University of Groningen

Master's thesis

Recognising Darknet Market Vendors using Author Verification Methods

Artificial Intelligence

Author:
L.N. Faber, s2500523
Internal supervisor: Dr. M.A. Wiering (Bernoulli Institute)
External supervisor: Drs. M. van der Ree (Web-IQ)

June 28, 2021

Bernouilli Institute for Mathematics, Computer Science and Artificial Intelligence
University of Groningen, the Netherlands

Web-IQ, the Netherlands

Contents

1 Introduction **5**

2 Text Representation **7**
2.1 Count Vectorization . . . 7
2.2 Word Embeddings . . . 7
2.2.1 Latent Semantic Analysis . . . 8
2.2.2 Word2Vec . . . 8
2.2.3 GloVe . . . 11
2.2.4 FastText . . . 13

3 Classification **14**
3.1 Support Vector Machine . . . 14
3.2 Decision Tree . . . 16
3.3 Random Forest . . . 16
3.4 Neural Networks . . . 16
3.4.1 Recurrent Neural Networks . . . 16
3.4.2 Siamese Neural Networks . . . 17

4 Experimental Setup **18**
4.1 Data Acquisition and Analysis . . . 18
4.1.1 PAN 2020 . . . 18
4.1.2 Darknet . . . 18
4.2 Data Encoding . . . 20
4.2.1 Stylometric Features . . . 20
4.2.2 Embeddings . . . 21
4.2.3 Meta-features . . . 22
4.2.4 Pre-processing . . . 22
4.3 Models . . . 22
4.3.1 Handcrafted Features . . . 23
4.3.2 Siamese Neural Network . . . 24

5 Results **26**
5.1 Handcrafted Features . . . 26
5.1.1 Support Vector Machines . . . 26
5.1.2 Decision Trees . . . 27
5.1.3 Random Forests . . . 27
5.2 Siamese Neural Networks . . . 28

5.2.1 GloVe Embeddings 28
5.2.2 FastText Embeddings 28
5.3 Model Comparison 29
5.4 Influence of Meta-features 31

6 Discussion and Conclusion **32**
6.1 Discussion 32
6.1.1 Research Questions 32
6.1.2 Future Work 34
6.2 Conclusion 34

References **35**

Appendices **40**

Abstract

The web can be used for anything, and people with malicious intent have found ways to exploit its countless possibilities. The dark web has provided criminals with new ways of selling their illegal goods. Law enforcement agencies are continuously trying to lift some of the anonymity provided by onion routing in order to run investigations and prosecute these individuals. One approach to this challenge is to find links between multiple profiles managed by a single person in the hope of discovering revealing information on one of them. We will support this endeavor by applying machine learning techniques to link profiles by means of author verification.

Several classification algorithms trained on handcrafted stylometric features are compared with Siamese neural networks relying on two types of word embeddings. To assess the capabilities of these models, they are evaluated using data scraped from the dark web, created by various numbers of authors, as well as texts provided for a well-known public shared task on author verification. Additionally, the benefit of adding meta-features to the input of some models is experimented with.

A support vector machine trained on handcrafted features performed best when dealing with texts written by a small set of authors. The Siamese neural network-based approach delivered the highest scores in many-author scenarios and comes with an additional benefit when applied in production. Furthermore, meta-features have shown to be significantly improving results.

Our research shows that applying machine learning techniques for author verification on Darknet market advertisements is feasible and that there are many development possibilities. Suggestions for future research include: experimenting with more types of text representations, the addition and transformation of meta-features, and the combination of handcrafted features with word embeddings. A system implementing such improvements could be successfully applied to link profiles of dark web users and help law enforcement agencies in tracking down vendors of illegal goods.

Acknowledgements

I want to thank my primary supervisor, Dr. Marco Wiering, who helped me find an interesting research topic and provided me with directions and advice whenever I needed them. Furthermore, my gratitude goes out to my external supervisor Michiel van der Ree. He was constantly available despite the work-from-home situation and taught me many practical lessons that are and will be of great use during my career. I would like to express my appreciation for the support and patience of my colleagues, friends, and family. Finally, I want to thank my girlfriend, Anne, for always being there and supporting and motivating me throughout my research.

List of Figures

2.1 The two Word2Vec architectures as introduced in [1], taken from [2]. 9

3.1 The Support Vector Machine (taken from [3]) 15

4.1 Histograms of word counts on the datasets. 19
4.2 The distribution of dark marketplace advertisements. 20
4.3 The structure of the Siamese neural network employed for author verification. 24

5.1 ROC curves for each dataset. The curves are produced by the model types that performed the best on each particular dataset. The types of these models are specified in table 5.6. In each figure, the blue line represents the mean ROC curve, while the transparent areas resemble the difference between the upper and lower ROC curves of the models tested during 5-fold cross-validation. . . 30

1 The distribution of meta-feature values in the datasets. 41
2 A visualisation of hyper-parameter optimization experiments of SVMs performed on the PAN 2020 dataset. The left figure shows SVMs with a linear kernel. The right figure shows SVMs with an RBF kernel. In red are the AUC scores of the best performing models. 42
3 A visualisation of hyper-parameter optimization experiments of SVMs performed on the DN10 dataset. The left figure shows SVMs with a linear kernel. The right figure shows SVMs with an RBF kernel. In red are the AUC scores of the best performing models. 42
4 A visualisation of hyper-parameter optimization experiments of SVMs performed on the DN100 dataset. The left figure shows SVMs with a linear kernel. The right figure shows SVMs with an RBF kernel. In red are the AUC scores of best performing models. 43
5 A visualisation of hyper-parameter optimization experiments of SVMs performed on the DN1000 dataset. The left figure shows SVMs with a linear kernel. The right figure shows SVMs with an RBF kernel. In red are the AUC scores of the best performing models. 43
6 A visualisation of hyper-parameter optimization experiments of decision trees performed on the PAN 2020 dataset. In red is the AUC score of the best performing model. 44
7 A visualisation of hyper-parameter optimization experiments of decision trees performed on the DN10 dataset. In red is the AUC score of the best performing model. 44

8 A visualisation of hyper-parameter optimization experiments of decision trees performed on the DN100 dataset. In red is the AUC score of the best performing model. 45
9 A visualisation of hyper-parameter optimization experiments of decision trees performed on the DN1000 dataset. In red is the AUC score of the best performing model. 45

List of Tables

4.1 Definitions of handcrafted features. 21
4.2 Parameter search space for the SVM classifier. 23
4.3 Parameter search space for the decision tree classifier. 23
4.4 Parameter search space for the random forest classifier. 24
4.5 Parameter search space for the Siamese neural network. 25

5.1 Parameters and results of the best performing SVM models on the four datasets. 27
5.2 Parameters and results of the best performing decision tree models on the four datasets. 27
5.3 Parameters and results of the best performing random forest models on the four datasets. 28
5.4 Parameters and results of the best performing Siamese neural network models with GloVe embeddings on the four datasets. 28
5.5 Parameters and results of the best performing Siamese neural network models with FastText embeddings on the four datasets. 29
5.6 Scores and types of the best performing models overall on the four datasets. . 29
5.7 AUC scores of the best performing Siamese neural network models after embeddings of meta-features have been added as initial states for the GRUs that are the base of these models. In **bold** are the best performing models. 31

1 Complete list of hyper-parameter optimization experiments of random forest classifiers performed on the PAN 2020 dataset. In **bold** is the AUC score of the best performing model. 46
2 Complete list of hyper-parameter optimization experiments of random forest classifiers performed on the DN10 dataset. In **bold** is the AUC score of the best performing model. 47
3 Complete list of hyper-parameter optimization experiments of random forest classifiers performed on the DN100 dataset. In **bold** is the AUC score of the best performing model. 48
4 Complete list of hyper-parameter optimization experiments of random forest classifiers performed on the DN1000 dataset. In **bold** is the AUC score of the best performing model. 49
5 Complete list of hyper-parameter optimization experiments of Siamese GRU models with GloVe and FastText embeddings performed on the PAN 2020 dataset. In **bold** are the AUC scores of best performing models. 50

6 Complete list of hyper-parameter optimization experiments of Siamese GRU models with GloVe and FastText embeddings performed on the DN10 dataset. In **bold** are the AUC scores of best performing models. 51
7 Complete list of hyper-parameter optimization experiments of Siamese GRU models with GloVe and FastText embeddings performed on the DN100 dataset. In **bold** are the AUC scores of best performing models. 52
8 Complete list of hyper-parameter optimization experiments of Siamese GRU models with GloVe and FastText embeddings performed on the DN1000 dataset. In **bold** are the AUC scores of best performing models. 53

Chapter 1

Introduction

Although the internet has proven its great value for humanity, it has also become apparent that the anonymity it can provide to its users has created new opportunities for people with malicious intent. Ever since the first devices were connected to the web, people and their digital, as well as real-world properties, have become victims of many forms of cybercrime. A large portion of these crimes can be brought in connection with the dark web, a part of the internet that is not indexed by search engines such as Google and on which users are provided a higher level of anonymity by the use of onion routing [4]. Law enforcement agencies aim to lift some of this anonymity to allow for investigations and prosecution of criminals. Linking multiple dark web profiles and combining the information gathered from them creates one opportunity to demystify the identity of dark web users. This thesis aims to support this effort by determining what methods are most suitable to link dark web accounts using the advertisements posted by them.

While the dark web can be used for lawful causes, such as journalism and opinion sharing without censorship, it has become infamous as a tool for criminals that facilitates the sale of drugs, weapons, and illegal services online. Many so-called Darknet marketplaces have risen and fell in recent years, with the best-known example being the Silk Road. This marketplace was shut down in 2013 and resurfaced as Silk Road 2.0 a month later. After this marketplace was also shut down in 2014 as part of operation Onymous[1], other marketplaces filled the gap that their predecessors left behind in continuous succession [5]. The reasons for the disappearance and reappearance of dark marketplaces vary. While some are shut down by law enforcement agencies, others are taken offline by their administrators as part of an exit scam or because their owners think they can no longer guarantee the anonymity of themselves or their clients.

Illegal vendors often continue their business by migrating to new marketplaces as their predecessors are shut down. While doing so, a vendor might decide to use a different account name. A reason for this might be to divide their business and thus spread risks. Another reason could be to impersonate well-known accounts to gain the trust of potential customers. The result of this behavior is that multiple accounts can be mapped to single distinct sellers [6].

This poses a challenge for law enforcement agencies: to be able to create a complete record of suspects it is necessary to identify these persons through all of their accounts, even if different account names are used. The anonymity-centered nature of the dark web makes

[1] `https://www.europol.europa.eu/newsroom/news/global-action-against-dark-markets-tor-network`

this an arduous task, but its necessity has been shown in the past: the founder of the Silk Road, Ross Ulbricht, better known under his alias Dread Pirate Roberts, was caught because authorities were able to link multiple of his accounts, and information found on one of them ultimately led researchers to his real identity [7].

Various techniques can be used to connect online profiles and complete records of suspects of online crimes. An obvious approach would be to link profiles with identical or highly similar account names. Another approach could be to use the PGP keys shared by vendors [8], or a combination of both [9]. These methods can serve as decent heuristics, but are susceptible to impersonation attacks, as nothing would stop users from copying names and PGP keys already used by others [6]. An attempt has been made to maintain a database that could also be used to track vendors through multiple marketplaces, but it appears such an undertaking comes at too high human operating cost [10].

In this thesis, we will contribute to the endeavor of linking profiles by researching the feasibility of applying machine learning algorithms to recognize authors by means of their writing styles. Known as author verification, the task at hand is to confirm whether a suspect is or is not the author of a document in question [11]. Author verification has been a subject of research for many years, and various techniques have been introduced to verify the authors of documents such as emails [11; 12], social media posts [13], and even novels from the 2nd-century [14].

We will investigate which author verification technique is most suitable for application on Darknet market data. More specifically, this research aims to answer the following question:

> *How well can authors of Darknet market advertisements be recognized using the available text and metadata?*

This question can be decomposed into the following sub-questions:

- What combination of representation and classification methods can most accurately verify authors of short texts?
- Can metadata collected from Darknet market advertisements be of use in an author verification system?

The research will be conducted at Web-IQ. This company assists authorities in stopping criminal activities by providing smart solutions to extract intelligence from online data. Performing author verification could be an interesting addition to their solutions. Web-IQ has gathered much dark web data by means of scraping over the past years, and the resulting datasets at their disposal provide an interesting opportunity for this research.

We search for the best approach to author verification on Darknet market data by comparing three well-known machine learning algorithms with a neural network-based classifier. The former will be trained on handcrafted stylometric features and the latter will be fed with more recently developed text representations. A dataset not related to the dark web will be experimented with as well to allow for a better comparison of the various classification and representation methods.

Chapter 2

Text Representation

A major challenge in the field of Natural Language Processing (NLP) is the representation of text. Various methods have been developed to convert words and sentences into numerical forms that preserve some form of meaning. These methods should produce a vector that can be used in statistical or machine learning models. This chapter discusses two types of methods applied in authorship verification: count vectorization and word embedding.

2.1 Count Vectorization

A representation method well-known in the field of NLP named Bag of Words (BoW) is based on counting. The term *Bag* in this case refers to the fact that words are counted without considering their relative position in the text. Texts are represented by vectors of vocabulary length $|V|$ on which the occurrence of each word is stored using counts or binary values resembling their presence or absence. These vectors were originally described as Vector Space Models (VSM) [15], and combined with linear classifiers provided an early approach to text classification [16]. Extensions of these models such as term frequency-inverse document frequency (tf-idf), which compares the occurrence rate of a word in a document relative to that of the word in an entire corpus, have shown to be particularly useful in information retrieval [17; 18].

BoW-type vectors can become highly dimensional as the size of the corpus increases. Furthermore, information is lost due to the fact that the order of words and their semantics are ignored. Another disadvantage of the BoW approach is that vectors are highly dependent on text topic, reducing effectiveness for authorship analysis [19].

Several other types of features applicable for authorship analysis are based on count vectorization. Many lexical, syntactical, structural, and content-specific features have been addressed [20; 19; 21]. While much research has been done on the challenge of finding the optimal set of features [22; 23; 24; 25], performance is influenced by data domains, preprocessing steps, and classifiers used as well, making it difficult to select a general solution.

2.2 Word Embeddings

Texts can also be represented by dense representations. As opposed to count vectors, these so-called *embeddings* provide a more efficient way of representing text and allow for the capture of meaning in the form of semantic or syntactic relationships between sentences, words, or

characters. The semantic relationships are discovered through the distributional hypothesis, which states that words that occur in similar contexts have similar meanings [26]. This hypothesis became popular being summarized as "You shall know a word by the company it keeps" [27]. Two model families can be distinguished: global matrix factorization methods, and local context window methods [28]. In this section we first discuss two methods, one of each family, with a substantial history in research, followed by two improvements upon these models that have been experimented with during our research.

2.2.1 Latent Semantic Analysis

A popular early method named Latent Semantic Analysis (LSA) [29] combines the BoW principle with the distributional hypothesis. Being a global matrix factorization-type method, the idea behind this method is to create a term-document matrix in which the columns represent documents and the rows represent words. The term-frequencies in this matrix are typically transformed into tf-idf scores. After dimension reduction using a reduced-rank singular value decomposition (SVD), the result is a k-dimensional approximation of the original matrix, in which each document and term is represented by a k-dimensional vector in the space derived by the SVD [30]. The document-document, document-term, and term-term similarity can then be determined using cosine-similarity. LSA was originally designed for information retrieval, sometimes described as Latent Semantic Indexing (LSI). While it can leverage statistical information and capture some semantic relation between words, choosing the right value for k significantly influences performance, and other methods have outperformed LSA in recent years. These models are discussed in the remainder of this section.

2.2.2 Word2Vec

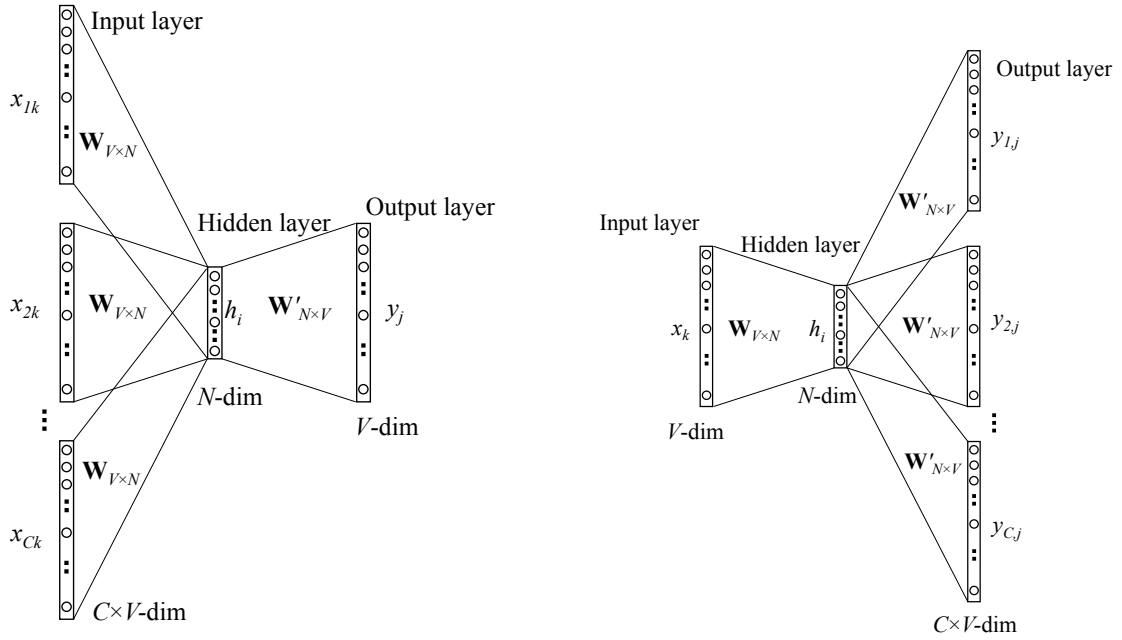
Neural Network Language Models (NNLM) have gained much attention after the first model based on a feed-forward neural network was introduced in [31] and the power of pre-trained word embeddings was signified in [32]. However, using neural networks for word representation learning has really become a common practice after Mikolov et al. introduced two novel local context window-based architectures wrapped in a framework called Word2Vec in [1; 33]. The central idea behind the two architectures named Continuous Bag of Words (CBOW) and Continuous Skip-gram (CSG) is to remove the non-linear hidden layer used in NNLM architectures to reduce computational complexity.

Continuous Bag of Words

The CBOW architecture aims to predict a target word w_t from C context words, w_I. This is achieved by employing a neural network with a single layer of size N, as illustrated in figure 2.1a. The network input consists of C one-hot encoded vectors $\mathbf{x} \in \mathbb{R}^V$, where V is the length of the vocabulary. These vectors are fed through weight matrix $\mathbf{W} \in \mathbb{R}^{V \times N}$, in which each row represents a word in the input layer with vector $\mathbf{v}_w \in \mathbb{R}^N$.

One-hot encoding implies that for every input word only one of V elements $\{x_1, ..., x_V\}$ will be 1. This means that in the case of a single context word ($C = 1$), the activation function of the hidden layer is

$$\mathbf{h} = \mathbf{v}_{w_I}^T \tag{2.1}$$



(a) Continuous Bag of Words.

(b) Continuous Skip-gram.

Figure 2.1: The two Word2Vec architectures as introduced in [1], taken from [2].

With larger context windows ($C > 1$), the vectors are averaged, and the activation function becomes

$$\mathbf{h} = \frac{1}{C} \sum_{c=1}^{C} \mathbf{v}_{w_{I,c}}^T \tag{2.2}$$

From the hidden layer to the output layer, a different weight matrix $\mathbf{W}' \in \mathbb{R}^{N\times V} = \{w'_{ij}\}$ is used to calculate score u_j for each word in the vocabulary. The vector $\mathbf{v}'_{w_j}$ consists of weights w'_{ij} from hidden unit i to output unit j. With $\mathbf{v}'_{w_j}$ as the j-th column in $\mathbf{W}'$, we define

$$u_j = {\mathbf{v}'_{w_j}}^T \mathbf{h} \tag{2.3}$$

Finally, the Softmax function is employed to produce the posterior distribution of words:

$$p(w_j|w_{I,1}, ..., w_{I,C}) = y_j = \frac{\exp(u_j)}{\sum_{j\prime=1}^{V} \exp(u_{j\prime})} \tag{2.4}$$

Given the input context words $\{w_{I,1}, ..., w_{I,C}\}$, the goal of the training procedure is to maximize the probability of the actual output word w_O. Applying log probabilities, this translates to

$$\begin{aligned} \max p(w_O|w_{I,1}, ..., w_{I,C}) &= \max \log y_{j^*} \\ &= u_{j^*} - \log \sum_{j'=1}^{V} \exp(u_{j\prime}), \end{aligned} \tag{2.5}$$

where j^* is the index of of the actual output word in the output layer. The loss function to minimize consequently becomes

$$E = -\log p(w_O|w_{I,1}, ..., w_{I,C}) \tag{2.6}$$

Applying back-propagation, we first take the derivative of E with regard to u_j and obtain

$$\frac{\partial E}{\partial u_j} = y_j - t_j := e_j, \tag{2.7}$$

where t_j is 1 if $j = j^*$, 0 otherwise, and e_j is the prediction error of the j-th word in the output layer. Next, the derivative on w'_{ij} is taken to obtain the gradient on the hidden $\rightarrow$ output weights:

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{ij}} = e_j \cdot h_i \tag{2.8}$$

Finally, stochastic gradient descent is utilized to form the updating equation for the hidden $\rightarrow$ output weights:

$$\mathbf{v}'^{(\text{new})}_{w_j} = \mathbf{v}'^{(\text{old})}_{w_j} - \eta \cdot e_j \cdot \mathbf{h} \qquad \text{for } j = 1, 2, ..., V, \tag{2.9}$$

where $\eta > 0$ is the learning rate. After updating the values in W', we proceed by finding the update equations for $\mathbf{W}$. First, the derivative of E on the output of the hidden layer is calculated using

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} := \text{EH}_i, \tag{2.10}$$

where h_i is the output of the i-th unit in the hidden layer and $\text{EH} \in \mathbb{R}^N$ is the sum of the output vectors of all words in the vocabulary, weighted by their prediction error e_j as defined in (2.7). Since h_i is defined by

$$h_i = \sum_{k=1}^{V} x_k \cdot w_{ki}, \tag{2.11}$$

and its derivative with respect to w_{ki} therefore is x_k, we subsequently acquire the derivative of E with regard to each element of $\mathbf{W}$:

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_{ki}} = \text{EH}_i \cdot x_k = x^T \text{EH} \tag{2.12}$$

Only C elements of $\mathbf{x}$ are non-zero, and it follows that just the vectors of their corresponding context words are updated in $\mathbf{W}$:

$$\mathbf{v}^{(\text{new})}_{w_{I,c}} = \mathbf{v}^{(\text{old})}_{w_{I,c}} - \frac{1}{C} \cdot \eta \cdot EH^T \qquad \text{for } c = 1, 2, ..., C, \tag{2.13}$$

where $\eta > 0$ is the learning rate, and $\mathbf{v}_{w_{I,c}}$ is called the *embedding* of context word $w_{I,c}$.

Continuous Skip-gram

As shown in figure 2.1b, the CSG model is the opposite of CBOW, with the target word at the input layer and the context words at the output layer. The objective is to predict context words given some target word.

With just one input word, $\mathbf{h}$ is defined as in (2.1). On the output layer, instead of the posterior distribution for one word, we now have C posterior distributions:

$$p(w_{c,j} = w_{O,c} | w_I) = \frac{\exp(u_{c,j})}{\sum_{j'=1}^{V} \exp(u_{j'})} = y_{c,j} \tag{2.14}$$

where w_I is the input word, $w_{O,c}$ is the actual c-th word in the output context words, and $y_{c,j}$ is the output for the j-th unit of the c-th context word. The loss function is therefore transformed to

$$\begin{aligned} E &= -\log p(w_{O,1}, ..., w_{O,C}|w_I) \\ &= -\log \prod_{c=1}^{C} \frac{\exp\left(u_{c,j_c^*}\right)}{\sum_{j'=1}^{V} \exp\left(u_{j'}\right)} \\ &= -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'=1}^{V} \exp\left(u_{j'}\right) \end{aligned} \tag{2.15}$$

The update equation for the hidden $\rightarrow$ output matrix W' is similar to (2.10), except that we now consider the error for every context word:

$$\mathbf{v}'^{(\text{new})}_{w_j} = \mathbf{v}'^{(\text{old})}_{w_j} - \eta \cdot \sum_{c=1}^{C} E_{c,j} \cdot \mathbf{h} \qquad \text{for } \; j = 1, 2, ..., V. \tag{2.16}$$

Conversely, the update equation for the input $\rightarrow$ hidden weights in (2.13) is tailored to suit a single input word and becomes

$$\mathbf{v}^{(\text{new})}_{w_I} = \mathbf{v}^{(\text{old})}_{w_I} - \eta \cdot EH^T. \tag{2.17}$$

An advantage of CSG over CBOW can be that embeddings of context words are not averaged during prediction, which prevents those of rare words from being smoothed away by more frequent examples throughout training. This allows for better representations of these words. However, since CSG involves multiple word classifications for every input word, complexity is increased, resulting in longer training times.

Negative Sampling

Mikolov et al. in [33] introduced negative sampling to reduce training times for CBOW and CSG models. The idea is to only update a sample of the output vectors in each iteration to save computations. The actual output word w_O always gets updated, as well as some random other words sampled through an arbitrarily chosen probabilistic distribution. Essentially being a simplified form of Noise Contrastive Estimation [34], the new training objective is defined as

$$E = -\log \sigma({\mathbf{v}'_{w_O}}^T \mathbf{h}) - \sum_{w_j \in W_{\text{neg}}} \log \sigma(-{\mathbf{v}'_{w_j}}^T \mathbf{h}), \tag{2.18}$$

where σ is the softmax function and $\mathbf{h}$ is the output of the hidden layer for either CBOW or CSG.

2.2.3 GloVe

Pennington et al. in [28] argue that since local context window methods such as CBOW and CSG are only trained on separate local context windows, they do not effectively utilize global corpus statistics. However, methods such as LSA (discussed in 2.2.1) aimed at doing so, show indications of a sub-optimal vector space structure. GloVe (Global Vectors) was introduced to combine the best of both worlds, in order to create word embeddings that perform well

in word analogy, similarity, and named entity recognition tasks while efficiently leveraging statistical information.

Similar to LSA, the idea is to first construct a co-occurrence matrix X. While this matrix in LSA consists of term-document counts, entries X_{ij} for GloVe represent the number of times word j occurs in the context of word i. Therefore,

$$P_{ij} = P(j|i) = \frac{X_{ij}}{X_i} \tag{2.19}$$

embodies the probability that word j appears in the context of word i. Recalling that words that appear in similar contexts have similar meanings, we can use probe words k to learn about the relationship between words i and j. That is, if P_{ik} is similar to P_{jk}, i.e. P_{ik}/P_{jk} is close to one, k is related to both i and j, or to neither.

Taking these ratios of co-occurrences as a starting point, the model takes the form

$$F(\mathbf{w}_i, \mathbf{w}_j, \tilde{\mathbf{w}_k}) = \frac{P_{ik}}{P_{jk}}, \tag{2.20}$$

where $\mathbf{w} \in \mathbb{R}^d$ are word vectors and $\tilde{\mathbf{w}} \in \mathbb{R}^d$ are context words vectors. Since the goal of constructing word vectors is to represent analogies between words, F should depend on the difference between two target words. Furthermore, to prevent the mixing of vector dimensions as occurs in neural network-based algorithms, Pennington et al. argue that F should merely take the dot products of its arguments. We consequently define

$$F((\mathbf{w}_i - \mathbf{w}_j)^T \tilde{\mathbf{w}}_k) = \frac{P_{ik}}{P_{jk}} \tag{2.21}$$

Then, after assuming F is a homomorphism between groups $(\mathbb{R}, +$ and $(R_{>0}, \times)$, we can write

$$F((\mathbf{w}_i - \mathbf{w}_j)^T \tilde{\mathbf{w}}_k) = \frac{F(\mathbf{w}_i^T)\tilde{\mathbf{w}}_k}{F(\mathbf{w}_j^T)\tilde{\mathbf{w}}_k}, \tag{2.22}$$

which by (2.21) is solved by

$$F(\mathbf{w}_i^T \tilde{\mathbf{w}}_k) = P_{ik} = \frac{X_{ik}}{X_i}. \tag{2.23}$$

The rationality behind this transformation is that the roles of target and context words can be exchanged, which requires F to be invariant under the relabeling of $\mathbf{w} \leftrightarrow \tilde{\mathbf{w}}$ and $X \leftrightarrow X^T$

Finally, after solving this equation by applying $F = \exp$ and adding biases $\tilde{b_k}$ and $\tilde{w_k}$, GloVe becomes a global log-bilinear regression model, of which the cost function is posed as a least squares problem:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w_j} + b_i + \tilde{b_j} - \log X_{ij})^2, \tag{2.24}$$

where

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & \text{if } x < x_{\max} \\ 1 & \text{otherwise.} \end{cases} \tag{2.25}$$

The authors in [28] found that setting $x_{\max} = 100$ and $\alpha = 3/4$ delivers the best results and have made a multitude of models trained on large datasets publicly available.

2.2.4 FastText

Another shortcoming of Word2Vec models is that since the algorithm learns vector representations for words as a whole, the internal structure of these words is ignored. This can pose a problem when systems have to deal with morphological variations of words that have not been seen during training. Bojanowski et al. in [35] propose a new approach, named FastText, in which words are represented as a bag of character n-grams. This allows models to compute word representations for words that were not previously encountered.

FastText is highly similar to Word2Vec with negative sampling as discussed in 2.2.2. Instead of using full words, each word is represented by a combination of its character $n-grams$ and the full word itself. If $n = 4$ for example, a word such as *selling* would be represented as

```
<sel, sell, elli, llin, ling, ing>, <selling>
```

All character n-grams for $3 \leq n \leq 6$ are extracted. Then, if we take $\mathcal{G}_w \subset \{1, ..., G\}$ as the set of n-grams comprised in word w, and the vector representation $\mathbf{z}_g$ for each n-gram g, a word is is represented by the sum of the vectors of the n-grams it contains. The scoring function for a word is therefore defined as

$$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^T \mathbf{v}_c \tag{2.26}$$

Much memory would be required for the storage of all character n-grams, which is circumvented by the use of a hashing trick.

Although the original version of FastText was already seen as a solid improvement to Word2Vec, opportunities for development were taken in [36] by changing the architecture to resemble the CBOW method instead of CSG. This architecture could be slightly improved by the addition of position-dependent weighting as introduced in [37]. Furthermore, instead of only considering uni-grams that are insensitive to word order, the improved version of FastText merges some word n-grams during training as in [33] to further improve the quality of the learned word representations.

Chapter 3

Classification

Classification is one of the most common tasks in machine learning. The goal for this task is to automatically specify which category k some input belongs to [38]. For author verification, the input consists of a pair of texts, which should be categorized as either same-author or different-author. In this chapter, we discuss some classic classification algorithms as well as neural network-based techniques that have been applied to author verification and have been experimented with for this thesis.

3.1 Support Vector Machine

Support Vector Machines (SVM) [39] are classifiers that aim at finding an optimal separating hyperplane between two input vectors to distinguish between two groups. Only the data points closest to the separating hyperplane, named support vectors, are considered in this process. As shown in Figure 3.1, the optimal separating hyperplane maximizes each support vector's distance to the hyperplane. This distance is referred to as the margin.

Finding a separating plane in the form of $\mathbf{wx} - b = 0$ with classes clearly separated by a margin is only feasible in case of a linearly separable problem. Since this is often not the case, two tricks have been introduced. The first trick is known as the soft-margin hyperplane, and its idea is to permit some degree of error while constructing the separating hyperplane.

For the unforgiving hard-margin SVM, a set of labeled training samples

$$(y_1, \mathbf{x}_1), ..., (y_\ell, \mathbf{x}_\ell), \qquad y_i \in \{-1, 1\} \tag{3.1}$$

would only be considered linearly separable if there exists vector $\mathbf{w}$ and a scalar b such that the following holds:

$$\mathbf{wx}_i + b \begin{cases} \geq 1 & \text{if} \quad y_i \text{ is } 1, \\ \leq -1 & \text{if} \quad y_i \text{ is } -1. \end{cases} \tag{3.2}$$

Given this inequality can be written as:

$$y_i(\mathbf{wx} + b) \geq 1, \qquad i = 1, ..., \ell \tag{3.3}$$

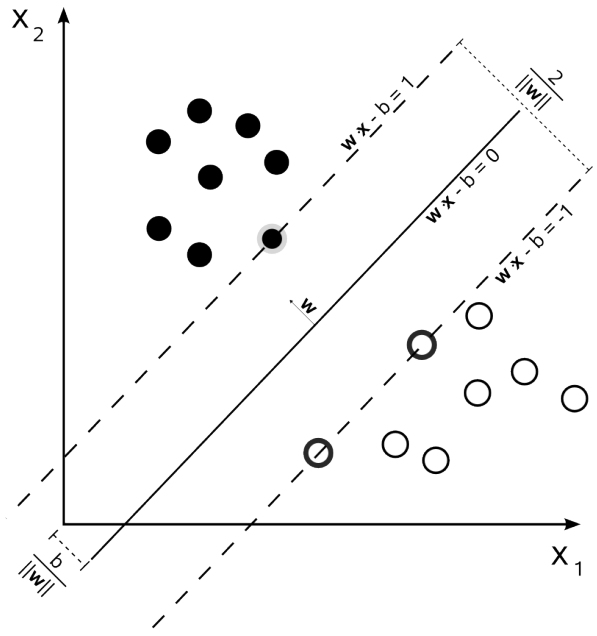


Figure 3.1: The Support Vector Machine (taken from [3])

the introduction of slack variable ξ_i creates constraints that are satisfiable while permitting some errors in the training data.

$$\begin{aligned} y_i(\mathbf{w}\mathbf{x} + b) &\geq 1 - \xi_i, \qquad i = 1, ..., \ell \\ \xi_i &\geq 0, \qquad i = 1, ..., \ell \end{aligned} \tag{3.4}$$

Finally, to maximize the margin $\frac{2}{||\mathbf{w}||}$, the cost function of the soft-margin SVM is defined as

$$\frac{1}{2}\mathbf{w}^2 + CF(\sum_{i=1}^{\ell} \xi_i^{\sigma}) \tag{3.5}$$

subject to the constraints in 3.4, and where $F(u)$ is a monotic convex function and C is a tuneable parameter. C controls the level error that is allowed during training. Increasing C will increase the penalty for misclassifications during training. The hyperparameter therefore influences the level of under- or overfitting, and is often an important hyperparameter to optimize.

Another trick employed by the SVM to allow for more accurate separation of non-linearly separable data is the possibility of replacing its *kernel function*. An example of a commonly used kernel function is the Radial Basis Function (RBF), which can be defined as

$$\exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2), \tag{3.6}$$

where γ is another tuneable hyperparameter. The value of γ controls the influence of single examples on the curvature of the polynomial decision surface, which might also have an effect on the generalization capacities of the model. No specific value for γ works for every dataset and the value is therefore often subject to optimization procedures as well.

3.2 Decision Tree

A method much more resemblant to the human decision-making process is known as the decision tree [40]. The idea behind it is to learn simple decision rules that together allow the model to make accurate predictions on complex data.

Each tree consists of a root node that represents the entire population. The training procedure is aimed at splitting this root node into branches, at the end of which are so-called *leaves* that contain samples separated by their class. The splits are based on single features that are strategically chosen based on properties such as impurity or information gain. The complexity of the model is in relation to the depth of the constructed decision tree. More splits and therefore deeper decision trees can more precisely model training data, but come with the risk of overfitting on this data.

Decision trees have shown to be capable of accurate classification in many scenarios. A much-appreciated property of decision trees is that they are completely white-box, in the sense that their decisions can be fully tracked. A weakness of decision trees however is that the risk of overfitting on training data is relatively high.

3.3 Random Forest

Random forests were introduced to mitigate the risk of overfitting decision tree models [41]. The essence of the random forest is to combine multiple decision trees into a single model. In these decision trees, a random subset of features is selected for each split, resulting in different decision trees. The trees are then averaged and can complement each other [42], acting as an ensemble.

Random forests unsurprisingly outperform decision tree models in most cases, mainly decreasing the chance of overfitting. The number of decision trees out of which the forest is constructed can be varied and is often subject to optimization procedures. Usually, more trees result in more accurate representations of data, but using many trees comes at the cost of significantly increased training times.

3.4 Neural Networks

Ever since the introductions of the perceptron [43] and back-propagation [44], neural networks have been applied to perform a wide variety of tasks. In this section, we will discuss the neural network-based techniques that are experimented with for this thesis.

3.4.1 Recurrent Neural Networks

For sequential data, time is an essential feature to the capture of much the information it holds. Natural language for example would lose much of its meaning when the words would not be presented in order. To capture time-based relations, research has led to the development of various types of neural networks capable of learning such relations. These networks, called Recurrent Neural Networks (RNN), aim to stay informed about previous inputs before processing the next.

An RNN does this by using its activation in a previous timestep for that of the current

step. More specifically, the activation is defined as

$$\mathbf{h}_t = \begin{cases} 0 & \text{if } t = 0 \\ \Phi(\mathbf{h}_{t-1}, \mathbf{x}_t) & \text{otherwise,} \end{cases} \tag{3.7}$$

where Φ is a nonlinear function such as composition of a logistic sigmoid with an affine transformation [45].

A problem that arises in the original RNNs is that as more time steps are taken, the gradient to be used for updates becomes increasingly small, and the training procedure of the network takes much time. This problem is known as the vanishing gradient problem [46], and new types of RNN networks were introduced to deal with it.

One example of such a network is named the Gated Recurrent Unit (GRU) [47]. Instead of processing all information seen in the past, as in the original RNN, the GRU implements an *update gate* z_t^j, which allows it to selectively remember information that is considered necessary for solving a problem. The activation of the j-th GRU unit in a network at time t is influenced by z_t^j, and is an interpolation between the previous activation and a candidate activation $\tilde{h}_t^j$:

$$\mathbf{h}_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \tilde{h}_t^j. \tag{3.8}$$

This capability, essentially the capability to 'forget' information that has been seen in the past, has led to improved results on various tasks [45].

3.4.2 Siamese Neural Networks

Many classification tasks, such as facial comparison, signature verification, and author verification, require some form of comparison. Siamese neural networks were designed to handle such tasks more naturally, by means of two identical sub-networks that share weights and are joined at their outputs [48]. Each sub-network learns features from two inputs, after which the difference between these features is used to make predictions.

The original Siamese neural networks were implemented with Time-Delay Neural Networks [49], but the sub-networks can be replaced by other types of architectures such as dense neural networks, convolutional neural networks (CNN), and GRUs. The Siamese neural network basically learns to embed its inputs such that the produced vectors of similar samples are close together in vector space, while unequal samples are not. After training a Siamese neural network, it is possible to detach the distance-calculating head from the sub-networks, allowing these networks to produce representations of inputs that hold information regarding their identity.

Chapter 4

Experimental Setup

This chapter describes the data and methodology used to answer the research questions mentioned in the introductory section. Section 4.1 describes the two datasets and presents some exploratory analysis. Section 4.2 discusses the trialed encoding techniques and clarifies the handcrafted features used in some of the models. Finally, section 4.3 describes the models and parameters employed during the classification of the encoded data.

4.1 Data Acquisition and Analysis

4.1.1 PAN 2020

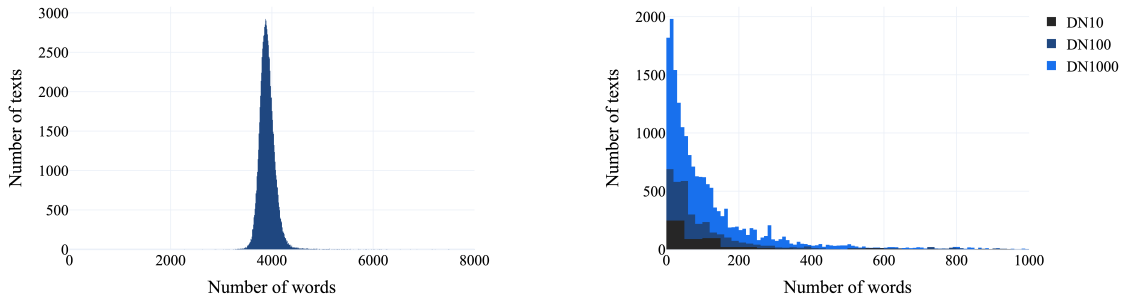
The first dataset was acquired from the International Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection (PAN). This workshop is held every year since 2007 and comprises various shared tasks related to NLP. The specific dataset used in this thesis originates from the PAN authorship verification task of 2020. The texts are stories crawled from *fanfiction.net*[1]: a website where users share stories in numerous thematic categories referred to as *fandoms*. There are 52601 pairs constructed from 93662 unique texts, written by 52655 unique authors. In total, there are around 367 million words. As shown in figure 4.1a, most texts contain between 3500 and 4500 words. Only a few texts contain more than 5000 words, with the largest extreme outlier reaching 55433 words. There are 27834 positive and 24767 negative samples, meaning the dataset is fairly balanced.

To facilitate the use of meta-features, the fandoms in which texts were written have been provided as well. There are 1600 unique fandoms, with examples such as *Sherlock Holmes*, *Psych*, and *Hunger Games*. An illustration of the distribution of meta-features can be found in the Appendix.

4.1.2 Darknet

The Darknet market data is provided by Web-IQ and was collected by scraping 17 Darknet markets between 2014 and 2020. It consists of millions of advertisements offering a wide range of illegal goods such as drugs, weapons, and counterfeit money. Since over 96% of the advertisements are written in English, we will not consider other languages. To investigate the influence of the number of unique authors on the performance of author verification methods, 3 datasets were constructed using texts written by 10, 100, and 1000 unique authors.

[1] `https://www.fanfiction.net/`



(a) PAN 2020.

(b) Darknet.

Figure 4.1: Histograms of word counts on the datasets.

For an author verification algorithm to be trained, pairs of text with positive, i.e. written by the same author, and negative examples are required. We follow [50] and create positive examples under the assumption that texts posted using a single username on one marketplace are written by one individual. A particularity of Darknet market advertisements however is that multiple advertisements written by one user might show extreme resemblance, as only a few terms are changed: sellers often use separate advertisements to offer various quantities of the same drug, append the same long disclaimer to multiple few-word offerings, or use template texts for every post. To exclude pairs of highly similar texts from our datasets, we employ the Jaccard similarity

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.1}$$

over the words as a heuristic. It was empirically determined that setting $J(A, B) < 0.2$ establishes text pairs that cannot be effortlessly matched by word counts and for which an author verification algorithm could provide a valuable solution.

