MASTER THESIS

# Towards Transfer Learning in E-Discovery:

## Finding the Optimal Classifier And Evaluating Domain Adaptation Methods

J.L. Pebesma

July 2020

STUDY PROGRAMS
MSc. Computer Science
MSc. Interaction Technology

EXAMINATION COMMITTEE
Dr. C. Seifert (chair)
Dr.ing. G. Englebienne
Prof.dr. D.K.J. Heylen
Y. van Son, MSc.

UNIVERSITY OF TWENTE.

# Executive Summary

The digitisation of modern society leaves footprints everywhere. In the legal field, it has led to the rise of e-discovery, short for electronic discovery. Once suspicion of fraud arises, possible evidence and other documents relevant to the matter have to be found in large, unstructured datasets, neither knowing what the documents look like nor how many there are. The process of manually reviewing these documents takes lots of time and can be supported using technology. This study focuses on finding the most optimal classification model for the relevant emails, using the most common data source for e-discovery, textual email data.

## Classification Model

The aim is to develop a classification model based on natural language processing and machine learning techniques. Thereby the optimal combination of a vectorisation method and a machine learning algorithm has to be found. It is currently best practise in e-discovery to use a Support Vector Machine combined with tf-idf weighting. In recent years, word embeddings have gained popularity. To see if these can outperform tf-idf vectorisation, Word2Vec embeddings using either Continuous Bag of Words or Skip-Grams and the pre-trained Flair embeddings are evaluated. As for the machine learning algorithms, Naive Bayes and the Support Vector Machine are the standards in text classification. These methods are compared to models that are increasingly being used in (imbalanced) text classification, such as ensemble methods and neural networks. The baseline models are compared to the Gradient Boosting Machine, Random Forest, Multilayer Perceptron, Long Short-Term Memory and Recurrent Convolutional Neural Network. The optimal model for the two evaluated datasets proved to be the Multilayer Perceptron with tf-idf vectorisation.

## Metadata Inclusion

After picking the optimal model, the metadata is considered. When using the subject line of the email, one dataset misclassified significantly less emails, while the performance fo the other dataset remained similar. Metadata features based on the timestamp, sender and recipients were filtered by feature selection. However, no subset of metadata features was found that improved the performance of either dataset.

## Transfer Learning

The use of transfer learning in the e-discovery domain is evaluated by an unsupervised, semi-supervised and supervised approach. Since two datasets were available, either dataset could be used as the source dataset once, resulting in two transfers. For the unsupervised approach, three methods are evaluated. The baseline is the *SrcPrior* method, during which the target data is classified using a model that is optimised and pre-trained on the source dataset. The *TgtPrior* method uses the model architecture that has been optimised for the target dataset. Two domain adaptation methods are applied, the Domain-Adversarial Neural Network (*DANN*) and the Transfer Component Analysis (*TCA*). The latter combined with either 1-Nearest Neighbors or the optimised Multilayer Perceptron. The *TgtPrior* method, the *TCA* combined with an MLP and the *DANN* all show an increase in performance for both transfers. This shows that domain adaptation is beneficial for the classification of an unlabelled set. For the semi-supervised and supervised approaches, the performance is more dependent on how balanced the source dataset is. The transfer from a more balanced set to a less balanced one shows an increase in performance, while the transfer vice versa shows either a decrease or no significant change. Two methods increase the performance for both transfers, one pre-trains a model on the source dataset using the model architecture optimised on the less imbalanced dataset, and the other one uses the prediction of a pre-trained source model on the target dataset as an extra feature in the target model. The methods using the data of both datasets at once scored in between the individual scores of the datasets.

### Explainable Artificial Intelligence

In the domain of legal technology, there is a lack of trust in artificial intelligence, due to machine learning models being considered as a black box. When using explainable Artificial Intelligence, the models could be made more explainable and transparent. This could help increase the trust in machine learning and enable the models to be controlled more easily. The 'why' for a prediction can be answered, which gains more insight into the working of the e-discovery model, enabling possible new insights and improvements. Unexpected classifications can be justified and potential bias can be eliminated, so fairness and compliance to legislation can be assured. An e-discovery model can be transformed into an explainable model using an interpretation method, such as the saliency map or (anchored) Local Interpretable Model-Agnostic Explanations.

## Acknowledgements

This thesis marks the end of seven years of being a student at the University of Twente. It has been a time I have thoroughly enjoyed and which taught me a lot, on so many levels. When I started in 2013 as a student for the bachelor Business & IT, I did not foresee that I would end up graduating two master degrees. I hope I can put my education to good use and forever continue learning.

First of all, I want to thank my supervisors Christin Seifert and Gwenn Englebienne, who have supervised me since starting the thesis. Together we found a way to effectively merge two graduation theses into one. During our meetings, you always took any new ideas I had into account and brainstormed with me to come to possible solutions. Our meetings helped me in finding a fitting scope for this thesis and aided me in looking into the right concepts, I learned a lot. I also want to express my gratitude to Dirk Heylen, who at the end agreed to join my examination committee. And last but not least, I want to thank my company supervisor, Youri van Son. I am really grateful for your supervision. You have read through many draft versions and have always provided me with extremely helpful feedback. Your punctuality and eye for detail are a gift, and I want to thank you for all the time you spent supervising me and your commitment.

Furthermore, I want to thank Johannes Scholtes for agreeing to meet me on two occasions and showing an interest in this study. Your extensive knowledge about e-discovery and its current practices were extremely helpful. I really appreciated your advice and tips.

I am extremely thankful to the Forensic Technology team at KPMG for giving me the opportunity of writing my thesis as part of their team. I am glad to have spent most of the time at your office and being part of such a great group of people, it made writing my thesis a much more enjoyable process.

Finally, I am grateful for my family and friends, who supported me throughout this process. Their encouraging and kind words were very much appreciated.

# Contents

# List of Tables

# List of Figures

## List of Abbreviations

| Abbreviation | Meaning |
|---|---|
| ADM | Automated Decision Making |
| AI | Artificial Intelligence |
| aLIME | Anchored Local Interpretable Model-Agnostic Explanation |
| AUC | Area Under Curve |
| AUPRC | Area Under Precision-Recall Curve |
| BO-GP | Bayesian Optimisation with Gaussian Process |
| CBOW | Continuous Bag of Words |
| CNN | Convolutional Neural Network |
| DANN | Domain-Adversarial Neural Network |
| E-discovery | Electronic Discovery |
| FN | False Negative |
| FP | False Positive |
| GBM | Gradient Boosting Machine |
| GloVe | Global Vectors |
| k-NN | k-Nearest Neighbours |
| L2X | Learning to Explain |
| LIME | Local Interpretable Model-agnostic Explanation |
| LRP | Layer-wise Relevance Propagation |
| LSTM | Long Short-Term Memory |
| MLP | Multilayer Perceptron |
| NB | Naive Bayes |
| NLP | Natural Language Processing |
| NN | Neural Network |
| ReLU | Rectified Linear Unit |
| RF | Random Forest |
| RCNN | Recurrent Convolutional Neural Network |
| RNN | Recurrent Neural Network |
| ROC | Receiver Operator Curve |
| SGD | Stochastic Gradient Descent |
| SHAP | SHapley Additive exPlanations |
| SVM | