wrongly express that the text is more likely to be written by the suspect. Therefore, to properly measure the performance of likelihood ratio systems, we need different performance metrics than those we just introduced for the multiclass classification problem.

To properly validate our LR systems we follow the validation framework proposed by Meuwly et al [35]. This paper gives a guideline to both performance metrics and graphical representations for the validation of likelihood ratio systems. This is done following three main performance characteristics: accuracy, discriminating power and calibration. In this context, accuracy means how closely the calculated LRs correspond to the truth values. This is measured by the log-likelihood-ratio cost (C_{llr}). The discriminating power is defined as the performance of the model in distinguishing between samples from H_p and samples from H_d when calibrated optimally. The main metric used to evaluate the discriminating power is the minimal log-likelihood-ratio cost (C_{llr}^{min}). Lastly, the performance characteristic calibration describes how well the likelihood ratio system is calibrated, where the discriminating power of the model is assumed as fixed. The calibration of an LR system is measured using the calibration log-likelihood-ratio cost (C_{llr}^{cal}). Next to the performance metrics, three graphical representations are used to graphically illustrate the performance of a likelihood ratio system, namely ECE plots, Tippett plots and PAV plots. An overview of the three performance characteristics and their corresponding metrics and graphical representations is given in Table 8.1. In the following subsections, we will go over each of the three main performance characteristics in depth, introducing the formal definitions of their respective metrics and explaining their graphical representations.

Table 8.1: Overview of performance characteristics and their corresponding metrics [35].

Performance characteristic	Performance metric	Graphical representation
Accuracy	C_{llr}	ECE plot
Discriminating power	C_{llr}^{min}	Tippet plot, ECE plot
Calibration	C_{llr}^{cal}	Tippet plot, PAV plot, ECE plot

8.2.1. Accuracy

The **log-likelihood-ratio cost** (C_{llr}) is a metric introduced by Brümmer and Du Preez [11] to measure the accuracy of a likelihood ratio system independent of the domain of application. Let V be the set of likelihood ratios (LRs) computed by a likelihood ratio system on a validation set. V can be split into two disjoint subsets V_p and V_d, with H_p and H_d being the true hypothesis for texts corresponding to the LRs in V_p and V_d respectively. The C_{llr} is then defined as follows

$$C_{llr} = \frac{1}{2|V_p|} \sum_{LR \in V_p} \log_2\left(1 + \frac{1}{LR}\right) + \frac{1}{2|V_d|} \sum_{LR \in V_d} \log_2\left(1 + LR\right). \tag{8.11}$$

We can interpret this formula as follows: if a sample from H_p has a large LR, the cost added by that sample, $\log_2\left(1 + \frac{1}{LR}\right)$, is close to zero. If the LR is 1 we get a cost of $\log_2(2) = 1$ for that sample and the added cost increases for LRs closer to zero. The inverse is true for samples from H_d, LRs close to zero contribute minimally to the overall cost compared to larger LRs. We notice that the contribution to the C_{llr} is 1 for samples under H_d with an LR of 1.

Consider a hypothetical system that returns an LR of 1 for every sample, effectively disregarding the evidence and classifying all samples as equally likely under H_p and H_d. This system, also called the reference system, can always be constructed, as it acts independently of the evidence. The reference system's C_{llr} is always 1, as every sample adds exactly 1 to the average. Thus, any likelihood ratio system used in court must achieve a C_{llr} lower than 1. Otherwise, the system's performance is worse than having no information at all.

Recall that we computed a lower and upper bound on the LRs of our LR system for our dataset size (Chapter 7.5.1). From this we can compute a lower bound on the C_{llr} for our LR system. For the optimal LR system, $LR = 301$, $\forall LR \in V_p$ and $LR = 0.125$, $\forall LR \in V_d$ when using an ELUB bounder with one misleading sample per hypothesis. This gives

$$\begin{aligned} C_{llr} &= \frac{1}{2|V_p|}|V_p| \log_2\left(1 + \frac{1}{301}\right) + \frac{1}{2|V_d|}|V_d| \log_2(1 + 0.125) \\ &= \frac{1}{2}\log_2\left(1 + \frac{1}{301}\right) + \frac{1}{2}\log_2(1 + 0.125) \approx 0.088. \end{aligned} \tag{8.12}$$

So we can conclude that with our sample sizes the C_{llr} has a lower bound of 0.088. This gives us a way to compare our results with the attainable optimum, as no model can reach a C_{llr} of 0 with our configuration of the ELUB bounder.

To illustrate the metrics used for the validation we study a simplified example with only six data points, as shown in Table 8.2. We will come back to this example for each of the forensic validation metrics.

Table 8.2: Simplified example of LRs generated by an LR system and their corresponding truth values

Sample	LR	H
s_1	0.01	H_d
s_2	0.1	H_d
s_3	1	H_d
s_4	5	H_p
s_5	10	H_d
s_6	100	H_p

For this example the log-likelihood-ratio cost is

$$\begin{aligned} C_{llr} &= \frac{1}{2\cdot 2}\left(\log_2\left(1+\frac{1}{5}\right)+\log_2\left(1+\frac{1}{100}\right)\right) \\ &+\frac{1}{2\cdot 4}\Big(\log_2(1+0.01)+\log_2(1+0.1)+\log_2(1+1)+\log_2(1+10)\Big) \\ &= \frac{1}{4}(0.263+0.014)+\frac{1}{8}(0.014+0.138+1+3.459) = 0.646. \end{aligned} \tag{8.13}$$

In this example, the C_{llr} value is less than 1, indicating that the log-likelihood-ratio system performs better than a system ignoring the evidence. Furthermore, the only misclassified sample, s_4, contributes 0.432 to the C_{llr}, which accounts for 67% of the total cost. This illustrates the significant impact of misclassified samples on the C_{llr}. The C_{llr} is influenced by both the frequency of misclassifications and the magnitude of the corresponding LRs.

8.2.2. Discriminating power

The minimal log-likelihood-ratio cost, C_{llr}^{min}, is also calculated using Equation (8.11). The difference between the C_{llr} and the C_{llr}^{min} is that the set V of LRs is first modified to V^{opt} using isotonic regression when calculating the C_{llr}^{min}. It has been shown that optimal calibration for binary scoring rules is found when recalibrating using isotonic regression [7, 10]. As the C_{llr} is a binary scoring rule [11], we can recalibrate our likelihood ratios to find the minimal likelihood ratio cost for this system. As the isotonic regression removes calibration errors, the remaining mistakes are solely caused by the discriminative power of the likelihood ratio system. As a result, we can use the C_{llr}^{min} as a metric for the discriminative power.

Isotonic regression, or monotonic regression, is a non-parametric method to find a monotonically increasing or decreasing function that minimizes the sum of squared differences between the observed and the predicted values. We will focus on increasing isotonic regression, but the principles are the same for the decreasing case, with inverted signs. To write the objective of isotonic regression in formula form, we suppose we have N data points of the form (x_i, y_i) with weights w_i, $i = 1, ..., N$, with ordered x values, so $x_1 \leq x_2 \leq .. \leq x_N$. Then we minimize the following objective under the monotonic non-decreasing condition:

$$\begin{aligned} &\text{min} \quad \sum_{i=1}^{N} w_i(\hat{y}_i - y_i)^2 \\ &\text{subject to:} \\ &\hat{y}_i \leq \hat{y}_j, \quad \forall i,j \text{ where } x_i < x_j \end{aligned} \tag{8.14}$$

The set of $\hat{y}$ that minimizes this objective can be found using the Pool Adjacent Violators (PAV) algorithm [2]. We initialize the algorithm by setting $\hat{y}_i = y_i$. The algorithm works by iteratively combining adjacent data points that violate the isotonicity constraint, so $\hat{y}_i \geq \hat{y}_{i+1}$ while $x_i < x_{i+1}$. The algorithm combines the adjacent points into a single point, with $\hat{y}_{i:i+1} = \frac{w_i\hat{y}_i + w_{i+1}\hat{y}_{i+1}}{w_i + w_{i+1}}$ and a weight $w_{i:i+1} = w_i + w_{i+1}$. Here we use the subscript $i : j$ to write a point that combines all points from x_i up to and including x_j. As this algorithm uses only the adjacency of points and not their specific x value, it only matters that $x_{i:j}$ keeps the same location in the ordering. Therefore we may set $x_{i:j} = x_i$.

If we iteratively perform this algorithm until the monotonicity constraint is satisfied we are left with a set of predictors of the form $\{\hat{y}_{1:a}, \hat{y}_{a+1:b}, ..., \hat{y}_{z+1:N}\}$. For the specific data points, we get that $\hat{y}_1 = ...\hat{y}_a = \hat{y}_{1:a}$, so the combined points all have the same predicted value. We can now define the function $f(x) : [x_1, x_N] \rightarrow [\hat{y}_1, \hat{y}_N]$ by linearly interpolating between the predicted values of $\hat{y}_i$. The resulting function is then piecewise linear. In Figure 8.1 an example of isotonic regression is shown to illustrate this process. The blue points represent the original dataset and the orange line represents the isotonic regression fit. We notice that the line consists of two types of line segments. We have constant line segments, where the isotonic regression combined multiple data points to satisfy the monotonic increasing constraint, and increasing line segments, where the estimator linearly interpolates between two adjacent estimates $\hat{y}_{i:j}$ and $\hat{y}_{j+1:k}$.

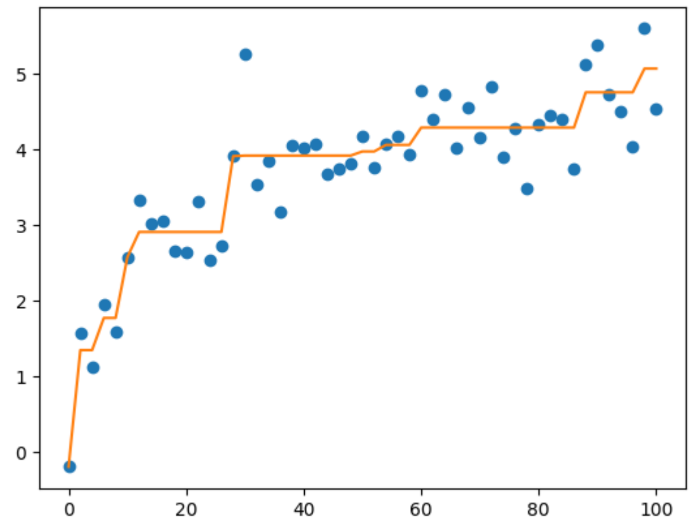


Figure 8.1: Example of isotonic regression on a simulated dataset. The blue points represent the original dataset and the orange line represents the isotonic regression fit.

We now translate this process to likelihood ratios. When comparing to our introduction of isotonic regression, x represents the original likelihood ratios in the set V, y the true likelihood ratios, 0 for samples under H_d and infinity for samples under H_p, and $\hat{y}$ the likelihood ratios after recalibration, so V^{opt}. The weight of the samples is chosen such that the total weight of the samples under both hypotheses is equal, giving

$$w_p = \frac{N_d}{N_p + N_d}, \quad w_d = 1 - w_p. \tag{8.15}$$

Here, N_p and N_d are the total number of samples in the validation set under respectively H_p and H_d, while w_p and w_d represent the weights assigned to the two classes.

For likelihood ratios the value of y for samples with true hypothesis H_p is positive infinity. If we averaged the LRs off several data points, the average would always be infinity if at least one sample from H_p is included. Instead, when combining two data points of equal weight, one under H_p and one under H_d, we want their combined LR to be 1, as both hypotheses are equally likely. In general, we can write the expected LR when averaging data points as

$$LR_{\text{avg}} = \frac{w_p n_p}{w_d n_d}. \tag{8.16}$$

Here, n_p and n_d are the number of samples under respectively H_p and H_d being combined.

To combat this issue we first transform the LRs to odds by using the following transformation

$$p(LR) = \begin{cases} \frac{LR}{1+LR}, & \text{if } LR < \infty \\ 1, & \text{else.} \end{cases} \tag{8.17}$$

This one-to-one mapping returns values of p between 0 and 1. If we now average n_p samples with weight w_p and $p = 1$ and n_d samples with weight w_d and $p = 0$ we get the following p_{avg}:

$$p_{\text{avg}} = \left(\sum_{s:H=H_p} w_p + \sum_{s:H=H_d} w_d \right)^{-1} \left(\sum_{s:H=H_p} w_p \cdot 1 + \sum_{s:H=H_d} w_d \cdot 0 \right) = \frac{w_p n_p}{w_p n_p + w_d n_d} \tag{8.18}$$

Now we can transform p_{avg} back into a LR using the inverse of the transformation from Equation (8.17), $LR(p) = \frac{p}{1-p}$. This gives

$$LR_{\text{avg}} = \frac{\frac{w_p n_p}{w_p n_p + w_d n_d}}{1 - \frac{w_p n_p}{w_p n_p + w_d n_d}} = \frac{w_p n_p}{w_p n_p + w_d n_d - w_p n_p} = \frac{w_p n_p}{w_d n_d} \tag{8.19}$$

This corresponds to the value we expect to get when combining LRs, as stated in Equation (8.16). The transformation of likelihood ratios into odds solves the problem with infinite LR values and thus allows us to perform isotonic regression using the PAV algorithm on likelihood ratios. So, the C_{llr}^{min} is defined as follows

$$C_{llr}^{min} = \frac{1}{2|V_p^{opt}|} \sum_{LR \in V_p^{opt}} \log_2 \left(1 + \frac{1}{LR} \right) + \frac{1}{2|V_d^{opt}|} \sum_{LR \in V_d^{opt}} \log_2 (1 + LR). \tag{8.20}$$

We finally note that the maximal possible value for C_{llr}^{min} is 1. The reason for this is that a system with poor discriminative power can always be recalibrated to the reference system, with an LR of 1 for each sample. As the reference system has a C_{llr} of 1, which we showed in the previous section, this results in a C_{llr}^{min} of 1.

We now return to our simplified example from Table 8.2 to illustrate the calculation of the C_{llr}^{min}. We first order the values based on their LR and convert their hypothesis to $y = \infty$ for samples under H_p and $y = 0$ for samples under H_d. We notice that samples with the three lowest LR have $y_{(1)} = y_{(2)} = y_{(3)} = 0$, which corresponds to the monoticity constraint. The fourth and fifth sample have $y_{(4)} = \infty$ and $y_{(5)} = 0$, violating the constraint. To combine these points we first calculate their weights, $w_p = \frac{N_d}{N_p + N_d} = \frac{2}{3}$ and $w_d = 1 - w_p = \frac{1}{3}$. Their combined LR then becomes $LR_{\text{avg}} = \frac{\frac{2}{3} \cdot 1}{\frac{1}{3} \cdot 1} = 2$. Lastly, $y_{(6)} = \infty$, satisfying the monoticity constraint. We summarize the calibrated LRs in Table 8.3.

Table 8.3: Simplified example of LRs generated by a likelihood ratio system, their corresponding truth values and their calibrated LRs.