Figure 4.2 shows the distribution of all advertisements in the Darknet market data. Most users post less than 50 advertisements. For each author, we calculate the possible number of positive examples P that could be generated under restriction $J(A, B) < 0.2$, and pick random authors with similar values for P for each of the three datasets. We define the range for P such that at least the required amount of authors can be selected for each dataset. This means that for the 10 authors dataset, from hereon named DN10, we select authors with $3500 < P < 3600$. For DN100 we take $1300 < P < 1400$ and for DN1000 choose $200 < P < 300$. After generating every possible positive pair for author A, we create negative examples by combining random texts of A with some random text of another author B until P is matched. As noted in [51], there is a possibility that B is in fact an alias of A. However, this is unlikely and if it is the case, will not enhance the results.

The DN10 dataset contains 71163 pairs, constructed from just 529 unique texts. It comprises around 15.4 million words. The DN100 dataset is built out of 3541 unique texts, forming 270895 pairs with around 53 million words. For DN1000, there are 18093 unique texts combined in 494840 pairs, containing a billion words. As shown in figure 4.1b, most texts in all three datasets consist of less than 200 words.

The Darknet market dataset contains many meta-features detailing the forum posts, such as extracted product prices, product tags, and product categories. However, since these

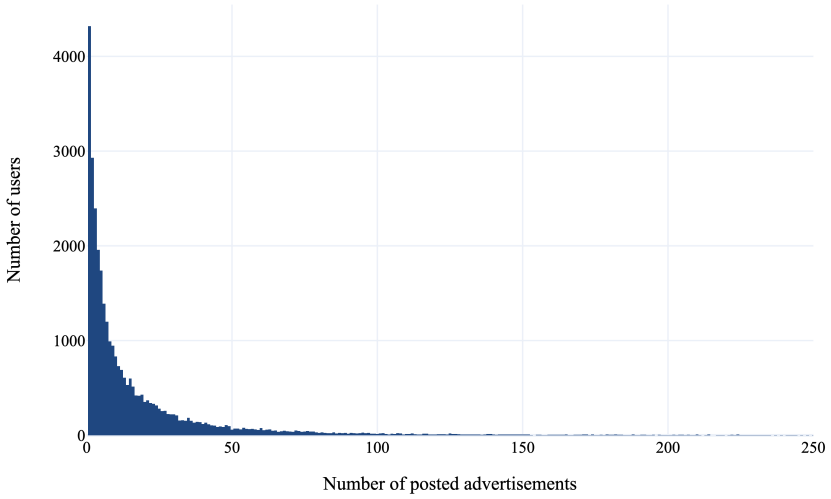


Figure 4.2: The distribution of dark marketplace advertisements.

properties are often missing because they could either not be extracted or were not present at all, we will only utilize the property *normalized_shipping_to*. This property was extracted for every forum post and describes to what regions sellers want to ship their products. This information could be valuable to an author verification system, as sellers will most likely determine where to send their products based on their personal beliefs about risks and costs imposed by sending to specific countries: some countries might be too far away from the senders own location, making shipping too expensive, while the laws and their enforcement in other countries around certain illegal goods might make sending them to these countries not worth the risk. There are 60 possible values for *normalized_shipping_to*, and their distribution is visualized in the Appendix.

4.2 Data Encoding

4.2.1 Stylometric Features

As mentioned in section 2.1, much research has been done on finding optimal sets of stylometric features to identify authors. We acknowledge these efforts, and instead, focus our research on finding a promising combination of representation and classification methods for author verification on Darknet market advertisements. Therefore we limit our set of handcrafted features to just nine different features that are known to be effective and are not domain -or topic-dependent [52]. The features are described in table 4.1. All features are *joint* features, which means they are single values calculated by comparing properties of two input texts.

For word and sentence tokenization we employ *word_tokenize* from the Natural Language Toolkit (NLTK) [53] and NLTK's *Punkt tokenizer* respectively. For the two features based on character n-grams, we take $M = 100$ and calculate the scores for uni, bi, and tri-grams,

Feature name	Description
Character n-gram dissimilarity	The dissimilarity between two normalized character n-gram distributions. The score is defined as $\sum_{i=1}^{M}(2 \cdot (f_{1,i} - f_{2,i})/(f_{1,i} + f_{2,i}))^2$, where f_1 and f_2 are the normalized frequencies of the top M most occurring n-grams (taken from [54]).
Character n-gram SPI	The Simple Profile Intersection [52] score of two character n-gram distributions. The score is defined as $\|F_1 \cap F_2\|$, where F_1 and F_2 are the sets of the M most frequent character n-grams for both texts (taken from [54]).
Character count difference	The absolute difference between the number of characters in each text.
Word count difference	The absolute difference between the number of words in each text.
Sentence count difference	The absolute difference between the number of sentences in each text.
Punctuation similarity	The cosine similarity between two vectors with normalized counts of punctuation marks $[, .?!; - :']$ in both texts (taken from [54]).
Word lengths difference	The absolute difference between the average lengths of the words in both texts.
Sentence lengths difference	The absolute difference between the average lengths of sentences in both texts.
Uppercasing difference	The absolute difference between the relative amount of uppercased characters in both texts.

Table 4.1: Definitions of handcrafted features.

which after combination with the other features lead to a 12-dimensional feature vector.

4.2.2 Embeddings

To confine the number of experiments done for this thesis, we only perform experiments using pre-trained word embeddings. The motivation behind this choice is that the quality of word embeddings highly depends on the amount and quality of the data they were trained on [55]. Considering the number of texts in the PAN 2020 dataset is small, and the uncurated posts on Darknet market forums are often full of noise, word embeddings pre-trained on other more commonly used datasets will most likely be of better quality.

For GloVe, we take 100-dimensional vectors trained on an English Wikipedia dump from 2014 and Gigaword 5 [56], a dataset containing newswire texts. In total, this training set consists of 6 billion tokens, of which the 400.000 most common words were used to create the co-occurrence matrix the algorithm relies upon. During training, a context of 10 words to the left and 10 words to the right was used while applying a decreasing weighting function such that word pairs that are d words apart contribute $1/d$ to the total count. The model was trained for 50 iterations.

The FastText embeddings were trained on texts provided by Common Crawl, an initiative that crawls the web for its full-text contents. More specifically, 2 million word vectors were trained using the 600 billion tokens in a Common Crawl dump of May 2017. The 300-

dimensional model was trained using CBOW with position weights, character n-grams of size 5, a window size of 5, and 10 negative samples. We convert the model to dimension 100 using the *reduce_model.py* script that is provided by the authors of [55].

4.2.3 Meta-features

We will research what influence features outside of texts can have by adding these meta-features to the inputs of our Siamese neural network architectures. For both datasets, this means we will transform the categorical feature into a one-hot encoded vector. This results in vectors of length 1600 and 60 for PAN 2020 and the Darknet market datasets respectively. We will specify how these vectors are used by the models in section 4.3.2.

4.2.4 Pre-processing

Because stylometric features are often expressed in details of texts, commonly used pre-processing methods such as lowercasing, stemming, and stop word removal would only limit the capture of author characteristics and thus negatively impact results. Therefore we use the raw texts as our input. For the Darknet market datasets, we prepend the title of each advertisement to its content to capture as much of each author's writing as possible.

As the models based on handcrafted features require their inputs to be standardized, we transform the 12-dimensional input feature vectors using Scikit-learn's *StandardScaler* to center their values around 0.

4.3 Models

This section describes the parameters of each predictive model that was experimented with and the methods used to find the optimal values for these parameters. First, we perform a grid search using *GridSearchCV* from Scikit-learn to search for the optimal parameters for each model. We use 60% of the data for training, 20% for validation, and 20% for testing, split by pair or author for the PAN 2020 and Darknet market datasets respectively. Given the PAN 2020 dataset consists of texts written by many different authors and only a handful of authors contributed more than two texts to the dataset, we can split the data by pair without risking that our model fits on specific authors. For the Darknet market datasets, written by just 10, 100, and 1000 authors, this risk does exist. Therefore we split this data by author to allow us to evaluate how well our models perform when dealing with texts of unknown authors. As discussed in 4.1.2, the datasets were constructed such that the number of text pairs is roughly the same for each author, which results in pair counts being proportionate to the author count in each split. During hyperparameter optimization, the validation set is only used as a means of determining an early stopping point for the Siamese neural network-based model: we stop training if the validation loss has not decreased in the last 4 epochs.

For the experiments with models with optimized parameters, we employ 5-fold cross-validation as implemented in Scikit-learn's *KFold* to minimize the risk of overfitting on subsets of the data. This means that 80% of the data will be used for training, and 20% for testing.

The receiver operating characteristic (ROC) curve and its area under the curve (AUC) are used for evaluation. Originally developed for operators of military radar receivers, hence its name, the ROC curve is a plot of the true positive rate (TPR) against the false positive rate (FPR). The AUC is the probability that a model ranks a randomly chosen positive sample

higher than a randomly chosen negative one [57]. A randomly guessing model would achieve an AUC score of 0.5, while 1.0 would be considered a perfect score. The main advantage of AUC over other measurements such as accuracy is that there is no need to define a decision boundary for evaluation, as the class probabilities can be used directly. While there are some drawbacks to using AUC [58], we follow the organizers of PAN 2020 [59] and consider it a suitable method for evaluating authorship verification models.

4.3.1 Handcrafted Features

Three well-known classification algorithms are experimented with to assess the potency of models based on handcrafted features in author verification systems.

Support Vector Machine

Scikit-learn's *SVC* is employed as implementation for the SVM. As discussed in section 3.1, the parameters most influential on the SVM's performance are the kernel, the regularization parameter C, and the level of influence of single training samples, Gamma. We note that for the linear kernel, Gamma is not applicable and will therefore not be specified if a model with a linear kernel appears in the final results. Table 4.2 displays the sets of parameters that are experimented with for this thesis.

Parameter	**Values**
Kernel	linear, RBF
C	1e−3, 1e−2, 1e−1, 1, 10
Gamma	1e−3, 1e−2, 1e−1, 1

Table 4.2: Parameter search space for the SVM classifier.

Decision Tree

The *DecisionTreeClassifier* from Scikit-learn serves as our model for decision tree classifiers, and we optimize two parameters that control the structure of the tree, which are listed in table 4.3. Max. depth is the maximum depth of the tree, which can limit the complexity and influence the generalization capabilities of the model. Min. samples split is the minimum number of samples required to split an internal node. This value affects whether a model under- or overfits as a decision tree that contains splits based on too few samples might model relations that are highly specific to a training set, while insufficient splitting might prevent models from learning crucial relations at all.

Parameter	**Values**
Max. depth	10, 100, 1000
Min. samples split	2, 5, 10

Table 4.3: Parameter search space for the decision tree classifier.

Random Forest

We train random forest models using Scikit-learn's *RandomForestClassifier*. The ratio of features to be considered at each split is set at the default value $\sqrt{N}$, where N is the total number of features. The parameters prone to optimization are shown in table 4.4. We take larger values for Min. samples split than in the decision tree classifier, as an ensemble of decision trees should have a reduced risk of overfitting in comparison to a single tree. Lastly, we vary the number of estimators that are allowed in the ensemble.

Parameter	**Values**
Max. depth	10, 50, 100, 200
Min. samples split	2, 5, 10
Estimators	10, 100, 1000

Table 4.4: Parameter search space for the random forest classifier.

4.3.2 Siamese Neural Network

We model the comparative nature of author verification tasks using a Siamese neural network, shown in figure 4.3. The *Keras* framework [60] is used for its implementation. First, we tokenize both texts using *Keras*' built-in tokenizer. The embeddings of the tokens are then retrieved from the pre-trained GloVe or FastText models. Next, these embeddings are passed through a GRU, of which the final state is the input to a dense layer on each side of the network. The network is trained by means of the contrastive loss function as described in [61]. This loss function is designed to decrease the euclidean distance between the output vectors of equally labeled samples while increasing it for unequally labeled pairs, making it suitable for tasks that are of comparative nature. In a production environment, the output of the network can be a Euclidean distance value, which can be thresholded such that text pairs are classified with preferred levels of sensitivity and specificity.

The possible benefit of using meta-features in author verification systems is investigated by adding them to the inputs of the Siamese neural network-based models. As mentioned in section 4.2.3, this is done by first transforming each meta-feature into a one-hot encoded

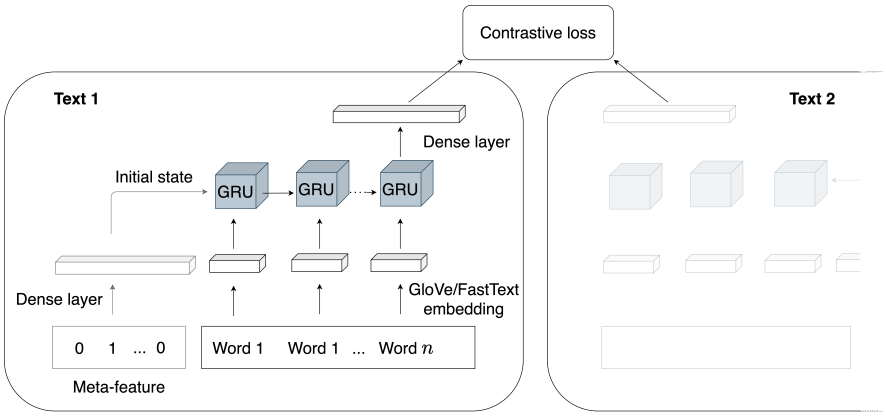


Figure 4.3: The structure of the Siamese neural network employed for author verification.

vector. Some Darknet advertisements have multiple values for *normalized_shipping_to*, but only the first value will be used during our experiments. The one-hot encoded vector is embedded by a dense layer with the same dimensions as the hidden layer in the GRU. The resulting vector is used as the initial state in the GRU.

The Adam optimizer [62] is employed for optimization. We use a batch size of 128. As mentioned earlier, during grid search we use the validation data to determine the number of epochs to train the network for the final experiments. We use the stopping point from the best-performing model in the grid search as the stopping point in the subsequent experiments.

During hyperparameter optimization, we vary the number of hidden units in the GRU, the dropout ratio within the GRU, and the number of output units in the dense layer following the GRU. Table 4.5 displays the values in the search space.