Sample	LR	H	LR_{cal}
s_1	0.01	H_d	0
s_2	0.1	H_d	0
s_3	1	H_d	0
s_4	5	H_p	2
s_5	10	H_d	2
s_6	100	H_p	∞

From this we can calculate the C_{llr}^{min} of this system:

$$\begin{aligned} C_{llr}^{min} &= \frac{1}{2\cdot 2}\left(\log_2\left(1+\frac{1}{2}\right)+\log_2\left(1+\frac{1}{\infty}\right)\right) \\ &+ \frac{1}{2\cdot 4}\Big(\log_2(1+0)+\log_2(1+0)+\log_2(1+0)+\log_2(1+2)\Big) \\ &= \frac{1}{4}(0.585+0)+\frac{1}{8}(0+0+0+1.585)=0.344. \end{aligned} \tag{8.21}$$

8.2.3. Calibration

The calibration log-likelihood-ratio cost is defined as the difference between the C_{llr} and the C_{llr}^{min}

$$C_{llr}^{cal} = C_{llr} - C_{llr}^{min}. \tag{8.22}$$

Intuitively this is a logical way of measuring the calibration of a system. Namely, if the C_{llr} measures the overall performance and the C_{llr}^{min} the performance of the optimally calibrated system, then the difference measures how much the system can be improved with better calibration. The split from a C_{llr} into a C_{llr}^{min} and C_{llr}^{cal} allows us to make more in-depth comparisons of LR systems and improve them more systematically. For example, if the C_{llr}^{cal} of a likelihood ratio system is larger than the C_{llr}^{min} it is often more useful to investigate ways of improving the calibration than to pursue a system with a greater discriminating power. Returning to our simplified example, we get $C_{llr}^{cal} = C_{llr} - C_{llr}^{min} = 0.646 - 0.344 = 0.302$, meaning that a significant part of our total cost is caused by poor calibration.

8.2.4. Graphical representations

PAV plot

A pool adjacent violators (PAV) plot shows the pre-calibrated LRs of a likelihood ratio system against the LRs fitted on their truth values using isotonic regression. While the PAV plot is used to illustrate the calibration of a likelihood ratio system, it is made using the PAV algorithm. In Figure 8.2 the PAV plot of the simplified example can be seen. Note that a cut-off is used to show data points with optimal log-likelihoods of infinity or negative infinity. For a well-calibrated system, all points are expected to be close to the $x = y$-diagonal, meaning that the post-calibrated LRs differ little from the pre-calibrated LRs. A PAV plot can be useful for identifying the cause for high values of C_{llr}^{cal}. Causes could include under- or overestimating LRs, meaning the points are placed consistently above or below the diagonal. Looking at our example in Figure 8.2 we notice that the LRs were slightly overestimated before the recalibration with isotonic regression.

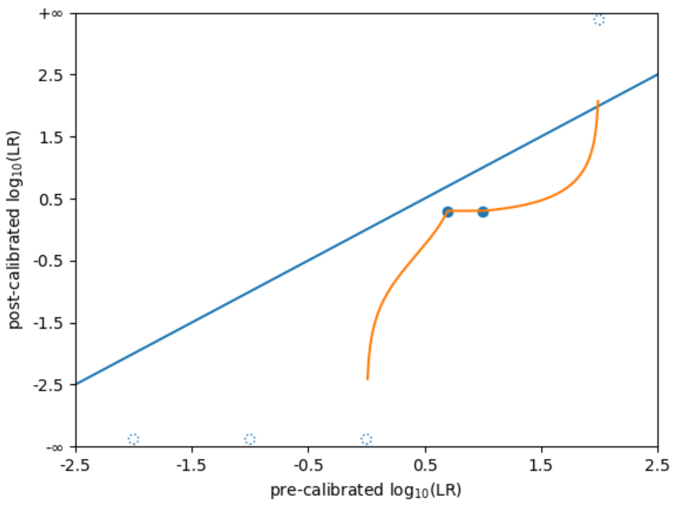


Figure 8.2: PAV plot of the example data set. The filled-in points have finite values. The blue line represents the $x = y$-diagonal and the orange line is gotten through linear interpolation of the points.

ECE plot

The empirical cross entropy (ECE) plot shows the value of the ECE against the log-odds of the prior probability of the likelihood ratio $\left(p = \frac{P(H_p)}{P(H_d)}\right)$. It is defined as [43]

$$ECE = \frac{p}{(p+1)N_p} \sum_{LR \in V_p} \log_2 \left(1 + \frac{1}{LR \cdot p}\right) + \frac{1}{(p+1)N_d} \sum_{LR \in V_d} \log_2 (1 + LR \cdot p)\,. \tag{8.23}$$

We notice that for $p = 1$ this equation reduces to the formula for the C_{llr} ((8.11)). In Figure 8.3 the ECE plot of our example dataset is shown.

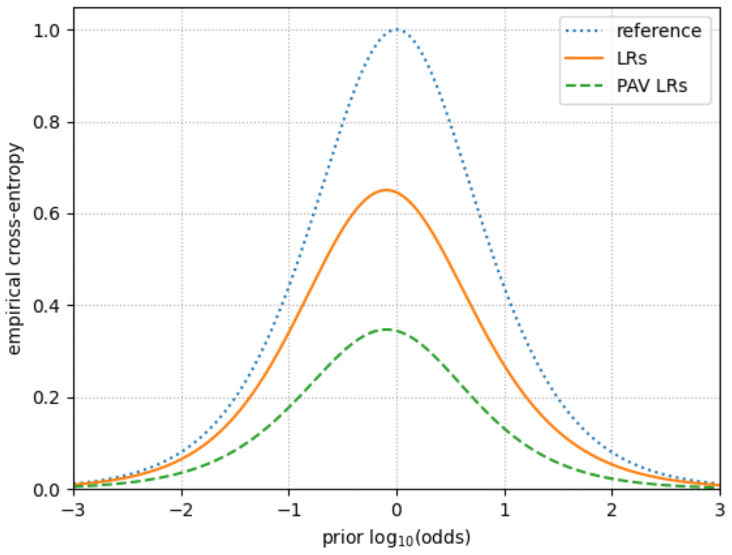


Figure 8.3: ECE plot of the example data set. The dotted blue line corresponds to the reference system, which always returns an LR of 1. The orange line represents the ECE of the computed likelihood ratios. The dashed green line corresponds to the PAV-calibrated likelihood ratios.

It contains the lines for three different systems. The dotted blue line corresponds to the reference system, which always returns an LR of 1. The continous orange line represents the ECE of the computed likelihood ratios in the set V. The dashed green line corresponds to the likelihood ratios after performing the PAV algorithm, so V^{opt}. Due to the optimality of isotonic regression for binary scoring rules the line for V^{opt} is always equal to or below the line for V. A large gap between these two lines points to poor calibration of the likelihood ratio system. If the ECE of the LRs is larger than that of the reference system for a certain range, the reference system performs better than the likelihood ratio system for those prior odds. If this range excludes $p = 1$ this cannot be seen from the C_{llr}. As a result, the ECE plot shows additional information.

Tippett plot

A Tippett plot can be used to illustrate both the discriminating power and the calibration of a likelihood ratio system. It plots the inverse cumulative density of the log-likelihood ratios under H_p (blue) and H_d (red). In Figure 8.4 a Tippett plot of the example dataset is shown. As an accurate likelihood ratio system returns LRs larger than 1 under H_p and lower than 1 under H_d we expect the line for H_p to cross the dashed line at a higher value than the line for H_d. The lines being further apart corresponds to a model having a larger discriminating power.

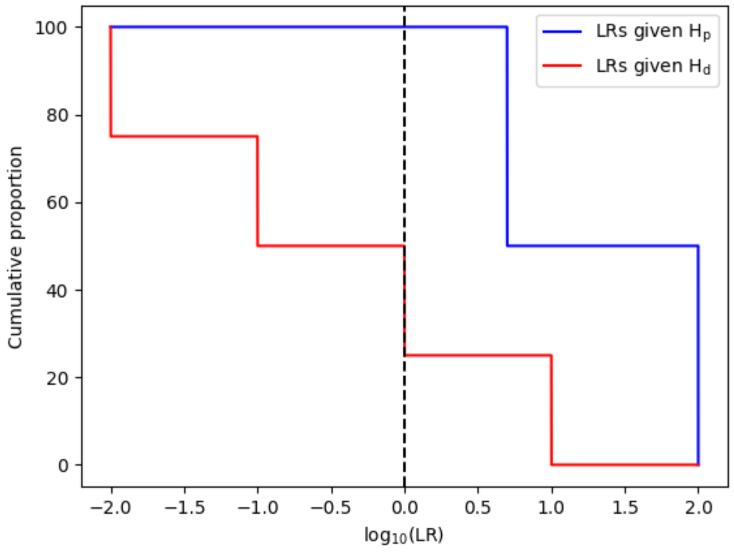


Figure 8.4: Tippett plot of the example data set. The blue and red lines correspond to the cumulative proportion of LRs under H_p (blue) and H_d (red) larger than the LR on the x-axis. The dashed line corresponds to an LR of 1, where both hypotheses are equally likely.

8.2.5. The choice of V

Now that we have introduced metrics for measuring the performance of a likelihood ratio system, we still have to decide which likelihood ratios to include at the moment we calculate these metrics. Suppose we have a total data set consisting of n authors, with m texts written by each author. To test an authorship attribution model on this set we split it into a training-and-calibration set and a validation set. When assessing a certain likelihood ratio system we must choose a certain suspect and see all other authors as the reference population. However, we do not just want to asses this for just one specific author as the suspect, as the closeness of the specific author to other authors could influence the results. Similarly, we want to assess the system for several splits of the dataset into training-and-calibration set and validation set, using k-fold cross-validation. In practice, we use 1 text per author in the validation set, and we use m-fold cross-validation, leading in total to $m \cdot n$ models being evaluated on n texts each. We write the resulting sets of likelihood ratios as $V_{i,j}$, with $i \in 1, ..., m$ corresponding with the validation texts and $j \in 1, ..., n$ to the author chosen as the suspect.

If we look at a singular set $V_{i,j}$, this includes the likelihood ratio of only 1 sample under H_p and $n-1$ under H_d. If we calculate the C_{llr}^{min} for such small sets of LRs, it is on average smaller than if we perform the same on a union of these sets. We illustrate this in an example with $m = 2$, $n = 2$ where we set the first author as the suspect. Suppose we get the following LRs

Table 8.4: Example of sets of validation LRs.

	H_p	H_d
$V_{1,1}$	1	0.1
$V_{2,1}$	10	1

Let V_1 be the union of $V_{1,1}$ and $V_{2,1}$. Then we compute the following values for our metrics

Table 8.5: Example of metrics applied to sets of validation LRs.

	C_{llr}	C_{llr}^{min}	C_{llr}^{cal}
$V_{1,1}$	0.569	0	0.569
$V_{2,1}$	0.569	0	0.569
V_1	0.569	0.500	0.069

We first note that the average C_{llr} of the two sets is equal to the C_{llr} of their union. This is a general result from the definition of the C_{llr}, under the condition that the ratio between the number of samples under H_p and under H_d is equal for all three sets. For the proof of this result, we refer to Appendix A. Secondly, we note a large discrepancy between the average C_{llr}^{min} and C_{llr}^{cal} of the two sets compared to that of their union. This is caused by the combination of the sets cancelling the perfect separation between the LRs of H_p and H_d in the individual sets, leading to a lower discriminative power. This example illustrates the significant difference between first calculating the metrics on individual sets $V_{i,j}$ and then taking the average compared with first taking the union of all sets $V_{i,j}$ and calculating the metrics on that set.

In practice, both methods have advantages and drawbacks. If we calculate the metrics on an individual set $V_{i,j}$ we measure the metrics on the validation results of an individual model, giving a good representation of its performance. The main drawback is that using this method we would perform isotonic regression on a set of LRs with only one sample under H_p. This gives too little data for an isotonic regression fit, leading to lower values of C_{llr}^{min}. On the other hand, if we first combine all likelihood ratios in a single set V before using our metrics, we have $m \cdot n$ samples under H_p when performing isotonic regression, leading to a better fit. However, these likelihood ratios come from different models with different suspects, and it is therefore questionable to state they can be evaluated as one likelihood ratio system.

We have chosen to use the middle ground of these two extremes. We bundle the likelihood ratios of all the models with the same author as the suspect, saying $V_j = \bigcup_{i \in 1,\ldots,m} V_{i,j}$ as set on which we compute our metrics. This gives us m samples under H_p when performing isotonic regression, leading to a better fit than in the case where we use individual set $V_{i,j}$. As we combine likelihood ratios from models with the same author as the suspect, we expect the difference between the models to be smaller. Furthermore, this allows us to compare the system's performance over the different authors. Additionally, this can show us if a method performs well for all suspects, or if there are outliers for which it performs poorly.

To summarize, Figure 8.5 shows how we combine the LRs from cross-validation before we compute our metrics. We do this with each author as the suspect and use both the averages of these metrics to evaluate the overall results of different methods as well as boxplots to investigate the distribution of these values over the authors.

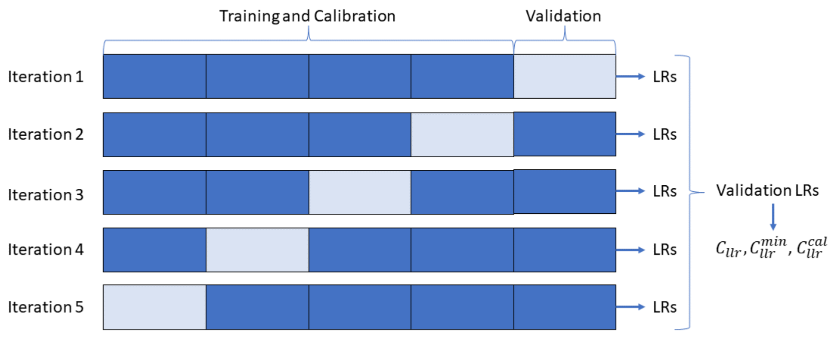


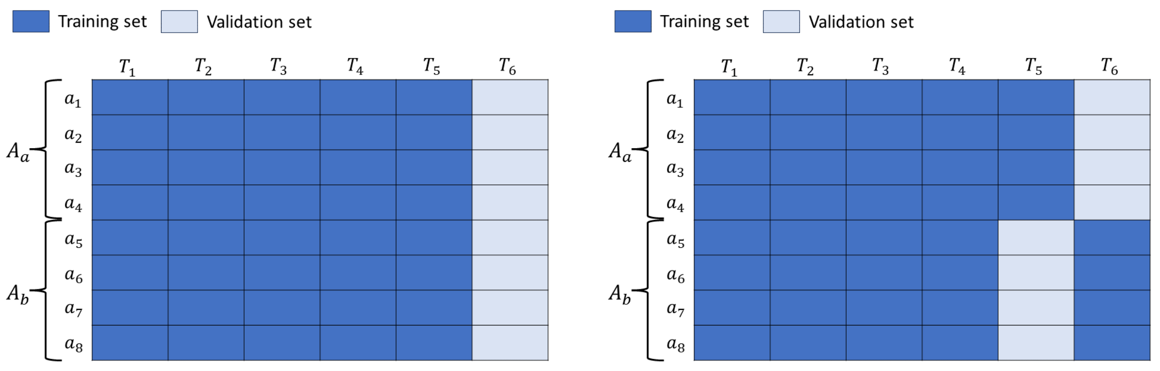
Figure 8.5: Illustration summarizing our use of cross-validation and describing when we calculate our metrics.

9 Topic and Conversation Impact

Authorship attribution models often attribute texts based on topic instead of the author's style. This is an undesirable property when used in forensic cases, as this could influence the attribution and introduce biases towards certain authors. To combat this effect, we want to be able to measure the topic-robustness of methods. The topic of texts heavily influences a model with low topic-robustness while a model with high topic-robustness attributes texts almost purely based on writing style. In this section, we propose two metrics to measure the topic-robustness of models for two different types of corpora. These metrics are the 'topic impact', which can be used on topic-controlled corpora, and the 'conversation impact', which can be used on conversational corpora.

9.1. Topic impact

Before we introduce our metric, we must explain both the standard and the confusion task. The **standard task** is extensively studied in previous research [3, 51] and consists of an intuitive way to ensure that the topic is excluded from the attribution reasons during authorship attribution. Suppose we have texts of n authors, $a_1, ..., a_n$, about N_t topics, where each author has written about each topic. For both tasks, the authors are split into two groups, A_a and A_b of the same size. In the standard task, we use all the texts about $N_t - 1$ topics as training texts and attribute the texts about the remaining topic. As the model has previously not seen any texts about the last topic, the validation texts are unlikely to be attributed based on their topic. In Figure 9.1a a schematic of this task can be seen.



Training set
Validation set
T_1 T_2 T_3 T_4 T_5 T_6
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8
A_a
A_b

(a) Standard case **(b)** Confusion case

Figure 9.1: Illustration showing the distribution of texts over the training and validation in the standard and confusion case for a topic-controlled dataset. The y-axis shows the different authors, while the x-axis shows the different topics. Dark blue squares represent texts in the training set, and light blue squares represent texts in the validation set.

Secondly, we study the **confusion task**, first introduced by Altakrori et al in 2021 [1]. In this task, for $N_t - 2$ topics all texts are added to the training set. For the remaining two topics, which we will call T_a and T_b, we divide them such that for T_a the texts written by authors in A_a are added to the validation set while the texts written by A_b are added to the training set. The texts about topic T_b are divided the other way around. The texts written by A_b are added to the validation set and those written by A_a to the training set. A schematic of this task can be seen in Figure 9.1b. Here $T_a = T_6$ and $T_b = T_5$.

Suppose a text from the validation set covering topic T_a, written by an author from A_a, is attributed to an author from A_b in the confusion task. This might be caused by the fact the texts about topic T_a from the author in A_b were present in the training set. To quantify this effect we first divide the attribution of the texts in the validation set into three groups, giving the following scores:

$s_c =$ # of texts attributed to the correct author, e.g. from a_i to a_i,
$s_s =$ # of texts attributed to a different author from the same group, e.g. from $a_i \in A_a$ to $a_j \in A_a \setminus a_i$,
$s_o =$ # of texts attributed to a different author from the other group, e.g. from $a_i \in A_a$ to $a_j \in A_b$.

These groups can be seen visually as in Figure 9.2.

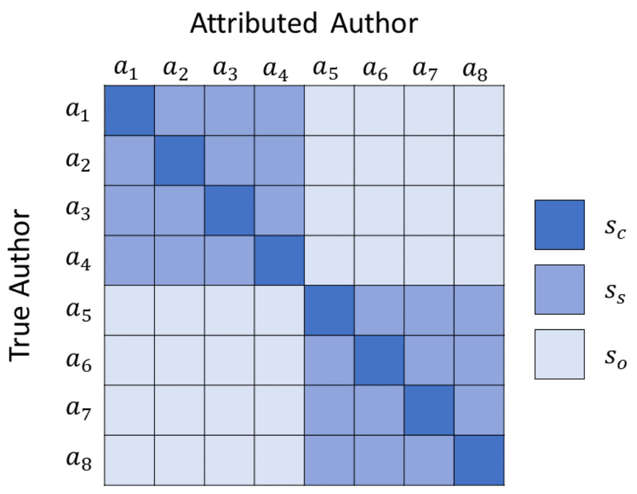

Attributed Author
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8
True Author
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8
s_c
s_s
s_o

Figure 9.2: Illustration showing in which a text from author a_i is counted as when it is attributed to author a_j in the topic impact case.

In the confusion task, texts from other authors about the same topic as the validation texts are present in the training set. As a result, we expect that for models that have a low topic-robustness, the number of attributions to s_o increases for the confusion task in comparison with the standard task. Therefore, to measure the topic impact I_t, we now subtract the percentage of texts falsely attributed to the other group in the standard case from the percentage of texts falsely attributed to the other group in the confusion task. In formula form, we write this as

$$I_t = \frac{s_o^c}{s_c^c + s_s^c + s_o^c} - \frac{s_o^s}{s_c^s + s_s^s + s_o^s}. \tag{9.1}$$

The superscripts s and c correspond with the standard and confusion tasks. With the topic impact, we can measure how much topic influences the attributions made by a model. If $I_t = 0$, it is completely independent of topic influence, while if $I_t = 1$ it changes from attributing all texts to authors in the same group to attributing all texts to an author from the other group.

9.2. Conversation impact

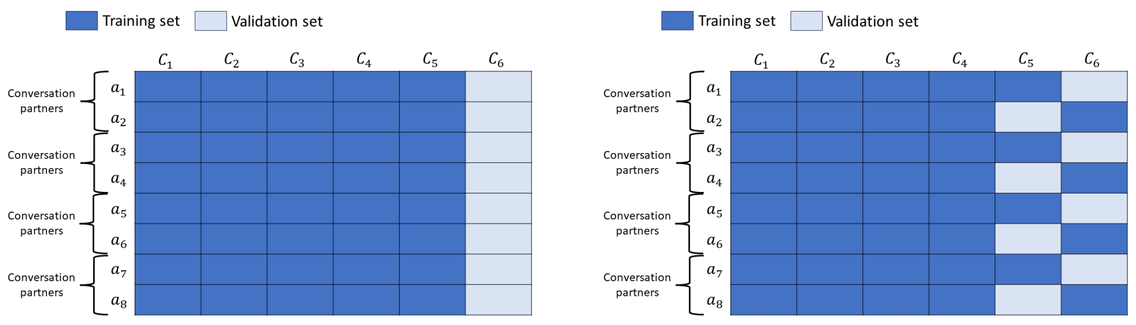
Suppose Alice and Bob have a conversation, for example, via chat messages, e-mail or on the phone. If Alice is talking to Bob about football it is quite unlikely that Bob only replies by talking about painting. As a result, we suppose that both halves of the same conversation, created by the two conversation partners, cover the same topic or topics. In conversational corpora, texts come from conversations in which two or more people write to each other, we can thus combine all messages per conversation partner into a **conversation half** and expect both conversation halves of the same conversation to cover the same topics. We can apply this to define a metric that allows us to study the impact of this conversational structure on the performance of AA methods, the conversation impact.

We suspect that both halves of a conversation cover the same topics. This would allow us to test the robustness of authorship attribution methods on datasets that are not topic-controlled but do include a conversational structure, like chat messages or transcriptions of phone calls. To test this theory we want to study the correlation between topic impact and what we will call conversational impact.

Suppose we now have n_p author pairs with each n_c conversations between the pair of authors. If we number these authors such that author a_{2i-1} has conversations with author a_{2i}, $i \in 1, ..., n_p$ we can create a standard and confusion task similar to the topic-controlled dataset. We again create sets A_a and A_b, but now such that $a_{2i-1} \in A_a, a_{2j} \in A_b$ for $i \in 1, ..., n_p$. As a result an author from A_a always has a conversation with an author from A_b.

In the standard case, Figure 9.3a, all but one of the authors' conversation halves are added to the training set, while the last conversation half is added to the validation set. This way, the conversation half of the authors' conversation partner corresponding to the text in the validation set is also in the validation set.

In the confusion case, it is ensured that the partners' conversation half corresponding to the texts in the validation set are in the training set. If conversation C_i is in the validation set for all odd authors, we add conversation C_{i-1} to the validation set for all even authors. This is illustrated in Figure 9.3b.


Training set
Validation set
C_1 C_2 C_3 C_4 C_5 C_6
Conversation partners
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8

(a) Standard **(b)** Confusion

Figure 9.3: Illustration showing the distribution of texts over the training and validation in the standard and confusion case for a dataset of conversations. The y-axis shows the different authors, while the x-axis shows the different conversations. Dark blue squares represent texts in the training set, and light blue squares represent texts in the validation set.

In the confusion case, the model has been trained on the other conversation half of all texts in the validation set. As a result, we now expect more attributions to the conversation partner in the confusion case than in the standard case. We use this expectation as the basis for the metric we call **conversation impact**, I_c. We again subdivide all attributions of the texts in the validation set into three groups, but we slightly redefine these groups compared to the topic impact case:

s_c = # of texts attributed to the correct author, e.g. from a_i to a_i,
s_p = # of texts attributed to the authors partner, e.g. from a_i to a_{i+1} for odd i,
s_r = # of texts attributed to a different author other than the authors' partner, e.g. from a_i to $a_j \notin \{a_i, a_{i+1}\}$ for odd i.

These groups are illustrated in Figure 9.4. We expect that in the confusion task, the number of attributions to the conversation partners is larger, so s_p is relatively larger than in the standard task. This is the basis for the conversation impact, which is defined as

$$I_c = \frac{s_p^c}{s_c^c + s_p^c + s_r^c} - \frac{s_p^s}{s_c^s + s_p^s + s_r^s}. \tag{9.2}$$

The superscripts c and s again refer to the confusion and standard task. With the conversation impact, we can measure how much the other half of a conversation influences the attributions made by a model. If $I_t = 0$, the model is completely independent of the influence of the other half of conversations, while if

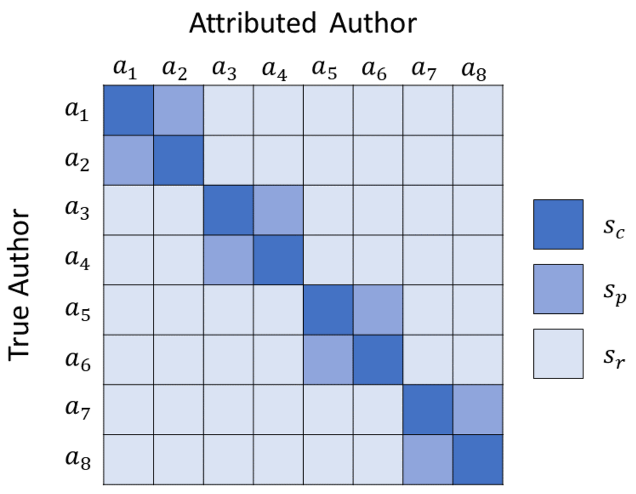


Figure 9.4: Illustration showing in which a text from author a_i is counted as when it is attributed to author a_j in the conversation impact case.

$I_t = 1$ it changes from attributing all texts either correctly or to an author that is not the author's partner to attributing all texts to the author's partner.

In general, we expect a higher relative percentage of false attributions to the author's partner compared to the other authors even in the standard case. This is due to people being more likely to speak with people of similar backgrounds and people copying phrases that have been said to them, which is an effect that might occur even over multiple conversations. However, due to taking the difference of the wrong attributions to the author's partner between the confusion and the standard case, we expect these effects to be removed while the effect of the topic of the other conversation half remains.

9.3. Practical implementation

When calculating the topic or conversation impact we use k-fold cross-validation, with k equal to the number of topics or conversations, respectively. During the confusion task, if T_i or C_i is the topic or conversation for which the texts are in the validation set for A_a we use the texts corresponding to T_{i-1} or C_{i-1} as the validation texts for A_b. This way every text is in the validation set precisely once. When for topic T_1 or conversation C_1 the texts are the validation texts for the authors in A_a, we have that the texts corresponding with T_{n_t} or C_{n_c} are in the validation set for the authors in A_b.

The datasets we use for investigating the correlation between topic and conversation impact are the topic-controlled corpus abc_nl1 and the conversational corpus FRIDA. We chose these datasets because they both consist of correctly spelled Dutch and therefore abc_nl1 is more comparable to FRIDA than RFM, in which many more deviations from the standard spelling are used. abc_nl1 consists of 8 authors with each 6 texts, where each text is about one topic. For an as equal comparison as possible we use a reduced version of FRIDA with 8 authors and 6 conversations per author.

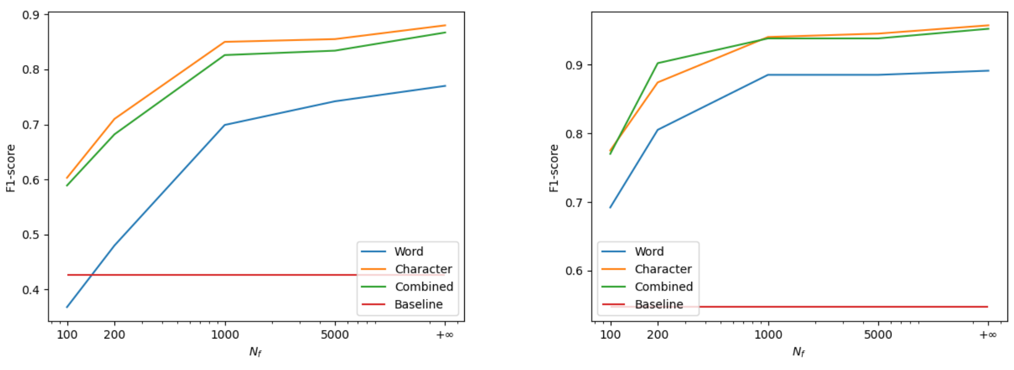
10 Results and Discussion

In this chapter, we will cover all results of our research. We divide the results into three parts, following our three main research aims. We start with the results of a variety of models when using metrics from computational authorship attribution. Using these results we choose a subset of these models for which we study the results of their corresponding likelihood ratio systems, using validation methods from the forensic sciences. We will compare the chosen methods on the forensic authorship attribution task and explain possible causes of performance differences. Lastly, we will show the results of our analysis into the impact of both conversation and topic on authorship attribution methods and their possible correlation. For the readability of this chapter, we will present some of our results in the form of figures. Tables containing the values for these plots can be found in Appendix B. Almost all code that was used to create the results presented in this chapter is publicly available at `https://github.com/wouterhajer/Authorship_Attribution`. Only the code to create the corpus dataframe for RFM has been excluded.

10.1. Computational authorship attribution

10.1.1. Feature vector and SVM models

In Figure 10.1 the 8-fold cross-validated F_1-scores of the SVM-based models against the number of unmasked words, N_f, are shown on the datasets RFM and FRIDA. The baseline is included as the horizontal red line, which is independent of N_f. The blue, orange and green lines represent models using words, characters or a combination as feature vectors, respectively.



(a) RFM **(b)** FRIDA

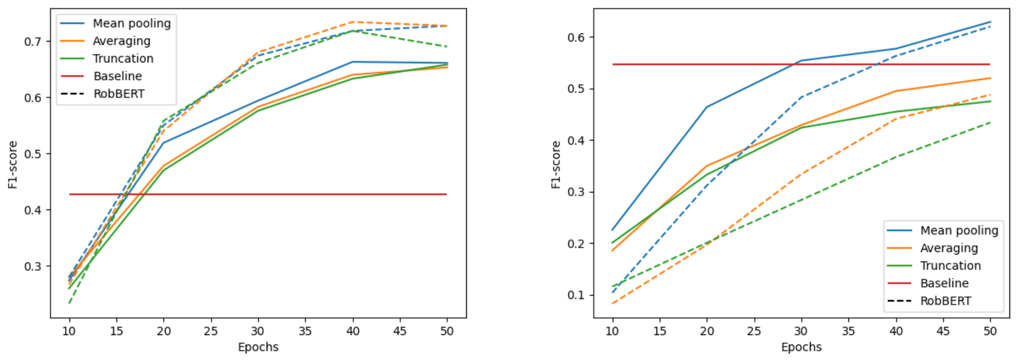
Figure 10.1: F_1-score of three types of SVM models against the number of unmasked words, N_f, on the datasets RFM and FRIDA. In red the performance of the baseline is included, which is independent of N_f.