Parameter	**Values**
Hidden units	32, 64, 128
Output units	32, 64, 128
Dropout	1e−1, 2e−1, 4e−1

Table 4.5: Parameter search space for the Siamese neural network.

Chapter 5

Results

This chapter describes the results of the experiments as described in the previous chapter. Section 5.1 will present results obtained through models that were trained using handcrafted features. Next, section 5.2 displays the performance of Siamese neural network-based models utilizing GloVe and FastText embeddings, after which the influence of adding a meta-feature will be presented in section 5.4. Finally, we compare the results from each model type in section 5.3.

As the experiments performed for this thesis were extensive, only the results of the models that were selected after an optimization procedure will be shown in this chapter. The full list of results can be found in the Appendix. As mentioned in the previous chapter, all AUC scores presented in this section were obtained through 5-fold cross-validation. In **bold** are the highest scores achieved for each dataset.

5.1 Handcrafted Features

In this section, we present the results of models trained using vectors constructed using handcrafted features. For each dataset, we show the best-performing parameters found during grid searches and the scores that were produced using these parameters.

5.1.1 Support Vector Machines

The results obtained using SVM models are displayed in table 5.1. For three of the four datasets, the best performing models are trained with parameter C set at 10, the largest value in the search space, while for DN10 a C value of 1 returned the highest AUC. Setting a high value for C, such as 10, implies an SVM will favor the correct classification of individual training samples over a large margin while constructing its separating hyperplane. Since the largest value was selected in all but one case, it appears the search space was not broad enough, and higher values for C should have been experimented with. Very high values for C however could result in overfitting, and it should be noted that K-fold cross-validation during grid search could have led to different outcomes.

For the DN100 dataset, a linear kernel performed best. A Gamma value of 0.01 returns the best performing models for all models with an RBF kernel. The SVM delivers its highest AUC on the PAN 2020 dataset. For the Darknet market datasets, the performance is comparable for 10, 100, and 1000 authors, but it should be noted that the AUC for 10 authors is worst.

Dataset	C	Gamma	Kernel	AUC
PAN 2020	10	1e−2	RBF	0.853
DN10	1	1e−2	RBF	**0.718**
DN100	10		linear	**0.752**
DN1000	10	1e−2	RBF	0.745

Table 5.1: Parameters and results of the best performing SVM models on the four datasets.

This is in line with expectations, as training on texts written by just six authors will most likely not allow a model to generalize well enough to make accurate predictions on texts written by two other authors. However, as the boldfacing gives away, none of the other tested models is able to outperform the SVM on the DN10 and DN100 datasets.

5.1.2 Decision Trees

Table 5.2 shows the selected parameters and their results for decision tree models. For the DN10 dataset, the best results during optimization were obtained using a model with depth one and Min. samples split set at 10, but after cross-validation, this seems to have a negative effect on performance. A decision tree of depth 1 allows just one feature to be considered before a decision is made. It is plausible that this single feature accurately discriminates texts of the two authors selected as a test set during hyperparameter optimization, but fails to do so for other pairs of authors encountered during cross-validation.

The texts in the Darknet market datasets were written without any guidelines and are full of noise, which is disadvantageous to decision tree classifiers, as they are susceptible to minor changes in data. This is exemplified by the AUC scores being lower than those achieved by the SVM on the same data.

Dataset	Max depth	Min samples split	AUC
PAN 2020	10	2	0.803
DN10	1	10	0.580
DN100	10	5	0.686
DN1000	10	5	0.698

Table 5.2: Parameters and results of the best performing decision tree models on the four datasets.

5.1.3 Random Forests

The selected parameters and results for random forest classifiers can be found in table 5.3. For these classifiers, the optimal number of estimators seems to be related to the number of authors: for the Darknet datasets these numbers match, and for the PAN 2020 dataset the highest number of estimators has been selected, as this dataset contains texts written by many authors. More experiments with differents author counts should however be performed before this correlation can be assumed.

The random forest classifier outperforms the decision tree classifier on every dataset. This was expected, since the random forest classifier is an ensemble of decision trees, and as

Dataset	Max depth	Min samples split	# Estimators	AUC
PAN 2020	50	2	1000	0.854
DN10	10	10	10	0.691
DN100	10	10	100	0.750
DN1000	200	2	1000	0.743

Table 5.3: Parameters and results of the best performing random forest models on the four datasets.

mentioned in section 3.3 was designed to decrease the risk of overfitting decision tree-based models.

5.2 Siamese Neural Networks

5.2.1 GloVe Embeddings

Table 5.4 displays the selected parameters and results obtained by Siamese neural networks based on GRUs that were fed GloVe embeddings. The number of epochs needed for training seems to be related to the lengths of the texts, as the stopping point for PAN 2020, with its average text length of 4000 words, is reached after 19 epochs while the validation loss stopped increasing much earlier when training on Darknet market data.

While a pattern seems to be visible in the increasing values for dropout as the number of authors of the Darknet market datasets increases, further inspection of the grid search result as presented in the Appendix reveals this pattern is highly likely to be coincidental.

The Siamese neural network-based model performs well on the PAN 2020 set, with a promising AUC score of 0.889, outperforming all other model types. This indicates that word embeddings and GRUs can be capable of representing writing styles when fed with enough examples. Looking at the lower scores for DN10 and DN100, the necessity for these examples to be diverse is concretized.

Dataset	Hidden units	Output units	Dropout	Epochs	AUC
PAN 2020	128	32	2e−1	19	**0.889**
DN10	32	128	1e−1	6	0.605
DN100	128	128	2e−1	4	0.660
DN1000	32	128	4e−1	6	0.766

Table 5.4: Parameters and results of the best performing Siamese neural network models with GloVe embeddings on the four datasets.

5.2.2 FastText Embeddings

We present the selected parameters and results obtained using the Siamese neural network-based model combined with FastText embeddings in table 5.5. The number of epochs needed for training shows a similar pattern as for the GloVe-based model, albeit with an even larger

Dataset	Hidden units	Output units	Dropout	Epochs	AUC
PAN 2020	64	64	2e−1	31	0.860
DN10	64	32	2e−1	6	0.674
DN100	32	64	4e−1	5	0.633
DN1000	32	32	4e−1	5	**0.769**

Table 5.5: Parameters and results of the best performing Siamese neural network models with FastText embeddings on the four datasets.

amount of epochs needed for the PAN 2020 dataset. No pattern is discovered in the network parameters.

For DN1000, the model slightly improves on the GloVe-based model, as well as outperforming all other model types with an AUC of 0.769. Furthermore, the FastText-based model improves on GloVe for DN10 but performs worse on DN100. Performance on both DN10 and DN100 can be considered weak, further strengthening the belief that more diverse training data is needed for neural network-based models to become successful.

5.3 Model Comparison

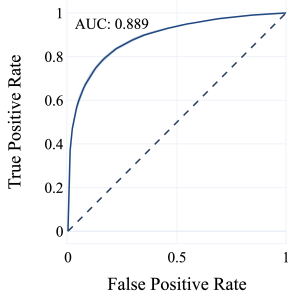
The best performing model types with their mean AUC scores and standard deviations are summarized in table 5.6. We only consider the models trained without the use of meta-features for a fair comparison. Figure 5.1 visualizes the ROC curves of each model.

Siamese neural network-based models outperform models relying on handcrafted features for PAN 2020 and DN1000. We note however that the difference in performance with SVM and random forest models is small. For DN10 and DN100, the lack of diverse examples impacts the performance of the neural network-based models, and the SVM and random forest models surpass these models in terms of AUC scores. Scores are not the only matter to consider when selecting models for author verification in production environments, as we will further discuss in section 6.1.

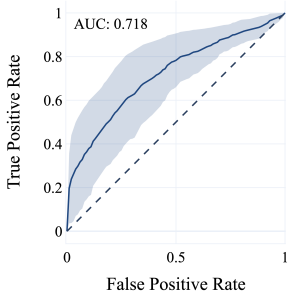
The standard deviations and ROC curves of the best performing models show that not just the scores, but also the variation within these scores are clues for the difficulty of training well-generalizing models on data containing little diversity. For PAN 2020, performance across all folds is practically equal. The same applies to DN1000, which is to be expected, as the data in both of these datasets is constructed using texts written by many authors. This allows models to learn to represent writing styles in general, irrespective of particular sets of authors by which the texts in the training set were written. Recalling that these sets of authors

Dataset	Model type	AUC	SD
PAN 2020	GRU + GloVe	0.889	0.005
DN10	SVM	0.718	0.120
DN100	SVM	0.752	0.025
DN1000	GRU + FastText	0.769	0.004

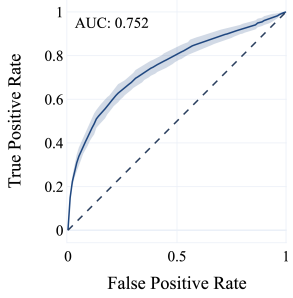
Table 5.6: Scores and types of the best performing models overall on the four datasets.



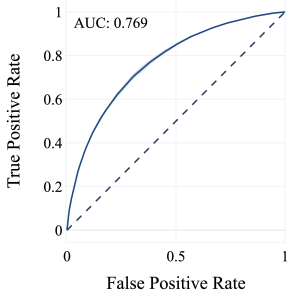
(a) PAN 2020.



(b) DN10.



(c) DN100.



(d) DN1000.

Figure 5.1: ROC curves for each dataset. The curves are produced by the model types that performed the best on each particular dataset. The types of these models are specified in table 5.6. In each figure, the blue line represents the mean ROC curve, while the transparent areas resemble the difference between the upper and lower ROC curves of the models tested during 5-fold cross-validation.

comprise a mere 6 and 60 authors for DN10 and DN100 respectively, the higher deviations in scores are logical.

Another observation is that the AUC scores on average are higher for PAN 2020 than for the Darknet market datasets. Actual participation in the PAN shared task failed because of deadlines that came too early in our research, denying our models to be tested on the hidden test set. However, what the higher AUC scores do tell us is that Siamese neural networks based on word embeddings can be capable of representing author writing styles, and the lower scores on the Darknet market data are not necessarily caused by weak model architectures.

In a production environment, a threshold would have to be set on the output score of the Siamese neural network to perform classification. As an example, we chose a threshold of 0.55 and performed classification using the GRU + FastText model trained on 369052 samples written by 800 authors from the DN1000 dataset. This model was tested on 98788 text pairs written by 200 different authors from the same dataset. The result was a precision score of 0.679 with a recall score of 0.748, represented by an F_1 score of 0.712. Further inspection of the predictions reveals that the model is capable of recognizing authors successfully if there are

related products being sold, if the advertisements show a similar structure, or when sentences are built up in a similar manner. More problematic are pairs of texts with a negative label that both consist of fewer than five words, as it appears there is not enough information available to distinguish among texts, and false positives occur as the Euclidean distance between the two output vectors is small. In other cases, where one text is long and the other is short, false negatives occur as the Euclidean distance between the two output vectors is large.

5.4 Influence of Meta-features

Table 5.7 shows that using meta-features as initial states in the GRUs at the base of Siamese neural networks can significantly affect performance. For the GRU + GloVe model, employing meta-features leads to a higher AUC score for every dataset. The improvement for PAN 2020 is minor. For the GRU + FastText model, meta-features lead to a decreased AUC. Most promising are the scores achieved on the DN100 and DN1000 datasets, where the additional information results in clearly improved scores reaching above 0.9.

Dataset	**AUC**			
	GRU + GloVe		**GRU + FastText**	
	best	+ meta	best	+ meta
PAN 2020	0.889	**0.898**	**0.860**	0.852
DN10	0.605	**0.652**	**0.674**	0.647
DN100	0.660	**0.942**	0.633	**0.951**
DN1000	0.766	**0.929**	0.769	**0.926**

Table 5.7: AUC scores of the best performing Siamese neural network models after embeddings of meta-features have been added as initial states for the GRUs that are the base of these models. In **bold** are the best performing models.

Chapter 6

Discussion and Conclusion

6.1 Discussion

This chapter will discuss the outcomes of the experiments presented in the previous chapter. The questions posed in the introductory section are answered, after which directions for future research are suggested based on the evaluations of the methods and procedures as they were experimented with for this thesis.

6.1.1 Research Questions

What combination of representation and classification methods can most accurately verify authors of short texts?

Based on our investigation, we find that when training an author verification system on texts written by a small set of authors, texts can best be represented by handcrafted stylometric features, and the SVM and random forest classifiers deliver the most accurate predictions on these vectors. In scenarios with large sets of authors, such as in the PAN 2020 and DN1000 datasets, models based on Siamese neural networks fed with word embeddings deliver the best results. These finds are in line with the general belief that deep neural networks are capable of outperforming classical machine learning methods, given large amounts of data.

No extensive research was done on the particular set of stylometric features to be used in the classical machine learning classifiers. The fact that some of these classifiers were still capable of capturing and differentiating author writing styles quite well exemplifies the power of handcrafted features. On the other hand, GloVe and FastText embeddings combined with GRUs have shown to be capable of capturing these unique features as well, albeit in a completely different manner. No significant difference between using GloVe and FastText embeddings at the base of the network has been found. While for DN1000 the model using FastText embeddings outperformed the model using GloVe embeddings, the difference in scores is marginal and could be caused by other network parameters.

Looking at the scenario for which this research was conducted, training a Siamese neural network to represent writing styles is most appropriate. Given Web-IQ has access to millions of advertisements posted on the dark web, written by thousands of authors, models can be trained on vast amounts of generated positive and negative examples. Another advantage of the Siamese neural network employed during our research is that its two branches both aim to learn to transform their inputs such that their output embeddings represent the identity

of authors. Given the head of the network is just a function that calculates the Euclidean distance between these embeddings, it can be detached, allowing the network to produce these identity embeddings for each text on the dark web in advance. These vectors could then be used in searches and comparisons at any given moment. A downside to using the Euclidean distance however is that short texts and text pairs with large differences in lengths can lead to false positives and false negatives respectively. Training a network and performing classification based on measures that work with angles instead of distances, such as cosine similarity, might have countered these issues.

Can metadata collected from Darknet market advertisements be of use in an author verification system?