Firstly, we notice that almost all models significantly outperform the baseline. In both images, a trend can be seen where the performance of all models increases as the number of unmasked words increases. This effect is the strongest at the start when the N_f is increased from 100 to 1000. For further increases in the N_f the curve flattens. The word n-gram model has a significantly lower performance than the other models on both datasets. The combined model performs similarly to the character model, so no significant information seems to be added by combining the word and character models. As a result, we will focus on the character-based model in our further results. We will continue to study the same range of N_f, as the choice of N_f significantly influences the topic and conversation impact, as we will show further in this chapter.

One possible explanation for the large gap between the performance of the word and character model is that the character model captures additional information, due to the inclusion of capitalization and punctuation marks, which cannot easily be included in word n-gram models. This, however, cannot explain the difference in performance on the FRIDA dataset. In this dataset, all authors have the same punctuation and capitalization, as this dataset consists of transcriptions of phone calls, where punctuation and capitalization are added consistently by the transcribers. Another possible explanation would be that character n-grams capture additional information about similar words, even when only a few words are left unmasked. An example of this could be that the words “jij, hij, zij, wij”, (you, he, she/they, we), all include the trigram “ij ”, and are used in similar sentence structures.

10.1.2. BERT-based models

In Figure 10.2 8-fold cross-validated F_1-scores of the BERT-based models against the number of finetuning epochs are shown for the datasets RFM and FRIDA. The baseline model is included as the horizontal red line, which is independent of the number of epochs. The dashed lines represent models based on RobBERT, while the continuous lines represent models based on BERTje. The blue, orange and green lines represent the mean pooling, averaging and truncation strategies for dealing with the token limit, respectively.



(a) RFM **(b)** FRIDA

Figure 10.2: 8-fold cross-validated F_1-score of six types of BERT-based models against the number of epochs they were finetuned on the datasets RFM and FRIDA. In red the performance of the baseline is included, which is independent of the number of epochs. All dashed lines represent RobBERT-based models, while all continuous lines correspond to BERTje-based models.

In both subfigures, the performance increases with longer finetuning, but the curve flattens around 40-50 epochs. We even notice a slight decrease in performance for some models when comparing their results after 50 epochs with finetuning for only 40 epochs on the RFM corpus, possibly indicating that the model starts overfitting on the finetuning data. We see that on the FRIDA dataset, the mean pooling strategy outperforms the other two strategies, while on the RFM dataset, the distance between the three strategies is much smaller. On the FRIDA data set the mean pooling strategy is also the only strategy that outperforms the baseline, after around 40 epochs of finetuning. On the RFM corpus, all models outperform the baseline after only 20 epochs of finetuning.

Another interesting observation from Figure 5.1 is that on RFM the RobBERT-based models outperform the BERTje-based models, while the opposite can be seen on the FRIDA corpus. Additionally, we see that on the RFM corpus, all models start with an equivalent performance after 10 epochs of finetuning and then the RobBERT-based models see a steeper increase in performance compared to the BERTje-based models. The same steeper performance increase for RobBERT-based models is seen on the FRIDA corpus, although here the initial performance of the RobBERT-based models is with an F_1-score of around 0.1, lower than that of the BERTje-based models, which have an F_1-score of around 0.2 after 10 epochs of finetuning. Due to the steeper incline of the RobBERT-based methods, this difference decreases after more epochs of finetuning.

A possible explanation for the result that BERTje-based models outperformed RobBERT-based models on FRIDA, while we saw the opposite on RFM, could be the pre-training corpora used for both underlying models. BERTje was pre-trained on a corpus consisting of mostly Wikipedia articles, newspaper articles and books. As a result, this includes mostly correctly written Dutch, which might correspond more closely to the transcribed phone conversations than the chat messages. RobBERT, on the other hand, was pre-trained on a corpus consisting of scraped internet pages, which also includes online forums on which the writing quality might more closely resemble that of the chat messages in RFM.

The result that the different strategies for dealing with the token limit perform similarly on RFM but not on FRIDA can likely be explained by the shorter texts in RFM. The RFM corpus averages 303 tokens per text compared to 774 tokens per text in FRIDA when both corpora are tokenized with RobBERT. As a result, the token limit of 512 tokens is only reached in 12.7% of the texts in RFM compared to 93.0% of texts in FRIDA. With the tokenizer of BERTje, similar token lengths were seen. As a result, the impact of the different strategies is more noticeable on FRIDA. As our mean pooling strategy outperforms the other strategies on FRIDA while having a similar performance on RFM, we will focus on this strategy in our further results.

10.1.3. Comparison

If we compare the performance of the BERT-based models with SVM models we notice that the SVM models outperform the BERT-based models. The peak F_1-score of the SVM models is 0.880 on RFM and 0.957 on FRIDA, compared to 0.727 and 0.629 for BERT-based models. So while the mean pooling BERT-based models outperform the baseline after 50 epochs of finetuning, they still get outscored by the SVM models. Similar results were found in previous comparisons on English corpora with a low amount of training text per author [37, 54]. In both studies, only on corpora with on average more than 100,000 characters of training texts per author BERT-based methods outperformed character n-grams. As our corpora consist of an average of 2,000-5,000 characters per author, our results agree with the previous literature. This also shows that the performance difference between these methods on English corpora is transferable to Dutch corpora.

We also notice that all BERT-based models perform better on the RFM corpus than on FRIDA, while all other methods perform better on FRIDA. We would have expected all methods to perform better on FRIDA, as it contains an average of 509 words per text compared to 239 in RFM, and therefore has more information per message. On the other hand, RFM does include punctuation and spelling mistakes as made by the author, which FRIDA does not as it consists of transcriptions of spoken text. It could be possible that BERT-based models lean more heavily on these written characteristics than our feature based models, leading to this difference.

10.1.4. Influence of text length, number of texts and number of authors

We additionally studied the influence of several factors on the F_1-score of the previously selected models. The select models are character n-gram SVM models with $N_f \in \{100, 200, 1000, 5000, \infty\}$, a RobBERT- and a BERTje-based model with the mean pooling strategy and 50 epochs of finetuning as well as the baseline. We tested the variables “text” length and “number of texts” with the RFM dataset, as this contains more texts per author, giving us more freedom to vary these variables. As FRIDA consists of more authors, we study the influence of the number of authors with FRIDA.

Text length

To create texts of varying lengths we varied the minimal number of messages we use to construct a text in RFM. Next to the standard version of RFM with a minimum of 10 messages, we studied a minimum of 15 messages and a minimum of 5 messages. In Table 10.1 the results are shown, with the highest value for each column highlighted in boldface.

The general trend we see in Table 10.1 is that longer texts increase the performance of all models. This is to be expected, as longer texts contain more information, and therefore are likely to contain

Table 10.1: Cross-validated F_1-scores of 8 models on the authorship attribution task on the RFM corpus with texts of varying average length. The highest value of each column is shown in boldface.

Model type	Minimal # of messages Avg # of words	5 150.7	10 238.7	15 293.6
	Model specification			
Character SVM	$N_f = \infty$	0.621	**0.853**	**0.940**
	$N_f = 5000$	**0.628**	0.834	0.928
	$N_f = 1000$	0.586	0.826	0.885
	$N_f = 200$	0.453	0.682	0.764
	$N_f = 100$	0.403	0.589	0.705
BERT-based, mean pooling, 50 epochs	RobBERT	0.581	0.727	0.795
	BERTje	0.538	0.661	0.753
Baseline	Baseline	0.306	0.427	0.544

more identifying information about the author. An interesting observation is that for the shortest text length, the SVM model with $N_f = 5000$ outperforms the model without masking. It must be noted that this difference is so small that this could also be caused by random variations.

The BERTje-based model, the RobBERT-based model and the SVM model with $N_f = 200$ have a comparable performance with the minimum of 10 messages per text, with F_1-scores of 0.727, 0.661 and 0.682, respectively. If the text length is lowered to a minimum of 5 messages per text, we see that the performance of the SVM model $N_f = 200$ drops to 0.453, compared to 0.581 and 0.538 for the RobBERT- and BERTje-based models, respectively. So the impact of smaller texts is larger on SVM models than on BERT-based models. If we look at the effects of longer text on these three models we notice that the relative performance is similar to the standard version of RFM, with F_1-scores of 0.795, 0.753 and 0.764 for the RobBERT-, BERTje-based and SVM model, respectively.

We are not completely sure why the effect of shorter texts is smaller on BERT-based models. From the literature, we expected that as BERT-based models only outperform SVM with large amount of training texts, their performance would deteriorate the fastest with short training texts. Our results show the opposite of our expectations. A possible cause for this effect could be that for short texts the frequency of individual features is very low, as not many combinations occur more than once. As a result, a situation could happen that a common word has not been written by the author in his training texts, but does occur in the validation text, influencing the attribution. This is much more likely to happen for shorter training and validation texts. Due to the use of embeddings, the BERT-based models might be less influenced by the occurrence of a single common word not used in the training texts.

Number of texts

In Table 10.2, the cross-validated F_1-scores of the 8 selected models are shown against the number of included texts per author. As we use leave-one-out cross-validation, the number of training texts is 3, 7 and 15 for the different columns, respectively.

Table 10.2: Cross-validated F_1-scores of 8 models on the authorship attribution task on the RFM corpus with varying number of texts per author. Cross-validation is done with one test text, meaning that there are 3, 7 and 15 training texts respectively. The highest value of each column is shown in boldface.

Model type	# of texts per author	4	8	16
	Model specification			
Character SVM	$N_f = \infty$	**0.692**	**0.853**	**0.956**
	$N_f = 5000$	0.647	0.834	0.893
	$N_f = 1000$	0.602	0.826	0.911
	$N_f = 200$	0.487	0.682	0.803
	$N_f = 100$	0.377	0.589	0.728
BERT-based, mean pooling, 50 epochs	RobBERT	0.368	0.727	0.807
	BERTje	0.307	0.661	0.821
Baseline	Baseline	0.266	0.427	0.551

As we would expect, the performance of all models increases when the number of training texts is increased. This effect is the strongest for the BERTje-based model, which see a larger absolute increase in the F_1-score when going from 4 to 8 and from 8 to 16 texts per author than all other models, with differences of 0.354 and 0.160, respectively. The RobBERT-based model also shows a large increase when going from 4 to 8 texts per author. That the effect of increasing the number of training texts is the largest on the BERT-based models is what we would have expected based on the literature. However, at 16 texts per author this effect is not large enough to outperform SVM models, the best of which attains an F_1-score of 0.956, compared to 0.821 for the best BERT-based model.

We also wanted to highlight the performance of the SVM model without masking with a small number of training texts. With only three training texts per author, it reaches a F_1-score of 0.692 on a set of 50 authors, where random attribution has an average F_1-score of 0.020. This shows how much information can be obtained from even a few texts per author and how authorship attribution can still be useful in cases with little training material available. This is extra relevant in the forensic context, where data scarcity is a common issue.

Number of authors

In Table 10.3, the cross-validated F_1-scores of the 8 selected models are shown against the number of authors included in the analysis. In this table we notice an unexpected pattern for the SVM models. We expected that the F_1-score decreases when the number of authors increases, as there is a higher chance of two authors having similar writing styles. Instead, for all SVM models we notice that the performance is stable or even increasing when the number of authors increases.

Table 10.3: Cross-validated F_1-scores of 8 models on the authorship attribution task on the FRIDA corpus with a varying number of included authors. The highest value of each column is shown in boldface.

Model type	# of authors	10	20	50	100
	Model specification				
Character SVM	$N_f = \infty$	**0.950**	0.950	**0.957**	**0.956**
	$N_f = 5000$	0.919	**0.951**	0.945	0.953
	$N_f = 1000$	0.935	0.914	0.940	0.940
	$N_f = 200$	0.825	0.851	0.874	0.885
	$N_f = 100$	0.779	0.795	0.775	0.777
BERT-based, mean pooling, 50 epochs	RobBERT	0.603	0.624	0.620	0.634
	BERTje	0.740	0.665	0.629	0.609
Baseline	Baseline	0.669	0.595	0.547	0.477

We think this effect has two main causes. Firstly, as we use a One-vs-Rest classifier, the "rest" class of each SVM has a larger number of samples when more authors are included in the analysis. This could improve these individual classifiers, leading to a better overall classification. This is also the reason why we chose the One-vs-Rest classifier over a One-vs-One classifier. For example, for the character SVM model with $N_f = 200$, the F_1-score using a One-vs-One classifier is 0.742 for 10 authors, against 0.477 for 50 authors. We can compare this to the results of the same model with a One-vs-Rest classifier, namely 0.825 for 10 authors, against 0.874 for 50 authors. We notice that the One-vs-One classifier decreases significantly in performance when the number of authors increases, while the performance of the One-vs-Rest classifier rises slightly.

However, this cannot be the full explanation, as the baseline model also uses One-vs-Rest classification as its multiclass strategy. We see a decrease in accuracy for the baseline model as the number of authors increases. This difference between the baseline and the SVM models could be explained by the number of features in the feature vector. The baseline has 100 features, against 30,000 for the character SVM model without masking. The baseline might perform worse in separate 100 authors, as it can only compare them based on 100 features. Even the character SVM model with $N_f = 100$ contains 5800 distinctive features and might therefore be impacted less by the increase in authors.

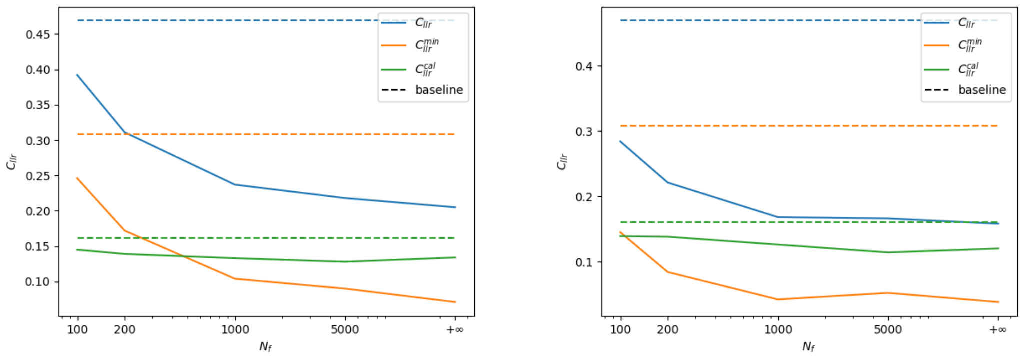
We notice that when the number of authors is increased from 10 to 100 the F_1-score of the BERTje-based model decreases significantly, from 0.740 to 0.609, while it increases slightly for the RobBERT-based model, from 0.603 to 0.634. We are unsure what could cause this big difference, as the architecture of these models is the same.

10.2. Forensic authorship attribution

10.2.1. Log-likelihood ratio cost

Support vector machines

In Figure 10.3, the log-likelihood ratio cost (C_{llr}) and its two subdivisions, the C_{llr}^{min} and C_{llr}^{cal}, of the SVM models with character n-grams as features are plotted against the number of unmasked words for the RFM and FRIDA corpora. Additionally, the value of these three variables for the baseline model is included by the horizontal dotted lines, independent of N_f. Remember that a lower C_{llr} implies a better-performing likelihood ratio system. We notice that our models significantly outperform the baseline and that their performance increases when N_f increases.



(a) RFM **(b)** FRIDA

Figure 10.3: C_{llr}, C_{llr}^{min} and C_{llr}^{cal} of character-based SVM models against the number of unmasked words, N_f, on the datasets RFM and FRIDA. The performance of the baseline is included by the dotted lines, which are independent of N_f.

In the breakdown of the C_{llr} into C_{llr}^{min} and C_{llr}^{cal} we see that for larger N_f the C_{llr}^{min} decreases sharply, while the C_{llr}^{cal} remains almost constant. The curve for the C_{llr}^{min} looks similar, but with the opposite direction, to the F_1-score we saw in Figure 10.1. Namely, it first decreases sharply and the curve flattens after about $N_f = 1000$.

As all compared likelihood ratio systems consist of an SVM with cross-calibration using KDEs we expected their results to differ in discriminative power and calibration quality. We also see this in Figure 10.3, where the line of C_{llr}^{cal} is almost constant. This shows that the separation of the C_{llr} into its subdivisions performs well. The lowest C_{llr} we found, reached by the model without masking, is 0.158. Recall that the minimal reachable value using an ELUB bounder with our dataset size was 0.088. So the majority of the remaining C_{llr} of the best-performing model is caused by the dataset size. As the C_{llr}^{min} of this model is 0.033, only small gains can still be achieved by further improving the discriminative power of the underlying models.

The best method to lower the total C_{llr} would be to increase the number of texts under H_p, as this would lead to better-calibrated KDEs and decrease the lower bound on our LRs. In practice, this is often impossible, as you are constrained by the number of known texts written by the suspect in forensic authorship attribution cases.

BERT-based models

Table 10.4 includes the same metrics for the BERT-based models using the mean pooling strategy and 10 epochs of finetuning. We recall that we finetune for fewer epochs compared to the computational authorship attribution case due to the lower number of classes, which are now limited to H_d and H_p. It must be noted that these values have been computed without cross-validation due to computational constraints. Using a NVIDIA GeForce GTX 1080 GPU these computations took 2 days per result, which cross-validating would have extended to 16 days per result.

Although we do not use cross-validation, the C_{llr} of the BERT-based models can still be compared to the results of the SVM models due to the result that the average C_{llr} of several subsets is the same as the C_{llr} of the union of these subsets, which we proof in Appendix A. We should, however, keep a slightly larger uncertainty in mind, due to the lack of cross-validation.

The C_{llr}^{min} and C_{llr}^{cal} computed for the BERT-based models cannot be directly compared to the corresponding results of the other models. This is due to our choice to perform PAV re-calibration on the union of all likelihood ratio results computed on the validation set with the same author as suspect, which we motivated in Chapter 8.2.5. Without cross-validation, this set consists of 50 likelihood ratios, of which only 1 is under H_p, compared to 400 LRs, of which 8 are under H_p with cross-validation. In general, we expect the C_{llr}^{min} to increase and the C_{llr}^{cal} to decrease when we would perform the same analysis with cross-validation. To allow for better comparisons, we also include the result of the baseline model if no cross-validation is used in Table 10.4.

Table 10.4: C_{llr}, C_{llr}^{min} and C_{llr}^{cal} of a RobBERT and BERTje-based model with mean pooling strategy and 10 epochs of finetuning. Baseline values without cross-validation are also included for comparison. The lowest value of each column is shown in boldface.

Dataset	RFM			FRIDA		
Model	C_{llr}	C_{llr}^{min}	C_{llr}^{cal}	C_{llr}	C_{llr}^{min}	C_{llr}^{cal}
RobBERT	0.746	0.390	0.355	0.811	0.407	0.404
BERTje	0.664	0.281	0.383	0.637	0.234	0.403
Baseline	**0.470**	**0.148**	**0.322**	**0.474**	**0.135**	**0.340**

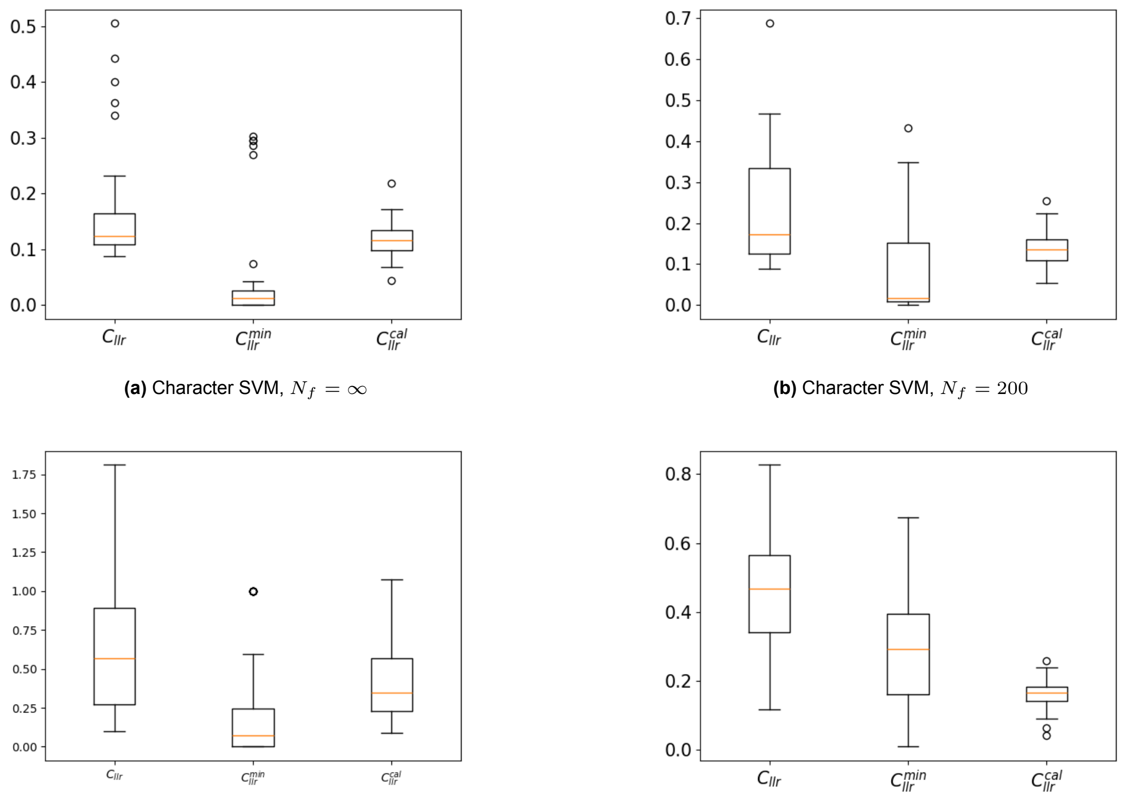
From Table 10.4 we see that with C_{llr}'s of 0.664 and 0.637, the BERTje-based model outperforms the 0.764 and 0.811 of the RobBERT-based model on RFM and FIDA, respectively. However, both models underperform the baseline significantly, which reaches 0.470 on RFM and 0.474 on FRIDA. This is surprising, as the BERT-based models did outperform the baseline in F_1-score. If we look at the breakdown into C_{llr}^{min} and C_{llr}^{cal} we see that the baseline model outperforms the BERT-based models for both metrics, but that this gap is the largest for the C_{llr}^{min}.

It is unexpected that the discriminative power of the BERT-based models is significantly worse than the baseline, especially as they outperformed the baseline in F_1-score on the computational authorship attribution task. A possible explanation could be that the FAA task more closely resembles the underlying architecture of the baseline and SVM model. Remember that in the FAA task, all texts from the suspect are in one class (H_p) while all other texts belong to the H_d class. As a result, this problem is identical to one of the many classifiers created with the One-vs-Rest strategy by the SVM and baseline models in the CAA task. For the BERT-based models, in the FAA case, a different model is created by finetuning with only two labels instead of one for each author, which leads to a different architecture than in the CAA case. Therefore, the resulting SVM and baseline models in the FAA task might more closely resemble the models in the CAA task compared to the BERT-based models. This could have led to the discriminative power being better transferable between the tasks for the SVM and baseline models than for the BERT-based models.

We also notice that the C_{llr}^{cal} is larger for the BERT-based models than for the baseline. We suspect this is caused by the use of cross-calibration in combination with the randomness present in the BERT model. In the finetuning of our BERT-based models, we use dropout to reduce overfitting. As a result, finetuning the same type of BERT model on the same data twice leads to two different final models. Therefore, the change in calibration texts during cross-calibration can lead to a larger effect on the BERT-based models than that caused by just the change in training data. On the other hand, the SVM and baseline models are deterministic, so given the same training data, the same resulting model is always returned. This could cause poorer calibration for the BERT-based models compared to SVM models and the baseline. Due to the low number of training texts under H_p in our datasets, we cannot eliminate the cross-calibration step to get better results with BERT-based models. In cases with much more known texts written by the suspect, this might be a possibility. Another possible method to lower this effect might be to fix which nodes are turned off by dropout for which training text during the finetuning of each model during cross-calibration, but this requires a serious alteration in the inner working of the BERT models and was therefore outside the scope of this thesis.

Performance per author

In Figure 10.4 boxplots of the C_{llr}, C_{llr}^{min} and C_{llr}^{cal} per author are shown for four models on FRIDA. As previously mentioned, the results of the BERTje-based model in Figure 10.4c are not cross-validated and therefore not directly comparable to the other three figures, especially for the C_{llr}^{min} and C_{llr}^{cal}.



(a) Character SVM, $N_f = \infty$

(b) Character SVM, $N_f = 200$

(c) BERTje mean pooling, 10 epochs

(d) Baseline

Figure 10.4: Boxplots of the C_{llr}, C_{llr}^{min} and C_{llr}^{cal} per author of three authorship attribution methods.

For the SVM and baseline models, we notice that the C_{llr}^{cal} is similarly distributed, without large outliers. For the C_{llr} and C_{llr}^{min} we notice that the character SVM models have significant outliers, with much larger values compared to the averages. However, the maximal values of these outliers for the C_{llr}, 0.512 in Figure 10.4a and 0.695 in Figure 10.4b, are still smaller than the largest value of C_{llr} for the baseline (0.823), where no outliers are seen. Such outliers are likely caused by authors with a similar writing style to other authors.

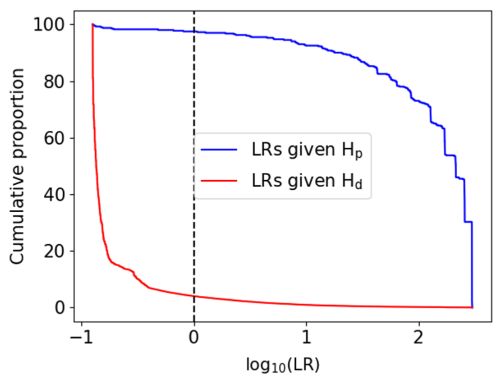
For the BERTje-based model (Figure 10.4c) we notice an outlier of the C_{llr}^{min} with a value of 1. This point represents 9 outliers plotted on top of each other, with each a value of exactly 1. This is the maximal possible value for the C_{llr}^{min} and attained when the model is recalibrated to the reference system during PAV recalibration. So for these 9 authors, the performance of the BERTje-based likelihood ratio system was so poor that the reference system would have outperformed it. We notice that for all included SVM and baseline models no individual author has a C_{llr} greater than 1. From this, we can conclude that for these models it is always beneficial to use the authorship attribution likelihood ratio system over the reference system when judging the value of evidence.

10.2.2. Tippett plots

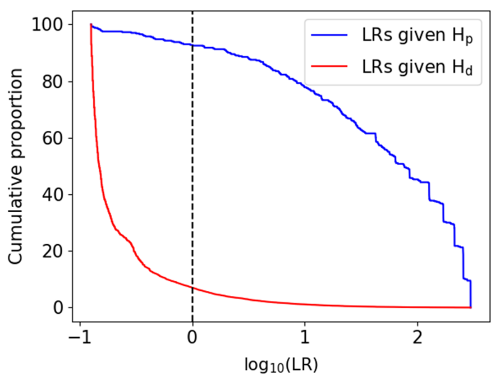
Tippett plots on RFM

To study the likelihood ratio systems more visually we will look at some of the corresponding Tippett plots. In Figure 10.5 four Tippett plots are shown, corresponding to the character SVM model with no masking, the character SVM model with $N_f = 200$, the BERTje-based model with mean pooling strategy and the baseline. These are created on the RFM dataset.

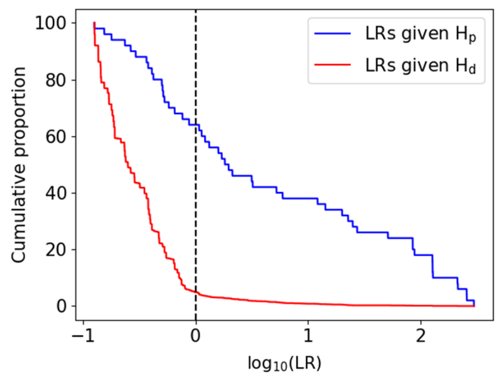
Note that the Tippett plot of the BERTje-based model in Figure 10.5c is less smooth than the other plots as fewer likelihood ratios are included due to the lack of cross-validation. However, the general trends are still comparable between all four models. We notice that the maximal and minimal attained LRs are the same in all 4 plots. This is due to the lower and upper bounds of the LRs we have calculated



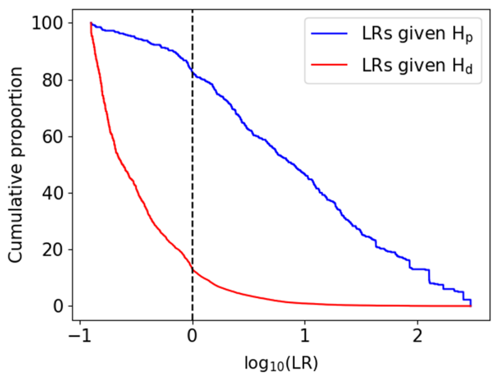
(a) Character SVM, $N_f = \infty$



(b) Character SVM, $N_f = 200$



(c) BERTje mean pooling, 10 epochs



(d) Baseline

Figure 10.5: Tippett plots of 4 types of authorship attribution models on the RFM dataset. The red and blue lines show the cumulative proportion of LRs that is smaller than that LR for the texts under H_d and H_p, respectively.

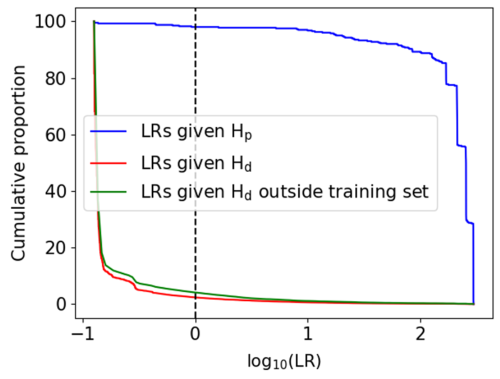
in Chapter 7.5.1. Due to the difference in the number of samples under H_p and H_d, the lower bound on the LRs is much tighter than the upper bound.