Our experiments show that meta-features can most certainly be of use in author verification systems used on the dark web. Given that just a single meta-feature was employed as additional input, the score improvement that it has caused for the larger Darknet datasets is astounding. The significant influence of a single meta-feature is most likely strengthened by the shortage of information in the short texts of Darknet market advertisements. For the PAN 2020 dataset, we found no significant positive effect, showing that for longer texts, that hold more information by themselves, the influence of a single almost unprocessed meta-feature is diminished.

How well can authors of Darknet market advertisements be recognized using available text and metadata?

The experimental results acquired during this research show that it is possible to accurately recognize authors of Darknet market advertisements using the available text and metadata. While models trained on just text do not perform well enough to be of much use in a production environment, the addition of just one meta-feature already significantly improves results to a level that could be considered sufficient.

Considering no cross-validation was performed during grid searches, there is still room for improvement. It is questionable whether or not the best hyperparameters have been chosen for the final experiments, as the scores achieved with the best performing hyperparameter settings during grid search were often not matched by those acquired during the final experiments, especially for the smaller Darknet market datasets.

The maximum number of authors considered during our experiments was 1000. As the results show that the performance of the Siamese neural network-based approach improves with the increase in the number of authors, it could be that even higher scores could be achieved when models are trained on texts written by much larger sets of authors.

While the results acquired in some experiments are promising, it should be noted that our experiments were conducted using artificially balanced data. In the real world, examples of multiple accounts being used by one individual are most likely scarce. It would be interesting to see how the models tested in our experiments would perform when running in this environment, but we acknowledge that evaluating such experiments is difficult, considering no accurate information on the actual distribution and identity of Darknet market sellers exists yet.

6.1.2 Future Work

The experiments run during this research show that multiple techniques are useful for author verification systems used on Darknet market advertisements. A direction for future research could be to increase the resources available during training. It could be interesting to research the effect of adding texts of more authors to the training data. Furthermore, combining more meta-features in the inputs of the Siamese neural network could lead to more accurate representations of the identity of authors.

Adding more resources is an interesting but rather obvious direction for future research. Changing the way the already available data is used however might lead to stronger models that could be used regardless of the amount of available training data. The first way in which this could be achieved is by further experimenting with different types of text representations. There exist a plethora of stylometric features that could be experimented with, and many types of new character, word, and sentence embeddings have been introduced in recent years, each of which might improve the quality of author writing style representations.

Another opportunity lies in the way in which meta-features are used by the model. In this thesis, the meta-feature was simply used in its categorical form, disallowing the model to use information that might be contained in the meaning of the meta-feature itself. The current model does learn some form of meaning in the dense layer to which the one-hot encoded meta-feature vector is fed, but a more interesting approach would for example be to first encode meta-features in a way that captures more of their original meaning. For the PAN dataset, the *fandom* meta-feature might be represented by word embedding vectors, while for the Darknet market data, the *normalized_shipping_to* meta-feature could be represented by some embedding representing its geographic location.

Finally, it would be interesting to combine the best of both worlds by adding handcrafted stylometric features to the input of the Siamese neural network. Our research has shown that these features are of much use to classical classifiers, and it is most likely the case that they will complement the word embeddings and meta-features already used in the Siamese neural network-based model.

6.2 Conclusion

In this thesis, we aimed at finding the most suitable approach for linking profiles of Darknet market vendors using the texts and metadata of their advertisements. Results show that while classical representation methods provide the best results in scenarios with a limited number of authors, a more complex method based on Siamese neural networks and word embeddings is favorable when dealing with more diverse data. Adding meta-features to the input of these models significantly improves performance. Steps can be taken to improve upon our results, and we suggested some future directions of research based on the results acquired and lessons learned during this research.

Bibliography

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[2] X. Rong, "word2vec parameter learning explained," *arXiv preprint arXiv:1411.2738*, 2014.

[3] Cyc, "Graphic showing the maximum separating hyperplane and the margin." Feb. 2008. [Online]. Available: https://commons.wikimedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png

[4] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous connections and onion routing," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 482–494, May 1998, conference Name: IEEE Journal on Selected Areas in Communications.

[5] A. Baravalle, M. S. Lopez, and S. W. Lee, "Mining the Dark Web: Drugs and Fake Ids," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, Dec. 2016, pp. 350–356, iSSN: 2375-9259.

[6] X. H. Tai, K. Soska, and N. Christin, "Adversarial Matching of Dark Net Market Vendor Accounts," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage AK USA: ACM, Jul. 2019, pp. 1871–1880.

[7] T. Hume, "How the FBI caught Ross Ulbricht, alleged creator of Silk Road," May 2013. [Online]. Available: https://www.cnn.com/2013/10/04/world/americas/silk-road-ross-ulbricht/index.html

[8] J. Broséus, D. Rhumorbarbe, C. Mireault, V. Ouellette, F. Crispino, and D. Décary-Hétu, "Studying illicit drug trafficking on Darknet markets: Structure and organisation from a Canadian perspective," *Forensic Science International*, vol. 264, pp. 7–14, Jul. 2016.

[9] K. Soska and N. Christin, "Measuring the longitudinal evolution of the online anonymous marketplace ecosystem," in *Proceedings of the 24th USENIX Conference on Security Symposium*, ser. SEC'15. USA: USENIX Association, Aug. 2015, pp. 33–48.

[10] C. Aliens, "The Darknet Search Engine 'Grams' is Shutting Down - Deep Dot Web," Jan. 2018. [Online]. Available: https://web.archive.org/web/20180124070700/https://www.deepdotweb.com/2017/12/15/darknet-search-engine-grams-shutting/

[11] F. Iqbal, L. A. Khan, B. C. M. Fung, and M. Debbabi, "E-mail authorship verification for forensic investigation," in *Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10*. Sierre, Switzerland: ACM Press, 2010, p. 1591. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1774088.1774428

[12] M. L. Brocardo, I. Traore, S. Saad, and I. Woungang, "Authorship verification for short messages using stylometry," in *2013 International Conference on Computer, Information and Telecommunication Systems (CITS)*, May 2013, pp. 1–6.

[13] B. Boenninghoff, R. M. Nickel, S. Zeiler, and D. Kolossa, "Similarity learning for authorship verification in social media," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 2457–2461.

[14] J. A. Stover, Y. Winter, M. Koppel, and M. Kestemont, "Computational authorship verification method attributes a new work to a major 2nd century A frican author," *Journal of the Association for Information Science and Technology*, vol. 67, no. 1, pp. 239–242, 2016, iSBN: 2330-1635 Publisher: Wiley Online Library.

[15] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975.

[16] T. Joachims, "Text categorization with Support Vector Machines: Learning with many relevant features," in *Machine Learning: ECML-98*, ser. Lecture Notes in Computer Science, C. Nédellec and C. Rouveirol, Eds. Berlin, Heidelberg: Springer, 1998, pp. 137–142.

[17] H. P. Luhn, "A Statistical Approach to Mechanized Encoding and Searching of Literary Information," *IBM Journal of Research and Development*, vol. 1, no. 4, pp. 309–317, Oct. 1957, conference Name: IBM Journal of Research and Development.

[18] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, Jan. 1988, publisher: Pergamon.

[19] R. Zheng, J. Li, H. Chen, and Z. Huang, "A framework for authorship identification of online messages: Writing[U+2010]style features and classification techniques," *Journal of the American society for information science and technology*, vol. 57, no. 3, pp. 378–393, 2006, iSBN: 1532-2882 Publisher: Wiley Online Library.

[20] O. de Vel, A. Anderson, M. Corney, and G. Mohay, "Mining e-mail content for author identification forensics," *ACM SIGMOD Record*, vol. 30, no. 4, pp. 55–64, Dec. 2001.

[21] N. Cheng, R. Chandramouli, and K. P. Subbalakshmi, "Author gender identification from text," *Digital Investigation*, vol. 8, no. 1, pp. 78–88, Jul. 2011.

[22] F. Mosteller and D. L. Wallace, "Inference in an Authorship Problem," *Journal of the American Statistical Association*, vol. 58, no. 302, pp. 275–309, 1963, publisher: [American Statistical Association, Taylor & Francis, Ltd.].

[23] M. Koppel and J. Schler, "Authorship verification as a one-class classification problem," in *Proceedings of the twenty-first international conference on Machine learning*,

ser. ICML '04. New York, NY, USA: Association for Computing Machinery, Jul. 2004, p. 62.

[24] M. Koppel, J. Schler, and E. Bonchek-Dokow, "Measuring Differentiability: Unmasking Pseudonymous Authors," *The Journal of Machine Learning Research*, vol. 8, pp. 1261–1276, Dec. 2007.

[25] J. Li, R. Zheng, and H. Chen, "From fingerprint to writeprint," *Communications of the ACM*, vol. 49, no. 4, pp. 76–82, Apr. 2006.

[26] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954, iSBN: 0043-7956 Publisher: Taylor & Francis.

[27] J. R. Firth, "A synopsis of linguistic theory, 1930-1955," *Studies in linguistic analysis*, 1957, publisher: Basil Blackwell.

[28] J. Pennington, R. Socher, and C. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://www.aclweb.org/anthology/D14-1162

[29] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990, iSBN: 0002-8231 Publisher: Wiley Online Library.

[30] S. T. Dumais, "Latent semantic analysis," *Annual review of information science and technology*, vol. 38, no. 1, pp. 188–230, 2004, iSBN: 0066-4200 Publisher: Wiley Online Library.

[31] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *The journal of machine learning research*, vol. 3, pp. 1137–1155, 2003, iSBN: 1532-4435 Publisher: JMLR. org.

[32] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 160–167.

[33] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *arXiv preprint arXiv:1310.4546*, 2013.

[34] M. U. Gutmann and A. Hyvärinen, "Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics." *Journal of Machine Learning Research*, vol. 13, no. 2, 2012, iSBN: 1532-4435.

[35] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017, iSBN: 2307-387X Publisher: MIT Press.

[36] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," *arXiv preprint arXiv:1712.09405*, 2017.

[37] A. Mnih and K. Kavukcuoglu, "Learning word embeddings efficiently with noise-contrastive estimation," *Advances in neural information processing systems*, vol. 26, pp. 2265–2273, 2013.

[38] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, issue: 2.

[39] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[40] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986, iSBN: 1573-0565 Publisher: Springer.

[41] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[42] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1, issue: 10.

[43] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958, iSBN: 1939-1471 Publisher: American Psychological Association.

[44] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986, iSBN: 1476-4687 Publisher: Nature Publishing Group.

[45] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[46] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994, iSBN: 1045-9227 Publisher: IEEE.

[47] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[48] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," *Advances in neural information processing systems*, vol. 6, pp. 737–744, 1993.

[49] I. Guyon, P. Albrecht, Y. Lecun, J. S. Denker, and W. Hubbard, "A time delay neural network character recognizer for a touch terminal," in *Proceedings of the International Neural Network Conference, Paris, June 1990*, 1990.

[50] F. Johansson, L. Kaati, and A. Shrestha, "Detecting multiple aliases in social media," in *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*. IEEE, 2013, pp. 1004–1011.

[51] M. Spitters, F. Klaver, G. Koot, and M. van Staalduinen, "Authorship Analysis on Dark Marketplace Forums," in *2015 European Intelligence and Security Informatics Conference*, Sep. 2015, pp. 1–8.

[52] E. Stamatatos, "A survey of modern authorship attribution methods," *Journal of the American Society for Information Science and Technology*, vol. 60, no. 3, pp. 538–556, 2009, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/asi.21001. [Online]. Available: http://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.21001

[53] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[54] M. Hurlimann, B. Weck, E. van den Berg, S. Suster, and M. Nissim, "pan-webis-de/huerlimann15," May 2021, original-date: 2015-09-06T06:44:31Z. [Online]. Available: https://github.com/pan-webis-de/huerlimann15

[55] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," *arXiv preprint arXiv:1802.06893*, 2018.

[56] R. Parker, D. Graff, J. Kong, K. Chen, and K. Maeda, "English Gigaword Fifth Edition," type: dataset. [Online]. Available: https://catalog.ldc.upenn.edu/LDC2011T07

[57] T. Fawcett, "An introduction to ROC analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006, iSBN: 0167-8655 Publisher: Elsevier.

[58] J. M. Lobo, A. Jiménez[U+2010]Valverde, and R. Real, "AUC: a misleading measure of the performance of predictive distribution models," *Global ecology and Biogeography*, vol. 17, no. 2, pp. 145–151, 2008, iSBN: 1466-822X Publisher: Wiley Online Library.

[59] M. Kestemont, E. Manjavacas, I. Markov, J. Bevendorff, M. Wiegmann, E. Stamatatos, M. Potthast, and B. Stein, "Overview of the Cross-Domain Authorship Verification Task at PAN 2020," in *CLEF*, 2020.

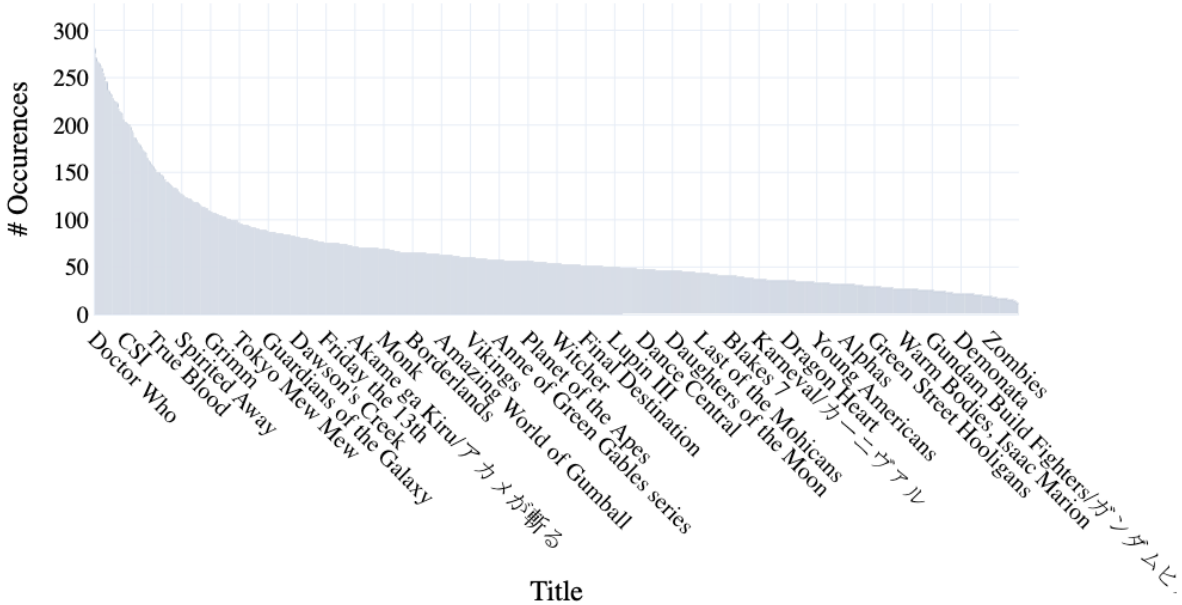
[60] F. Chollet and others, *Keras*, 2015. [Online]. Available: https://keras.io

[61] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742.