The Tippett plots of the likelihood ratio systems with a low C_{llr}, like the SVM models, differ in two ways from those with a higher C_{llr}, like the baseline and BERTje model. We notice this the best when we compare the SVM model without masking in Figure 10.5a and with the BERTje model in Figure 10.5c. Firstly there are fewer misclassifications in Figure 10.5a. This means the red line crosses the $\log_{10}(\text{LR}) = 0$ boundary lower, and the blue line higher. We notice that for the character-based SVM model without masking less than 5 % of the samples under H_p are misclassified, while this is more than 30% for the BERTje-based model. For the samples under H_d the difference in misclassifications between these two models is much smaller. Secondly, the correctly classified texts have an LR closer to $\log_{10}(\text{LR}) = 0$, meaning that the evidence strength is lower, for systems with a high C_{llr}, like the BERTje-based system. In Figure 10.5c, we see that for the BERTje-based system while more than 60% of the samples under H_p are classified correctly, only around 40% of them have an LR greater than 10. If we compare this to the SVM model without masking in Figure 10.5a, 95% of the samples under H_p are correctly classified and more than 90% of these samples also have an LR greater than 10. So next to having a higher percentage of correctly classified samples, the ratio of these correctly classified samples with a high value of evidence is also higher for a better-performing likelihood ratio system.

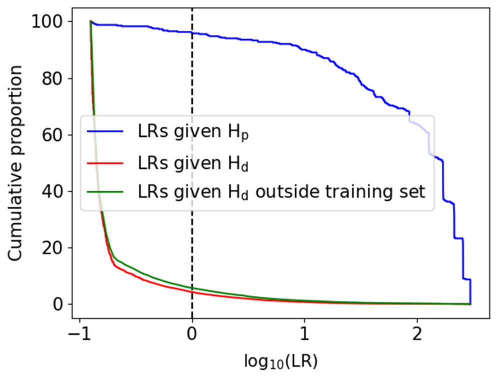
Tippett plots on FRIDA with outside authors

Another interesting sub-question is how the model handles texts from authors outside of the training set. In the hypothesis H_d the entire background population is included, but before we only looked at test texts of the 50 authors included in the training set. For the LRs to be accurate, the authors included in the training set under H_d must be a representative subset of the background population. To study these effects, we evaluate the Tippett plots on FRIDA with an additional third line, corresponding to

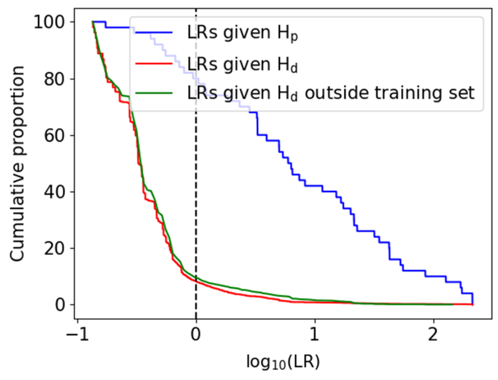
LRs generated on texts written by authors outside the training set. These plots can be seen in Figure 10.6. As these additional texts are not written by the suspect, these texts are under H_d. If the authors in the training are indeed a representative subset of the additional authors, we expect the curve to be similar to that of the H_d curve.



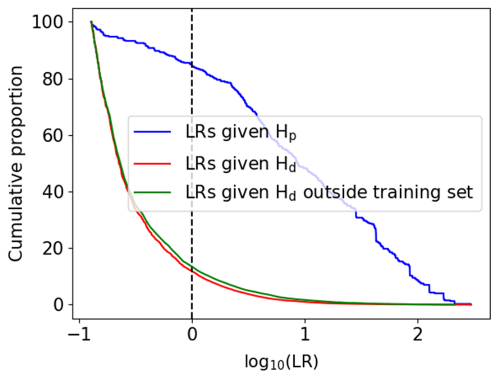
(a) Character SVM, $N_f = \infty$



(b) Character SVM, $N_f = 200$



(c) BERTje mean pooling, 10 epochs



(d) Baseline

Figure 10.6: Tippett plots of 4 types of authorship attribution models on the FRIDA dataset. The red and blue lines show the cumulative proportion of LRs that is smaller than that LR for the texts under H_d and H_p, respectively. In green the same is shown for texts by 50 authors outside of the training set, which is also under H_d.

We notice that the green line for the texts written by the additional authors stays slightly above the red line for the authors in the training set. This is not unexpected, as it is harder to classify texts by a not previously seen author as under H_d than a text by an author for which reference material is present in the training set. The difference is quite small, and it does not change the performance order of the models. Lowering the number of authors in the reference set worsens the performance of all models on texts written by authors outside of the training set, as the reference set becomes a less representative sample of the population. It is important to keep these effects in mind when constructing and validating likelihood ratio systems for forensic authorship attribution.

10.2.3. ECE plot

In Figure 10.7 the ECE plots of the baseline, two SVM methods with character n-grams as features and masking with $N_f = \infty$ or 200 as well as the BERTje-base model are shown. Additionally, the ECE plot of the reference system, where every sample is given an LR of 1, is included as the dashed blue line for comparison.

Recall that a method with a lower empirical cross-entropy is preferable. We notice that none of the lines intersect. Therefore, as the lines have the ordering reference, BERTje, baseline, char SVM ($N_f = 200$) and char SVM ($N_f = \infty$) from top to bottom, this means that the character SVM without masking strictly outperforms the character SVM method using masking with $N_f = 200$, and both outperform the baseline for all priors. The BERTje model performs worse than the baseline for all priors, but still

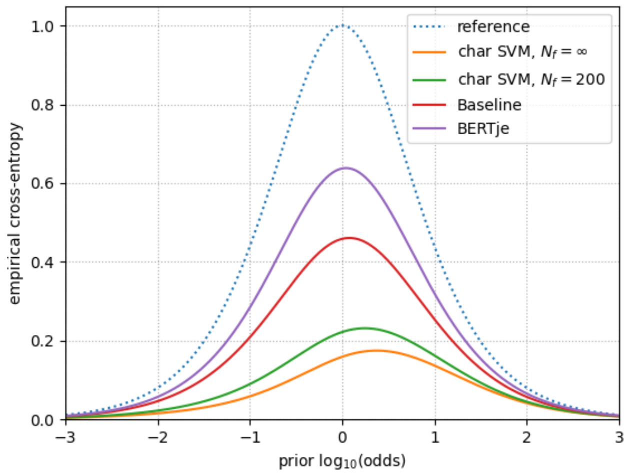


Figure 10.7: ECE plot of four different authorship attribution models. In the dashed line the ECE plot of the reference system, where every sample is given an LR of 1, is included. Results are generated on the FRIDA dataset.

outperforms the reference system.

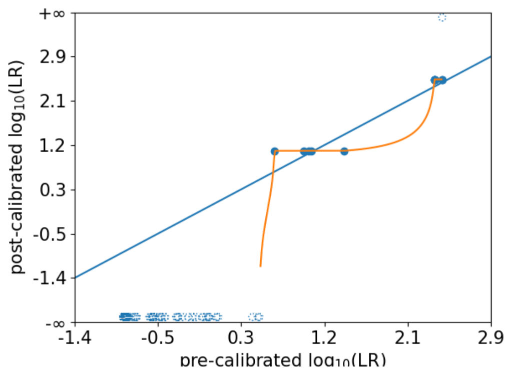
An interesting observation is that the peak of the ECE plot for the character SVM method without masking is further to the right in comparison with the other methods. As a result, the relative performance difference between this method and other methods is lower when the prior odds are higher than 1, meaning that the suspect is more likely to be the perpetrator than the background population based on the initial non-scientific evidence.

10.2.4. PAV plots

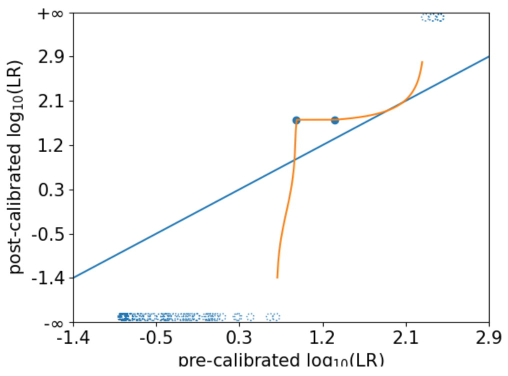
In Figure 10.8 PAV plots using the first author as the suspect are shown for three different models on FRIDA. The reason we show these plots for the results of a specific author is the choice to compute the C_{llr}^{min} and C_{llr}^{cal} per author, which we made in Chapter 8.2.5. The PAV plot of the BERTje-based model is not included, as due to the lack of cross-validation this includes only one sample under H_p, which completely determines the recalibration. As a result, we cannot show general trends based on this plot.

We notice a similar structure in all three plots, with some distinctive differences. Firstly, for both SVM methods, the largest and smallest LRs are recalibrated to positive infinity and 0 respectively, meaning that none of the samples with the highest or lowest LR were misclassified. This effect is especially significant for the low LR, whereas no samples under H_p are given an LR smaller than 1, all LR smaller than 1 are recalibrated to 0, leading to a significant number of samples far below the $x = y$-diagonal.

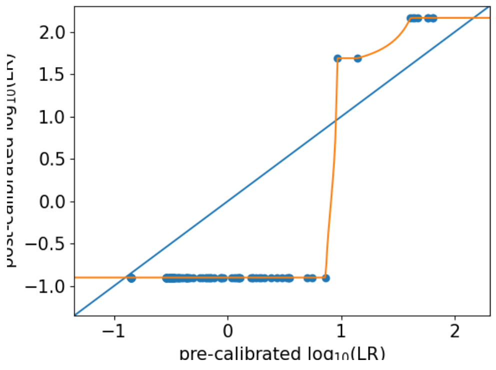
For the baseline model, we see that all values with an LR smaller than 10 are given the same value, but here this is not 0 but around 0.125. This occurs due to one sample under H_p being completely misclassified and given a low LR. Still, all points with an LR smaller than 10 pre-calibration lie below the diagonal, meaning that their LR was too high before calibration. This is the main cause of calibration error for our likelihood ratio system and is likely caused by the high bandwidth in the KDE estimation of the samples under H_p due to the low amount of data under H_p during calibration.



(a) Character SVM, $N_f = \infty$



(b) Character SVM, $N_f = 200$



(c) Baseline

Figure 10.8: PAV plots of 3 types of authorship attribution models on the FRIDA dataset.

10.3. Topic and conversation impact

To study the relation between the metrics of topic impact and conversation impact we compute these metrics for 25 different models. The chosen models are 15 SVM models, namely all the combinations of word, character and combined feature sets with no masking, or masking everything but either the 5000, 1000, 200 or 100 the most frequent words. Additionally, 10 BERT-based models are evaluated: mean pooling methods with 10, 20, 30, 40 or 50 epochs of finetuning with either RobBERT or BERTje as the starting BERT model.

We computed the conversation impact of these models on the FRIDA corpus and their topic impact on abc_nl1. In Figure 10.9, the conversation topic on FRIDA is plotted against the topic impact on abc_nl1. We notice a general trend between the conversation impact on a model and the topic impact on the model. The correlation coefficient ρ of this relation is 0.68. The interpretation of such a correlation coefficient is difficult, as we cannot expect an exact, physical law-like relation and therefore do not expect a ρ very close to 1. It could also be possible that a non-linear relation between the two impact metrics is present, although we do not directly expect this based on the metrics' definitions.

We think it is best to judge the correlation based on Figure 10.9. We can conclude that a model that slightly ($\leq$0.05) outperforms another model in conversation impact does not directly imply that it has a higher topic-robustness. However, for large (>0.05) differences in conversation impact between two models, a plausible argument can be made that the model with lower conversation impact has a higher topic-robustness.

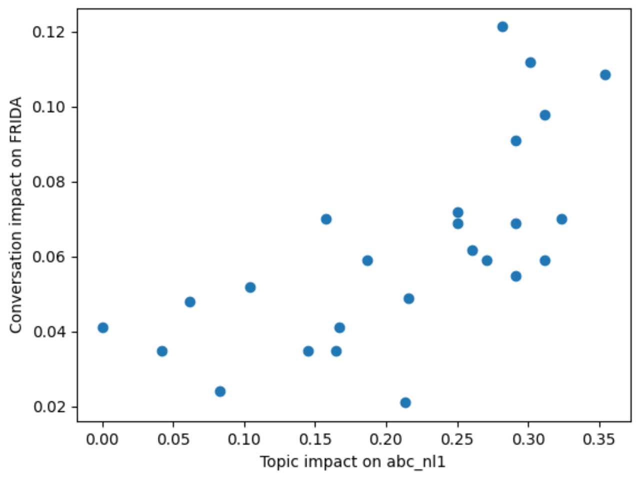


Figure 10.9: Plot of the conversation impact against the topic impact for 25 different authorship attribution.

We want to highlight two interesting additional results in the correlation between topic and conversation impact. Firstly, in Figure 10.10 the results from the SVM models are highlighted and grouped by the number of frequent words left after masking, N_f. We notice that both the topic impact and the conversation impact are larger for models with a high N_f. This corresponds with what we would expect, as the 100 or 200 most frequent words in both the frequency list created for FRIDA and the SUBTLEX-NL list used for the abc_nl1 corpus contain no topic-related words. In the SVM models with fewer masked words, more content-related words are not masked and therefore we would expect a higher topic and conversation impact.

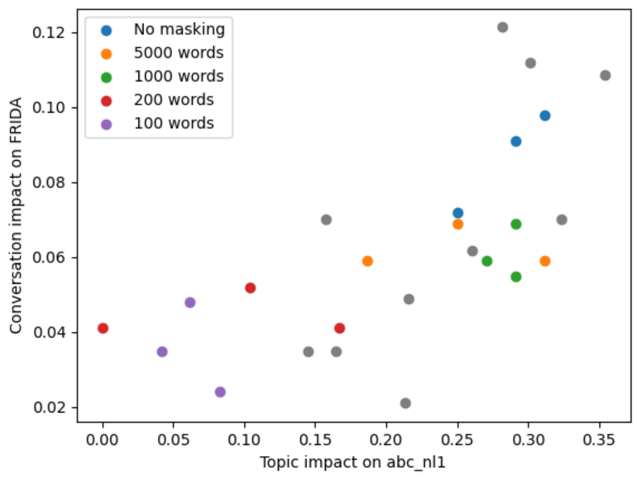
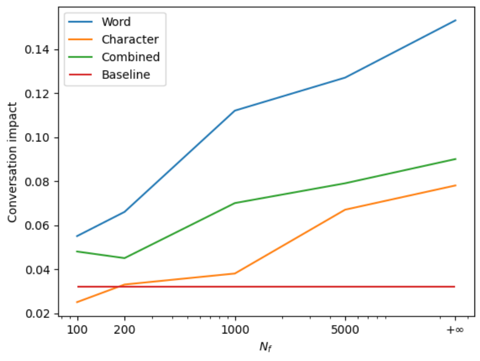
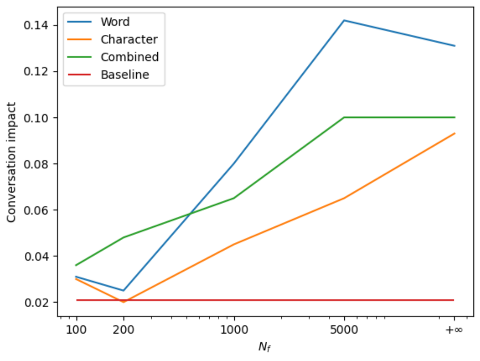


Figure 10.10: Plot of the conversation impact against the topic impact for 25 different authorship attribution. The results of the SVM models using masking are highlighted, with labels for the number of words that are unmasked.

We will now look deeper into the conversation impact on several SVM models. In Figure 10.11 the conversation impact for the word, character and combined SVM models are shown against the number of unmasked words, for both RFM and FRIDA. Also, the baseline is included as the continuous red line, independent of N_f. We notice that in both corpora the models using character n-grams as features have a significantly lower conversation impact than the models using word n-grams. The performance of the combined SVM models is in between the other two types of SVM models. The model using character n-grams as features also had better or equivalent F_1-scores to the SVM models using word n-grams or combined features. Therefore, it is clear that it is preferable to use character n-grams as features, as models using these features have a lower conversation impact while also attaining a larger F_1-score.



(a) RFM



(b) FRIDA

Figure 10.11: Conversation impact of three types of SVM models against the number of unmasked words, N_f, on the datasets RFM and FRIDA. In red impact on the baseline is included, which is independent of N_f.

We again see the pattern that the conversation impact increases when the number of unmasked words increases. It is important to note that a lower topic or conversation impact does not directly correspond to a better-performing model. For example, a model that attributes every text to author a_1 has a topic and conversation impact of 0, but also an extremely low F_1-score. As the F_1-scores increase when N_f increases it becomes clear that there exists no value for N_f that is optimal for both the F_1-score as the topic impact. Instead, a trade-off between a low conversation impact and a high F_1-score should be made before choosing which model to utilize in real forensic cases. In forensic cases where all texts cover the same topic, this choice might lean toward a model with better performance but a higher topic impact. On the other hand, in cases where the unknown text and the reference texts cover different topics, a more topic-robust model might be preferable, even if this comes at a cost in performance.

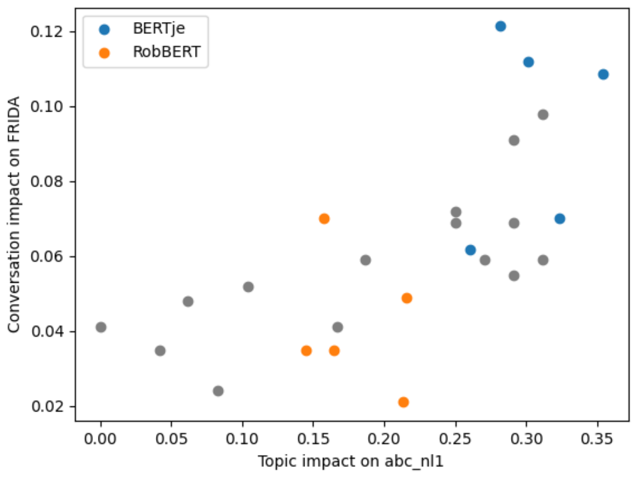


Figure 10.12: Plot of the conversation impact against the topic impact for 25 different authorship attribution. The results for the models based on BERTje and RobBERT are highlighted in blue and orange.

In Figure 10.12, the second additional result in the correlation between topic and conversation impact is shown. Here the results of the BERTje and RobBERT-based models are highlighted. We notice that BERTje-based models have a significantly higher topic and conversation impact compared to

RobBERT-based models. Per model type, the dots represent the impact at 10, 20, 30, 40 and 50 epochs of finetuning. No relation was found between the length of finetuning and the topic or conversation impact.

We are not sure what causes this large distinction between BERTje and RobBERT-based models. One cause could be the training material. BERTje is pre-trained on Wikipedia articles, news articles and books, which are in general written more in a more generic style with fewer style differences between the texts. On the other hand, RobBERT is pre-trained on an internet scrape, which is likely to contain a much larger variety of individual writing styles. Therefore, during pre-training BERTje sees texts which differ more in topic than in style, while RobBERT sees texts that differ greatly in topic and style. This could lead BERTje to rely more on the topic of texts to attribute them then RobBERT in the difficult confusion case.

11 Conclusions and Recommendations

11.1. Conclusions

We studied the performance of two state-of-the-art types of models and a baseline on the computational authorship attribution task, where a text must be attributed to a specific author. The studied model types were support vector machines with feature vectors, BERT-based methods and the baseline, a model using logistic regression with the 100 most frequent words as features. We conclude that on forensically relevant Dutch corpora support vector machines with character 2 to 5-grams as features currently result in the best performance. Specifically, we received an F_1-score of 0.880 on the RFM dataset and 0.957 on the FRIDA dataset, which both consist of 50 authors and 8 texts per author. Compared to the baseline values of 0.427 and 0.547 this gives a significant improvement.

We also constructed a score-based likelihood ratio system to compute the value of evidence of texts suspected to be written by a specific author, which is the forensic authorship attribution task. Cross-calibration was used to be able to compute kernel density estimators even when only 7 texts were available for training and calibration. This method can be used jointly with all studied computational authorship attribution methods. In general, the relative performance of models used as part of the likelihood ratio system was similar to that on the computational authorship attribution task. One big difference was the performance of BERT-based methods, which performed significantly worse than the baseline in a likelihood ratio system while outperforming the baseline in the F_1-score evaluation. As a result, BERT-based methods are currently not a good alternative for the baseline in likelihood ratio systems where cross-calibration is used.

Additionally, we introduced two metrics to measure the topic-robustness of authorship attribution methods on both topic-controlled and conversational datasets. We found a correlation of 0.68 between the proposed metrics 'topic impact' and 'conversation impact', but this correlation was not strong enough to decisively conclude whether the conversation impact can indeed be used as a proxy for the topic impact to measure topic-robustness on conversational datasets or not.

Masking, a technique where infrequent words are replaced by hashtags, was used to improve the topic-robustness of models. This decreased the performance of models, especially if only 200 or fewer of the most frequent words were left unmasked. However, using masking did decrease the topic impact and conversation impact on models, with this effect being the largest when only 100 words were left unmasked. The support vector machine with character n-grams as features with 100 unmasked words had a similar conversation impact to the baseline while outperforming it in performance on both the computational and forensic authorship attribution tasks. Therefore, we recommend using a support vector machine with character n-grams as features in practice. This model can be used in combination with features found manually by linguists, which is also currently done at the Netherlands Forensic Institute in combination with the baseline.

To choose the specific number of most frequent words that should be left unmasked, a consideration has to be made between performance and topic-robustness. We leave this choice to forensic scientists to decide on a case-by-case basis, as this decision can be heavily influenced by aspects that differ between forensic cases. For example, whether the topics of the training texts and the unknown text are the same or not. Therefore, additional case-specific factors should be kept in mind to avoid bias based on topic and the possible false convictions or acquittals that could result from it. With the metrics topic impact and conversation impact, we developed an additional tool to assist in these considerations.

11.2. Future research

Many avenues of research into forensic authorship attribution remain. In this section, we want to highlight possible research questions or methods for future research that occurred to us during our study but were out of scope for this project.

We noticed that to have a topic-robust authorship attribution model, it is necessary to mask uncommon words from the texts, as they contain relatively much topic-related information. However, as we currently mask all words outside of frequent word lists, misspelt versions of the words on the frequency list still get masked, even though the word contains no topic information. As consistent, uncommon misspellings could indicate authorship, we lose information by masking these misspellings. To make an authorship attribution model with masking able to capture these features, it might be possible to first correct spelling errors in a text using a large language model, then mask the text using the most frequent words, before reverting the texts to their original spelling. This way, misspelt versions of frequent words are kept in the text. A possible drawback of this method is that a misspelling of a frequent word could be the same as a misspelling of a much less frequent word, leading to some less frequent words not being masked, possibly lowering the topic-robustness of the method. In forensic cases, such features might also be found by the manual search by language experts. These features can then be added to the feature set.

A drawback of using BERT-based methods for authorship attribution is that all pre-training tasks focus more on the text's topic and sentence structure than on the author's style. To possibly improve the performance of these methods on the authorship attribution task a style-related pre-training task could be performed. For example, a task where the BERT model is given two sentences and has to determine whether they have been written by the same author or not. This would prime the BERT model for tasks about authorship before being finetuned on the specific texts of a group of authors. However, pre-training a new BERT model specifically for authorship attribution would have a very high computational cost.

There is also room for further research into the metrics we introduced and their relationship. We found that the 'topic impact' and 'conversation impact' were correlated for our authorship attribution methods, but that this correlation is not strong enough to be able to draw strong conclusions about the topic-robustness of models based on the conversation impact. Further research, which could for example include more authorship attribution methods or test the hypothesis on different corpora, could support or disprove our theory about the use of conversations to measure the topic-robustness of authorship attribution methods.

References

[1] Malik Altakrori, Jackie Chi Kit Cheung, and Benjamin CM Fung. “The topic confusion task: A novel evaluation scenario for authorship attribution”. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. 2021, pp. 4242–4256.

[2] Miriam Ayer et al. “An empirical distribution function for sampling with incomplete information”. In: *The annals of mathematical statistics* (1955). Publisher: JSTOR, pp. 641–647.

[3] Harald Baayen et al. “An experiment in authorship attribution”. In: *6th JADT*. Vol. 1. Citeseer, 2002, pp. 69–75.

[4] Douglas Bagnall. *Author Identification using Multi-headed Recurrent Neural Networks*. Aug. 16, 2016. arXiv: `1506.04891[cs]`.

[5] Georgios Barlas and Efstathios Stamatatos. “Cross-Domain Authorship Attribution Using Pre-trained Language Models”. In: *Artificial Intelligence Applications and Innovations*. Vol. 583. Series Title: IFIP Advances in Information and Communication Technology. Cham: Springer International Publishing, 2020, pp. 255–266.

[6] Georgios Barlas and Efstathios Stamatatos. “A transfer learning approach to cross-domain authorship attribution”. In: *Evolving Systems* 12.3 (Sept. 2021), pp. 625–643.

[7] R. E. Barlow and H. D. Brunk. “The Isotonic Regression Problem and its Dual”. In: *Journal of the American Statistical Association* 67.337 (Mar. 1972), pp. 140–147.

[8] Dasha Bogdanova and Angeliki Lazaridou. “Cross-Language Authorship Attribution.” In: *LREC*. Citeseer, 2014, pp. 2015–2020.

[9] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. “A training algorithm for optimal margin classifiers”. In: *Proceedings of the fifth annual workshop on Computational learning theory*. COLT92: 5th Annual Workshop on Computational Learning Theory. Pittsburgh Pennsylvania USA: ACM, July 1992, pp. 144–152.

[10] Niko Brummer and Johan du Preez. *The PAV algorithm optimizes binary proper scoring rules*. Apr. 8, 2013. arXiv: `1304.2331[cs,stat]`.

[11] Niko Brümmer and Johan Du Preez. “Application-independent evaluation of speaker detection”. In: *Computer Speech & Language* 20.2 (2006). Publisher: Elsevier, pp. 230–275.

[12] Chih-Chung Chang and Chih-Jen Lin. “LIBSVM: A library for support vector machines”. In: *ACM Transactions on Intelligent Systems and Technology* 2.3 (Apr. 2011), pp. 1–27.

[13] Pieter Delobelle, Thomas Winters, and Bettina Berendt. *RobBERT: a Dutch RoBERTa-based Language Model*. Sept. 16, 2020. arXiv: `2001.06286[cs]`.

[14] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. May 24, 2019. arXiv: `1810.04805[cs]`.

[15] Ian W. Evett. “Towards a uniform framework for reporting opinions in forensic science casework”. In: *Science & Justice* 3.38 (1998), pp. 198–202.

[16] Maël Fabien et al. “BertAA: BERT fine-tuning for Authorship Attribution”. In: *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*. 2020, pp. 127–137.

[17] Gerda Frankenhuis. *NFI blaast na doorbraak ‘vergeten’ vakgebied nieuw leven in*. Telegraaf. Apr. 26, 2024. URL: `https://www.telegraaf.nl/nieuws/34104946` (visited on 05/23/2024).

[18] Jade Goldstein, Ransom Winder, and Roberta Sabin. “Person identification from text and speech genre samples”. In: *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*. 2009, pp. 336–344.

[19] Helena Gómez-Adorno et al. “Document embeddings learned on various types of n-grams for cross-topic authorship attribution”. In: *Computing* 100.7 (July 2018), pp. 741–756.

[20] Tim Grant. *The Idea of Progress in Forensic Authorship Analysis*. 1st ed. Cambridge University Press, May 19, 2022.

[21] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY: Springer New York, 2001.

[22] Shunichi Ishihara. “A likelihood ratio-based evaluation of strength of authorship attribution evidence in SMS messages using N-grams.” In: *International Journal of Speech, Language & the Law* 21.1 (2014).

[23] Shunichi Ishihara. “Strength of linguistic text evidence: A fused forensic text comparison system”. In: *Forensic Science International* 278 (Sept. 2017), pp. 184–197.

[24] Patrick Juola. “Authorship attribution”. In: *Foundations and Trends® in Information Retrieval* 1.3 (2008). Publisher: Now Publishers, Inc., pp. 233–334.

[25] Patrick Juola. “Verifying authorship for forensic purposes: A computational protocol and its validation”. In: *Forensic Science International* 325 (Aug. 2021), p. 110824.

[26] Patrick Juola and R. Harald Baayen. “A controlled-corpus experiment in authorship identification by cross-entropy”. In: *Literary and Linguistic Computing* 20 (Suppl 2005). Publisher: EADH: The European Association for Digital Humanities, pp. 59–67.

[27] Mike Kestemont et al. “Overview of the cross-domain authorship attribution task at {PAN} 2019”. In: *Working Notes of CLEF 2019-Conference and Labs of the Evaluation Forum, Lugano, Switzerland, September 9-12, 2019*. 2019, pp. 1–15.

[28] Emmanuel Keuleers, Marc Brysbaert, and Boris New. “SUBTLEX-NL: A new measure for Dutch word frequency based on film subtitles”. In: *Behavior Research Methods* 42.3 (Aug. 2010), pp. 643–650.

[29] Moshe Koppel, Jonathan Schler, and Shlomo Argamon. “Computational methods in authorship attribution”. In: *Journal of the American Society for Information Science and Technology* 60.1 (Jan. 2009), pp. 9–26.

[30] Moshe Koppel, Jonathan Schler, and Shlomo Argamon. “Authorship attribution in the wild”. In: *Language Resources and Evaluation* 45.1 (Mar. 2011), pp. 83–94.

[31] Maarten Lambers and Cor J. Veenman. “Forensic Authorship Attribution Using Compression Distances to Prototypes”. In: *Computational Forensics*. Vol. 5718. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 13–24.

[32] Anna Jeannette Leegwater et al. “From data to a validated score-based LR system: a practitioner’s guide”. In: *Forensic Science International* (2024). Publisher: Elsevier, p. 111994.

[33] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. July 26, 2019. arXiv: `1907.11692[cs]`.

[34] Kim Luyckx and Walter Daelemans. “The effect of author set size and data size in authorship attribution”. In: *Literary and linguistic Computing* 26.1 (2011). Publisher: Oxford University Press, pp. 35–55.

[35] Didier Meuwly, Daniel Ramos, and Rudolf Haraksim. “A guideline for the validation of likelihood ratio methods used for forensic evidence evaluation”. In: *Forensic science international* 276 (2017). Publisher: Elsevier, pp. 142–153.