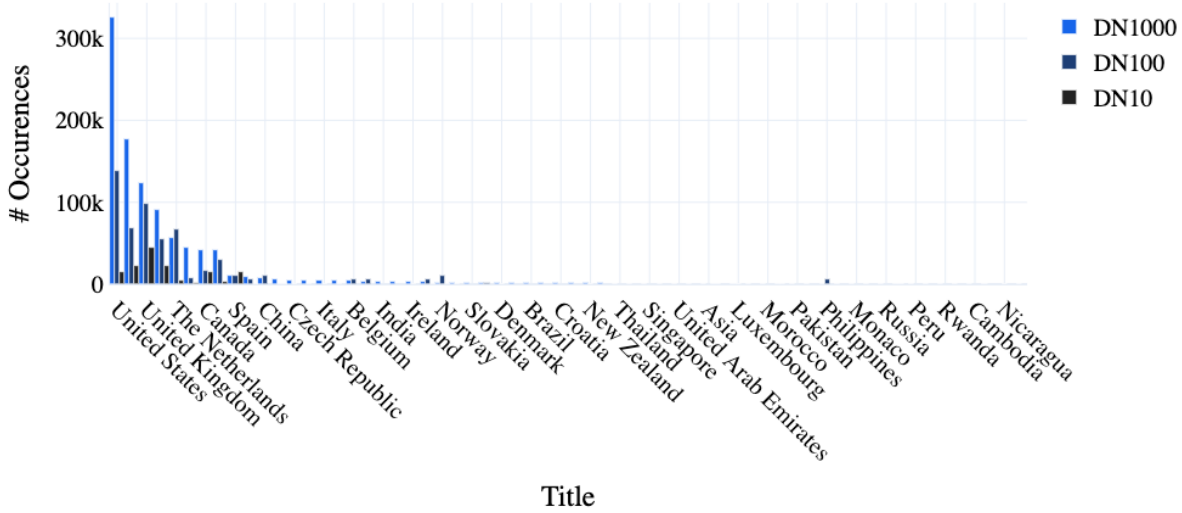
[62] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

Appendices

Figures



(a) PAN 2020.



(b) Darknet.

Figure 1: The distribution of meta-feature values in the datasets.

Full Experiment Results

Figure 2: A visualisation of hyper-parameter optimization experiments of SVMs performed on the PAN 2020 dataset. The left figure shows SVMs with a linear kernel. The right figure shows SVMs with an RBF kernel. In red are the AUC scores of the best performing models.

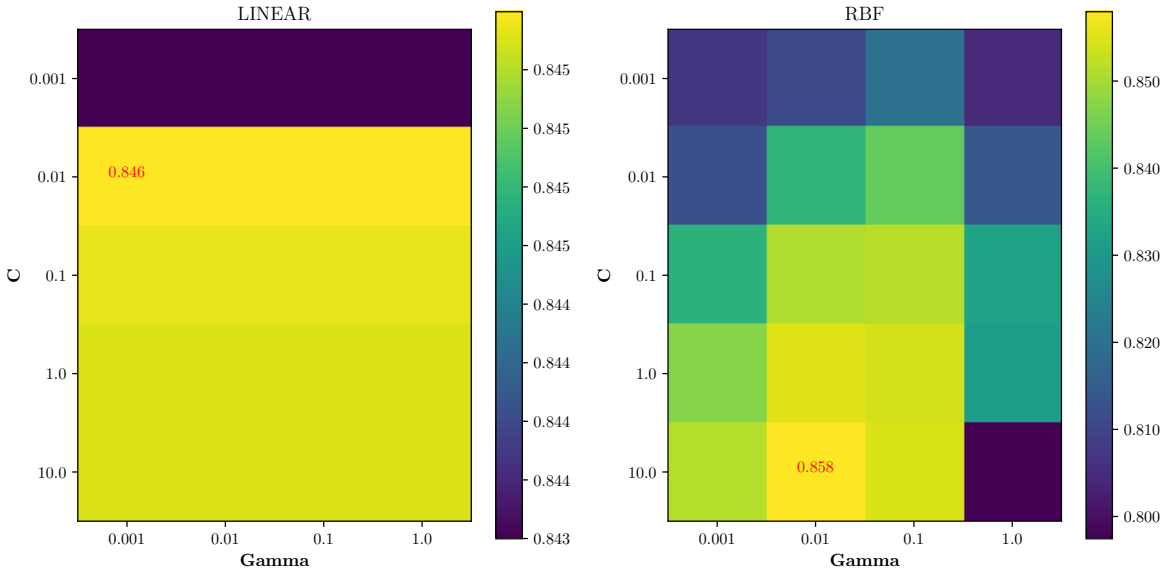


Figure 3: A visualisation of hyper-parameter optimization experiments of SVMs performed on the DN10 dataset. The left figure shows SVMs with a linear kernel. The right figure shows SVMs with an RBF kernel. In red are the AUC scores of the best performing models.

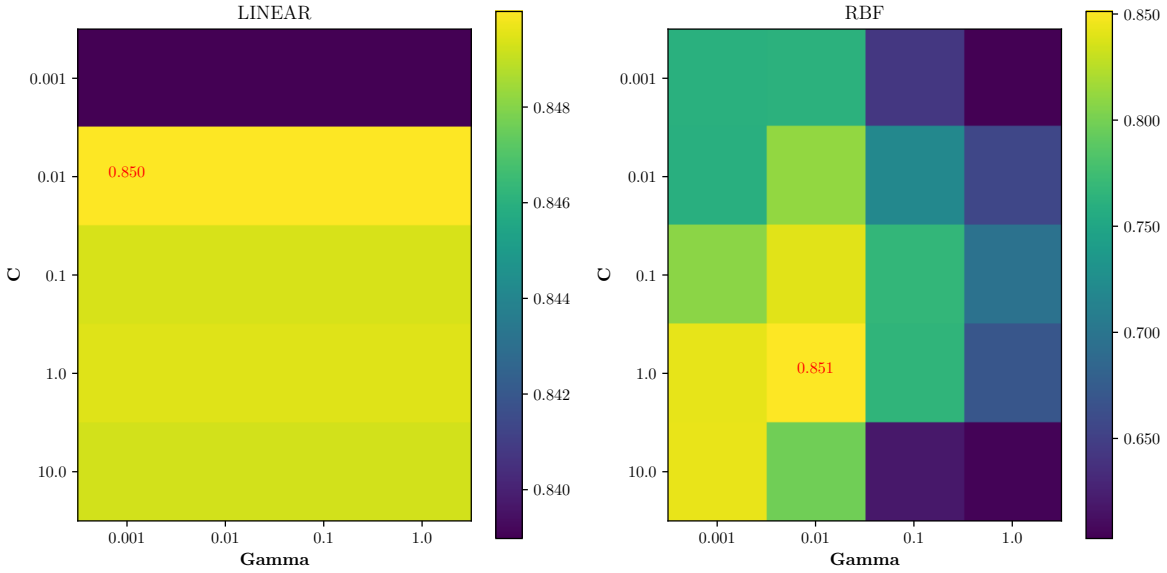


Figure 4: A visualisation of hyper-parameter optimization experiments of SVMs performed on the DN100 dataset. The left figure shows SVMs with a linear kernel. The right figure shows SVMs with an RBF kernel. In red are the AUC scores of best performing models.

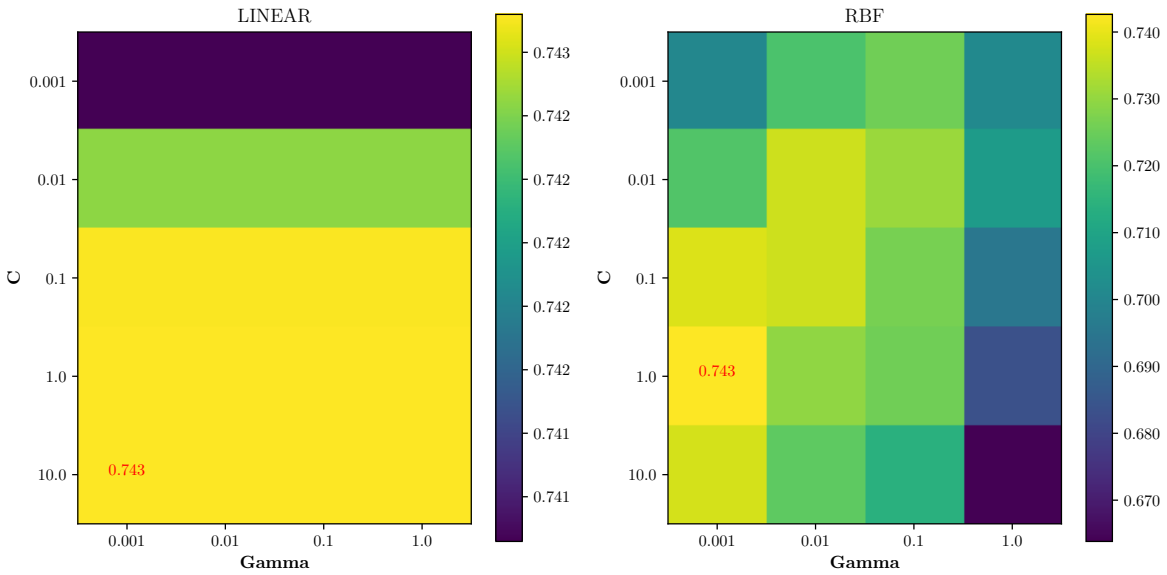


Figure 5: A visualisation of hyper-parameter optimization experiments of SVMs performed on the DN1000 dataset. The left figure shows SVMs with a linear kernel. The right figure shows SVMs with an RBF kernel. In red are the AUC scores of the best performing models.

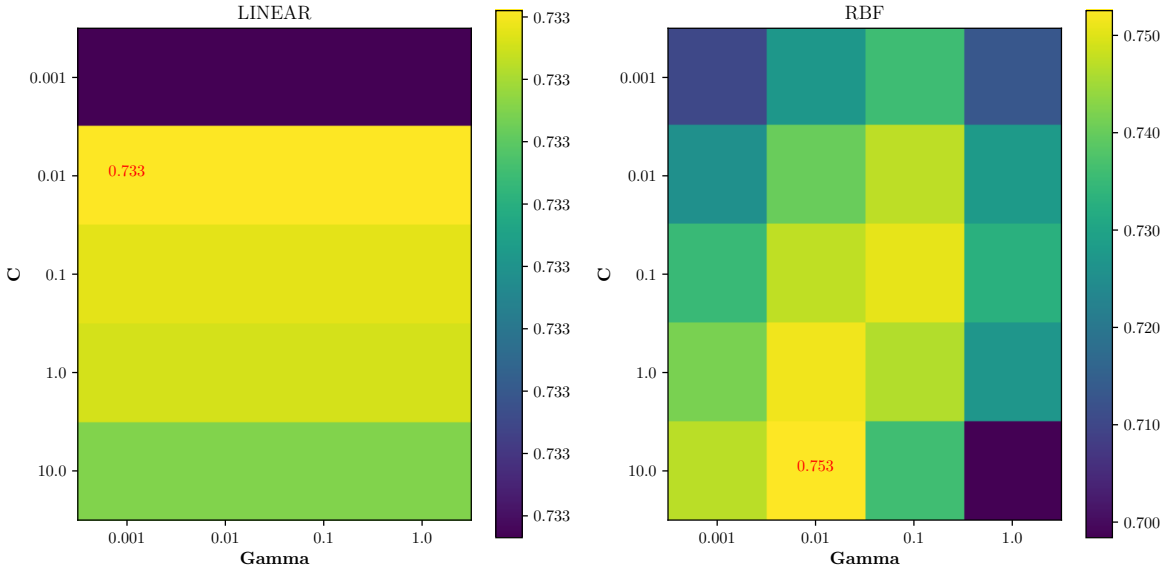


Figure 6: A visualisation of hyper-parameter optimization experiments of decision trees performed on the PAN 2020 dataset. In red is the AUC score of the best performing model.

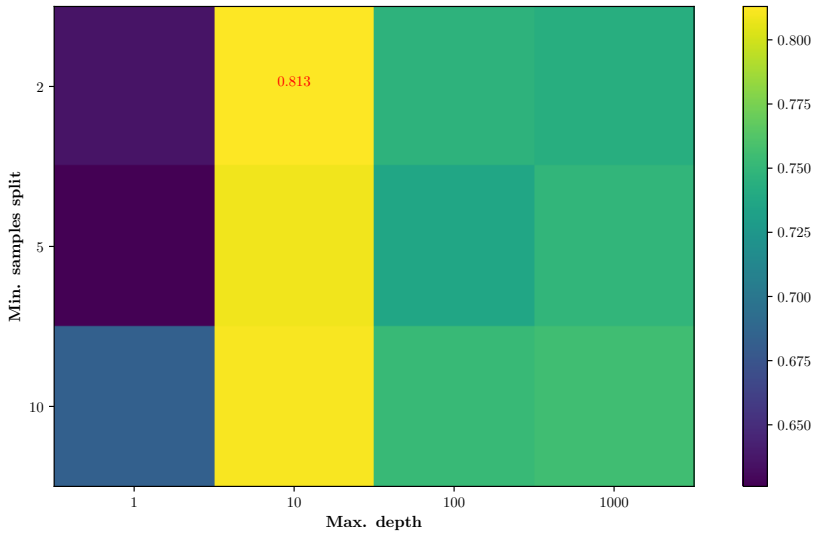


Figure 7: A visualisation of hyper-parameter optimization experiments of decision trees performed on the DN10 dataset. In red is the AUC score of the best performing model.