[36] Frederick Mosteller and David L. Wallace. “Inference in an Authorship Problem: A Comparative Study of Discrimination Methods Applied to the Authorship of the Disputed *Federalist* Papers”. In: *Journal of the American Statistical Association* 58.302 (June 1963), pp. 275–309.

[37] Benjamin Murauer and Günther Specht. “Developing a benchmark for reducing data bias in authorship attribution”. In: *Proceedings of the 2nd Workshop on Evaluation and Comparison of NLP Systems*. 2021, pp. 179–188.

[38] Benjamin Murauer and Günther Specht. *DT-grams: Structured Dependency Grammar Stylometry for Cross-Language Authorship Attribution*. June 10, 2021. arXiv: `2106.05677[cs]`.

[39] Lukas Muttenthaler, Gordon Lucas, and Janek Amann. “Authorship Attribution in Fan-Fictional Texts given variable length Character and Word N-Grams”. In: (2019).

[40] Danica M. Ommen and Christopher P. Saunders. “Building a unified statistical framework for the forensic identification of source problems”. In: *Law, Probability and Risk* 17.2 (2018). Publisher: Oxford University Press, pp. 179–197.

[41] John Platt. *Sequential minimal optimization: A fast algorithm for training support vector machines*. 1998.

[42] Alec Radford et al. *Improving language understanding by generative pre-training*. Publisher: OpenAI. 2018.

[43] Daniel Ramos et al. “Deconstructing cross-entropy for probabilistic binary classifiers”. In: *Entropy* 20.3 (2018). Publisher: MDPI, p. 208.

[44] Anderson Rocha et al. “Authorship Attribution for Social Media Forensics”. In: *IEEE Transactions on Information Forensics and Security* 12.1 (Jan. 2017), pp. 5–33.

[45] Victor Sanh et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. Feb. 29, 2020. arXiv: `1910.01108[cs]`.

[46] Upendra Sapkota et al. “Not all character n-grams are created equal: A study in authorship attribution”. In: *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies*. 2015, pp. 93–102.

[47] Nelleke Scheijen. “Forensic speaker recognition”. Master’s thesis. Delft University of Technology.

[48] Eleni-Konstantina Sergidou et al. “Frequent-words analysis for forensic speaker comparison”. In: *Speech Communication* 150 (May 2023), pp. 1–8.

[49] Bernard W. Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[50] Efstathios Stamatatos. “A survey of modern authorship attribution methods”. In: *Journal of the American Society for Information Science and Technology* 60.3 (Mar. 2009), pp. 538–556.

[51] Efstathios Stamatatos. “Authorship attribution using text distortion”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. 2017, pp. 1138–1149.

[52] Efstathios Stamatatos. “Masking topic-related information to enhance authorship attribution”. In: *Journal of the Association for Information Science and Technology* 69.3 (Mar. 2018), pp. 461–473.

[53] William J. Teahan and David J. Harper. “Using Compression-Based Language Models for Text Categorization”. In: *Language Modeling for Information Retrieval*. Dordrecht: Springer Netherlands, 2003, pp. 141–165.

[54] Jacob Tyo, Bhuwan Dhingra, and Zachary C. Lipton. *On the State of the Art in Authorship Attribution and Authorship Verification*. Oct. 5, 2022. arXiv: `2209.06869[cs]`.

[55] David Van der Vloed et al. “NFI-FRIDA–Forensically realistic interdevice audio database and intial experiments”. In: *27th Annual Conference of the International Association for Forensic Phonetics and Acoustics (IAFPA)*. 2018, pp. 25–27.

[56] Hans Van Halteren et al. “New Machine Learning Methods Demonstrate the Existence of a Human Stylome”. In: *Journal of Quantitative Linguistics* 12.1 (Apr. 2005), pp. 65–77.

[57] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[58] Peter Vergeer et al. “Numerical likelihood ratios outputted by LR systems are often based on extrapolation: When to stop extrapolating?” In: *Science & Justice* 56.6 (2016). Publisher: Elsevier, pp. 482–491.

[59] Wietse de Vries et al. *BERTje: A Dutch BERT Model*. Dec. 19, 2019. arXiv: `1912.09582[cs]`.

[60] Yonghui Wu et al. *Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. Oct. 8, 2016. arXiv: `1609.08144[cs]`.

A
Proof that the weighted average of the C_{llr} of two sets is the same as the C_{llr} of their union

Suppose we have the sets V_1 and V_2, consisting of likelihood ratios under H_p, $(V_{1,p}, V_{2,p})$, and likelihood ratios under H_d, $(V_{1,d}, V_{2,d})$ and the set $V = V_1 \cup V_2$, with disjoint subsets, V_p and V_d. We want to proof that the C_{llr} of V is equal to weighted average of the C_{llr} of V_1 and V_2, under the condition that the ratio of samples under H_p and under H_d is equal for each set, so

$$\frac{|V_{1,p}|}{|V_{1,d}|} = \frac{|V_{2,p}|}{|V_{2,d}|} = \frac{|V_p|}{|V_d|} \tag{A.1}$$

We first rewrite this to find the following equality's, which we set equal to new constants a and b

$$\frac{|V_{1,p}|}{|V_p|} = \frac{|V_{1,d}|}{|V_d|} = a, \quad \frac{|V_{2,p}|}{|V_p|} = \frac{|V_{2,d}|}{|V_d|} = b \tag{A.2}$$

Note that $a + b = 1$. Recall the definition of the log-likelihood-ratio cost C_{llr}

$$C_{llr}(V) = \frac{1}{2|V_p|} \sum_{LR \in V_p} \log_2 \left(1 + \frac{1}{LR}\right) + \frac{1}{2|V_d|} \sum_{LR \in V_d} \log_2 (1 + LR). \tag{A.3}$$

We can rewrite this as follows:

$$\begin{aligned}
C_{llr}(V) &= \frac{1}{2|V_p|} \sum_{LR \in V_p} \log_2 \left(1 + \frac{1}{LR}\right) + \frac{1}{2|V_d|} \sum_{LR \in V_d} \log_2 (1 + LR) \\
&= \frac{1}{2|V_p|} \left(\sum_{LR \in V_{1,p}} \log_2 \left(1 + \frac{1}{LR}\right) + \sum_{LR \in V_{2,p}} \log_2 \left(1 + \frac{1}{LR}\right) \right) \\
&+ \frac{1}{2|V_d|} \left(\sum_{LR \in V_{1,d}} \log_2 (1 + LR) + \sum_{LR \in V_{2,d}} \log_2 (1 + LR) \right) \\
&= \frac{|V_{1,p}|}{|V_p|} \frac{1}{2|V_{1,p}|} \sum_{LR \in V_{1,p}} \log_2 \left(1 + \frac{1}{LR}\right) + \frac{|V_{1,d}|}{|V_d|} \frac{1}{2|V_{1,d}|} \sum_{LR \in V_{1,d}} \log_2 (1 + LR) \\
&+ \frac{|V_{2,p}|}{|V_p|} \frac{1}{2|V_{2,p}|} \sum_{LR \in V_{2,p}} \log_2 \left(1 + \frac{1}{LR}\right) + \frac{|V_{2,d}|}{|V_d|} \frac{1}{2|V_{2,d}|} \sum_{LR \in V_{2,d}} \log_2 (1 + LR)
\end{aligned}$$

$$
\begin{aligned}
&= a\left(\frac{1}{2|V_{1,p}|}\sum_{LR\in V_{1,p}}\log_2\left(1+\frac{1}{LR}\right)+\frac{1}{2|V_{1,d}|}\sum_{LR\in V_{1,d}}\log_2\left(1+LR\right)\right)\\
&+ b\left(\frac{1}{2|V_{2,p}|}\sum_{LR\in V_{2,p}}\log_2\left(1+\frac{1}{LR}\right)+\frac{1}{2|V_{2,d}|}\sum_{LR\in V_{2,d}}\log_2\left(1+LR\right)\right)\\
&= aC_{llr}(V_1)+bC_{llr}(V_2)
\end{aligned}
$$

As $a + b = 1$, this is a weighted average. So we have proven that the C_{llr} of a set V is equal to the weighted average of the C_{llr} of its subsets V_1 and V_2, under the condition that the ratio of samples under H_p and H_d is equal for V and its subsets.

B
Tables of results presented in figures

This appendix contains tables corresponding to all results presented as figures in Chapter 10.

Table B.1: F_1-scores of 15 SVM models with varying feature types and number of unmasked words. These values correspond to Figure 10.1. The highest value in each column is included in boldface.

Corpus	RFM			FRIDA		
Feature type	Word	Character	Combined	Word	Character	Combined
N_f						
∞	**0.770**	**0.880**	**0.850**	**0.891**	**0.957**	**0.952**
5000	0.742	0.855	0.834	0.885	0.945	0.938
1000	0.699	0.850	0.826	0.885	0.940	0.938
200	0.480	0.710	0.682	0.805	0.874	0.902
100	0.368	0.603	0.589	0.692	0.775	0.770
Baseline	0.427	-	-	0.547	-	-

Table B.2: F_1-scores of several BERT-based models with varying epochs of training on the datasets RFM and FRIDA. These values correspond to Figure 10.2. The highest value in each column is highlighted in boldface.

(a) RFM

Epochs		10	20	30	40	50
BERTje	Truncated	0.260	0.470	0.576	0.633	0.658
	Averaging	**0.280**	0.478	0.583	0.640	0.653
	Mean pooling	0.274	0.519	0.594	0.663	0.661
RobBERT	Truncated	0.233	**0.558**	0.661	0.718	0.690
	Averaging	0.267	0.540	**0.680**	**0.734**	**0.727**
	Mean pooling	**0.280**	0.550	0.674	0.718	**0.727**

(b) FRIDA

Epochs		10	20	30	40	50
BERTje	Truncated	0.201	0.333	0.424	0.455	0.475
	Averaging	0.186	0.350	0.429	0.495	0.520
	Mean pooling	**0.226**	**0.464**	**0.554**	**0.577**	**0.629**
RobBERT	Truncated	0.116	0.201	0.284	0.367	0.434
	Averaging	0.083	0.197	0.334	0.441	0.488
	Mean pooling	0.104	0.312	0.483	0.563	0.620

Table B.3: C_{llr}, C_{llr}^{min} and C_{llr}^{cal} of SVM models with character n-grams as features and varying N_f on two datasets, FRIDA and RFM. These values correspond to Figure 10.3. The lowest value in each column is highlighted in boldface.

Corpus	RFM			FRIDA		
Metric	C_{llr}	C_{llr}^{min}	C_{llr}^{cal}	C_{llr}	C_{llr}^{min}	C_{llr}^{cal}
N_f						
∞	**0.205**	**0.071**	0.134	**0.158**	**0.038**	0.120
5000	0.218	0.090	**0.128**	0.166	0.052	**0.114**
1000	0.237	0.104	0.133	0.168	0.042	0.126
200	0.311	0.172	0.139	0.221	0.084	0.138
100	0.392	0.246	0.145	0.284	0.145	0.139
Baseline	0.526	0.372	0.154	0.469	0.308	0.161

Table B.4: Ratio of attributions to the other or partner group for abc_nl1 and FRIDA corpus, respectively. The topic or conversation impact can be computed from these numbers by subtracting the value for the confusion case from that for the standard case. These values correspond to Figure 10.9. The lowest value in each column is highlighted in boldface.

	Corpus	abc_nl1		FRIDA	
Model type	Model specification	Standard case	Confusion case	Standard case	Confusion case
SVM models	N_f	$\frac{s_o}{s_c+s_s+s_o}$	$\frac{s_o}{s_c+s_s+s_o}$	$\frac{s_p}{s_c+s_p+s_r}$	$\frac{s_p}{s_c+s_p+s_r}$
Character	∞	0.271	0.583	**0.024**	0.115
	5000	0.292	0.542	0.031	0.090
	1000	0.25	0.562	0.042	0.097
	200	0.229	0.521	0.049	0.090
	100	0.396	0.417	0.042	0.090
Word	∞	0.188	0.688	0.056	0.128
	5000	**0.146**	0.625	0.056	0.125
	1000	0.25	0.521	0.049	0.118
	200	0.396	0.562	0.056	0.097
	100	0.292	0.542	0.076	0.111
Combined	∞	0.208	0.646	**0.024**	0.122
	5000	0.229	0.625	0.038	0.097
	1000	0.188	0.458	0.035	0.094
	200	0.312	0.583	0.038	0.090
	100	0.312	0.417	0.059	**0.083**
BERT-based models	# of epochs				
BERTje	10	0.407	0.667	0.163	0.225
	20	0.312	0.634	0.128	0.198
	30	0.281	0.634	0.093	0.202
	40	0.282	0.583	0.089	0.202
	50	0.270	0.552	0.074	0.195
RobBERT	10	0.368	0.533	0.170	0.205
	20	0.257	0.473	0.142	0.191
	30	0.212	0.423	0.142	0.163
	40	0.226	0.384	0.121	0.191
	50	0.191	**0.336**	0.128	0.163

Table B.5: Conversation impact of 15 SVM models with varying feature types and number of unmasked words. These values correspond to Figure 10.11. The lowest value in each column when excluding the baseline is included in boldface.

Corpus	RFM			FRIDA		
Feature type	Word	Character	Combined	Word	Character	Combined
N_f						
∞	0.153	0.078	0.090	0.131	0.093	0.100
5000	0.127	0.067	0.079	0.142	0.065	0.100
1000	0.112	0.038	0.070	0.080	0.045	0.065
200	0.066	0.033	**0.045**	**0.025**	**0.020**	0.048
100	**0.055**	**0.025**	0.048	0.031	0.030	**0.036**
Baseline	0.032	-	-	0.021	-	-

C
Tables of additional experiments results

This chapter contains the results of some small experiments ran during this thesis to make decisions about which strategies to focus on in the actual results (Chapter 10). The choices based on these results have been motivated in Chapter 4.

Table C.1: F_1-scores of SVM model with character n-grams as features and varying N_f for two multiclass classification strategies. Highest value in each row is highlighted in boldface. Created on the FRIDA corpus.

N_f	One-vs-Rest	One-vs-One
∞	**0.957**	0.659
5000	**0.945**	0.648
1000	**0.940**	0.567
200	**0.874**	0.459
100	**0.775**	0.396

Table C.2: F_1-scores of SVM model with character n-grams as features and varying N_f for two different SVM kernels. Highest value in each row is highlighted in boldface. Created on the FRIDA corpus.

N_f	Linear kernel	Gaussian kernel
∞	**0.957**	0.795
5000	**0.945**	0.795
1000	**0.940**	0.885
200	**0.874**	0.859
100	**0.775**	0.744

Table C.3: F_1-scores of SVM model with character n-grams as features and varying N_f for two different masking strategies. Highest value in each row is highlighted in boldface. Created on the FRIDA corpus.

N_f	Single masking	Multiple masking
5000	0.945	**0.954**
1000	**0.940**	0.934
200	0.874	**0.890**
100	**0.775**	0.759

Table C.4: F_1-scores of a RobBERT-based model with mean pooling strategy for various epochs of training, either using no masking or masking with $N_f = 1000$. Highest value in each column is highlighted in boldface. Created on the FRIDA corpus.

Epochs	10	20	30	40	50
No masking	**0.104**	**0.312**	**0.483**	**0.563**	**0.620**
Masking, $N_f = 1000$	0.047	0.232	0.416	0.505	0.563