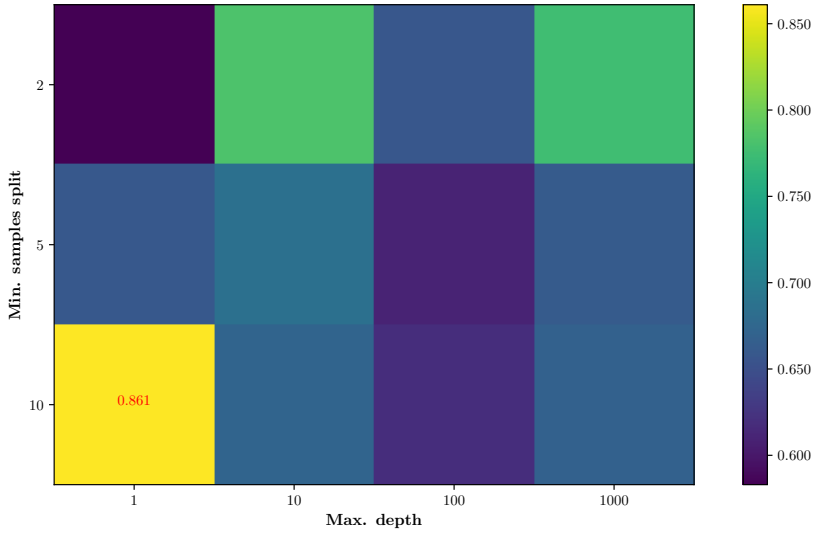


Figure 8: A visualisation of hyper-parameter optimization experiments of decision trees performed on the DN100 dataset. In red is the AUC score of the best performing model.

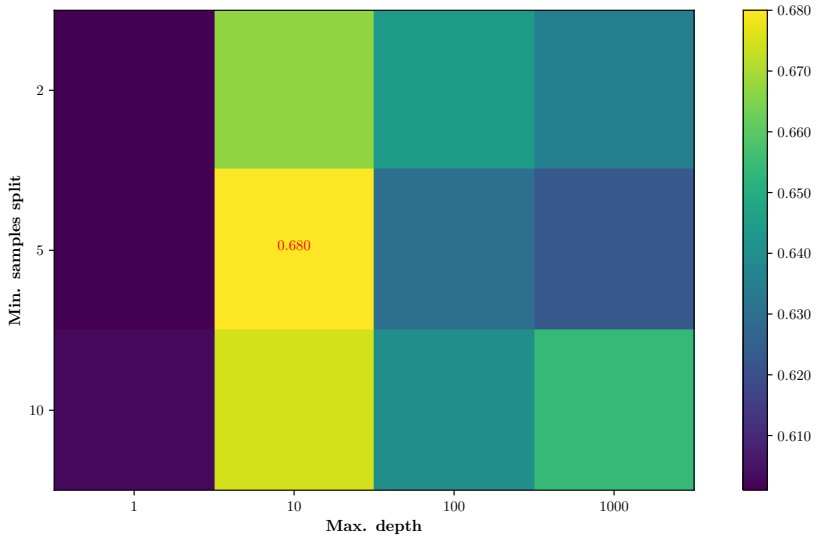


Figure 9: A visualisation of hyper-parameter optimization experiments of decision trees performed on the DN1000 dataset. In red is the AUC score of the best performing model.

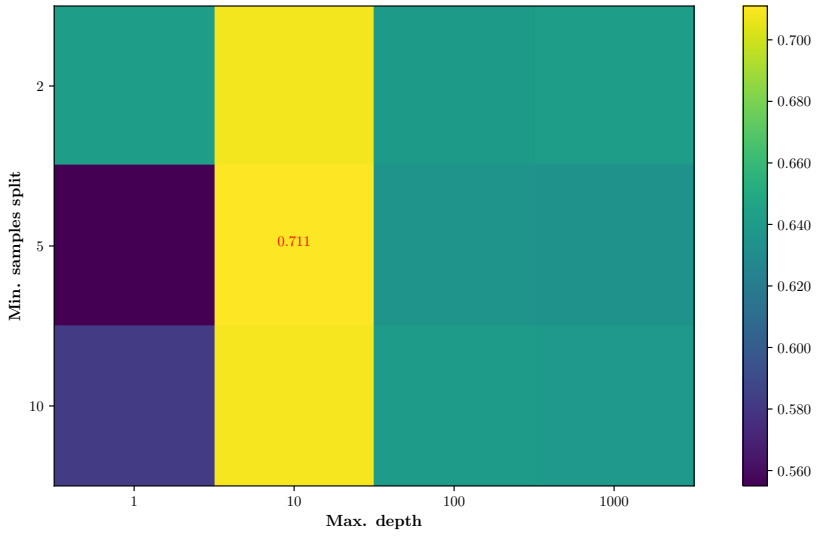


Table 1: Complete list of hyper-parameter optimization experiments of random forest classifiers performed on the PAN 2020 dataset. In **bold** is the AUC score of the best performing model.

Max depth	**Min samples split**	**# Estimators**	**AUC**
10	2	10	0.850
10	2	100	0.855
10	2	1000	0.855
10	5	10	0.848
10	5	100	0.855
10	5	1000	0.855
10	10	10	0.849
10	10	100	0.855
10	10	1000	0.855
50	2	10	0.840
50	2	100	0.856
50	2	1000	**0.859**
50	5	10	0.841
50	5	100	0.857
50	5	1000	0.859
50	10	10	0.841
50	10	100	0.857
50	10	1000	0.858
100	2	10	0.838
100	2	100	0.857
100	2	1000	0.858
100	5	10	0.840
100	5	100	0.856
100	5	1000	0.858
100	10	10	0.843
100	10	100	0.857
100	10	1000	0.858
200	2	10	0.840
200	2	100	0.856
200	2	1000	0.859
200	5	10	0.843
200	5	100	0.857
200	5	1000	0.858
200	10	10	0.840
200	10	100	0.857
200	10	1000	0.858

Table 2: Complete list of hyper-parameter optimization experiments of random forest classifiers performed on the DN10 dataset. In **bold** is the AUC score of the best performing model.

Max depth	**Min samples split**	**# Estimators**	**AUC**
10	2	10	0.768
10	2	100	0.832
10	2	1000	0.847
10	5	10	0.832
10	5	100	0.839
10	5	1000	0.844
10	10	10	**0.872**
10	10	100	0.852
10	10	1000	0.847
50	2	10	0.805
50	2	100	0.846
50	2	1000	0.842
50	5	10	0.794
50	5	100	0.839
50	5	1000	0.839
50	10	10	0.841
50	10	100	0.846
50	10	1000	0.845
100	2	10	0.812
100	2	100	0.846
100	2	1000	0.841
100	5	10	0.799
100	5	100	0.848
100	5	1000	0.842
100	10	10	0.848
100	10	100	0.839
100	10	1000	0.843
200	2	10	0.831
200	2	100	0.848
200	2	1000	0.843
200	5	10	0.809
200	5	100	0.828
200	5	1000	0.845
200	10	10	0.809
200	10	100	0.831
200	10	1000	0.842

Table 3: Complete list of hyper-parameter optimization experiments of random forest classifiers performed on the DN100 dataset. In **bold** is the AUC score of the best performing model.

Max depth	**Min samples split**	**# Estimators**	**AUC**
10	2	10	0.707
10	2	100	0.718
10	2	1000	0.720
10	5	10	0.702
10	5	100	0.717
10	5	1000	0.720
10	10	10	0.706
10	10	100	**0.723**
10	10	1000	0.721
50	2	10	0.705
50	2	100	0.716
50	2	1000	0.716
50	5	10	0.702
50	5	100	0.712
50	5	1000	0.716
50	10	10	0.718
50	10	100	0.718
50	10	1000	0.718
100	2	10	0.711
100	2	100	0.713
100	2	1000	0.716
100	5	10	0.709
100	5	100	0.718
100	5	1000	0.716
100	10	10	0.705
100	10	100	0.712
100	10	1000	0.717
200	2	10	0.707
200	2	100	0.714
200	2	1000	0.716
200	5	10	0.701
200	5	100	0.714
200	5	1000	0.716
200	10	10	0.704
200	10	100	0.714
200	10	1000	0.717

Table 4: Complete list of hyper-parameter optimization experiments of random forest classifiers performed on the DN1000 dataset. In **bold** is the AUC score of the best performing model.

Max depth	**Min samples split**	**# Estimators**	**AUC**
10	2	10	0.741
10	2	100	0.746
10	2	1000	0.746
10	5	10	0.740
10	5	100	0.745
10	5	1000	0.747
10	10	10	0.739
10	10	100	0.745
10	10	1000	0.747
50	2	10	0.727
50	2	100	0.750
50	2	1000	0.754
50	5	10	0.728
50	5	100	0.751
50	5	1000	0.753
50	10	10	0.730
50	10	100	0.750
50	10	1000	0.754
100	2	10	0.717
100	2	100	0.750
100	2	1000	0.753
100	5	10	0.725
100	5	100	0.749
100	5	1000	0.753
100	10	10	0.727
100	10	100	0.752
100	10	1000	0.754
200	2	10	0.726
200	2	100	0.752
200	2	1000	**0.754**
200	5	10	0.730
200	5	100	0.749
200	5	1000	0.753
200	10	10	0.730
200	10	100	0.750
200	10	1000	0.754

Table 5: Complete list of hyper-parameter optimization experiments of Siamese GRU models with GloVe and FastText embeddings performed on the PAN 2020 dataset. In **bold** are the AUC scores of best performing models.

			AUC	
Hidden units	**Output units**	**Dropout**	**GloVe**	**FastText**
32	32	0.1	0.834	0.704
32	32	0.2	0.822	0.726
32	32	0.4	0.822	0.776
32	64	0.1	0.823	0.597
32	64	0.2	0.821	0.724
32	64	0.4	0.824	0.796
32	128	0.1	0.809	0.731
32	128	0.2	0.820	0.719
32	128	0.4	0.809	0.699
64	32	0.1	0.848	0.677
64	32	0.2	0.829	0.701
64	32	0.4	0.841	0.665
64	64	0.1	0.834	0.793
64	64	0.2	0.835	**0.855**
64	64	0.4	0.847	0.747
64	128	0.1	0.854	0.724
64	128	0.2	0.842	0.792
64	128	0.4	0.846	0.763
128	32	0.1	0.842	0.704
128	32	0.2	**0.871**	0.730
128	32	0.4	0.852	0.783
128	64	0.1	0.857	0.683
128	64	0.2	0.859	0.731
128	64	0.4	0.865	0.795
128	128	0.1	0.864	0.718
128	128	0.2	0.870	0.726
128	128	0.4	0.853	0.762

Table 6: Complete list of hyper-parameter optimization experiments of Siamese GRU models with GloVe and FastText embeddings performed on the DN10 dataset. In **bold** are the AUC scores of best performing models.

			AUC	
Hidden units	**Output units**	**Dropout**	**GloVe**	**FastText**
32	32	0.1	0.775	0.808
32	32	0.2	0.766	0.753
32	32	0.4	0.727	0.761
32	64	0.1	0.767	0.734
32	64	0.2	0.743	0.784
32	64	0.4	0.773	0.720
32	128	0.1	**0.859**	0.831
32	128	0.2	0.766	0.850
32	128	0.4	0.726	0.819
64	32	0.1	0.814	0.745
64	32	0.2	0.673	**0.855**
64	32	0.4	0.640	0.757
64	64	0.1	0.666	0.758
64	64	0.2	0.690	0.830
64	64	0.4	0.724	0.798
64	128	0.1	0.795	0.743
64	128	0.2	0.742	0.567
64	128	0.4	0.697	0.693
128	32	0.1	0.745	0.740
128	32	0.2	0.758	0.734
128	32	0.4	0.806	0.771
128	64	0.1	0.668	0.789
128	64	0.2	0.792	0.641
128	64	0.4	0.845	0.675
128	128	0.1	0.662	0.841
128	128	0.2	0.832	0.797
128	128	0.4	0.636	0.666

Table 7: Complete list of hyper-parameter optimization experiments of Siamese GRU models with GloVe and FastText embeddings performed on the DN100 dataset. In **bold** are the AUC scores of best performing models.

			AUC	
Hidden units	**Output units**	**Dropout**	**GloVe**	**FastText**
32	32	0.1	0.620	0.641
32	32	0.2	0.692	0.602
32	32	0.4	0.684	0.643
32	64	0.1	0.637	0.597
32	64	0.2	0.692	0.647
32	64	0.4	0.695	**0.672**
32	128	0.1	0.651	0.615
32	128	0.2	0.659	0.646
32	128	0.4	0.676	0.660
64	32	0.1	0.649	0.646
64	32	0.2	0.647	0.626
64	32	0.4	0.668	0.639
64	64	0.1	0.634	0.632
64	64	0.2	0.669	0.625
64	64	0.4	0.669	0.646
64	128	0.1	0.673	0.629
64	128	0.2	0.676	0.616
64	128	0.4	0.650	0.671
128	32	0.1	0.684	0.648
128	32	0.2	0.684	0.641
128	32	0.4	0.649	0.666
128	64	0.1	0.656	0.659
128	64	0.2	0.657	0.642
128	64	0.4	0.684	0.654
128	128	0.1	0.693	0.644
128	128	0.2	**0.710**	0.656
128	128	0.4	0.682	0.650

Table 8: Complete list of hyper-parameter optimization experiments of Siamese GRU models with GloVe and FastText embeddings performed on the DN1000 dataset. In **bold** are the AUC scores of best performing models.

			AUC	
Hidden units	**Output units**	**Dropout**	**GloVe**	**FastText**
32	32	0.1	0.741	0.754
32	32	0.2	0.753	0.756
32	32	0.4	0.766	**0.764**
32	64	0.1	0.745	0.752
32	64	0.2	0.760	0.755
32	64	0.4	0.765	0.758
32	128	0.1	0.736	0.754
32	128	0.2	0.753	0.753
32	128	0.4	**0.770**	0.763
64	32	0.1	0.738	0.743
64	32	0.2	0.765	0.753
64	32	0.4	0.768	0.757
64	64	0.1	0.743	0.748
64	64	0.2	0.746	0.755
64	64	0.4	0.768	0.760
64	128	0.1	0.733	0.752
64	128	0.2	0.750	0.746
64	128	0.4	0.769	0.751
128	32	0.1	0.740	0.733
128	32	0.2	0.751	0.743
128	32	0.4	0.749	0.752
128	64	0.1	0.748	0.736
128	64	0.2	0.749	0.750
128	64	0.4	0.752	0.746
128	128	0.1	0.746	0.733
128	128	0.2	0.763	0.746
128	128	0.4	0.765	0.755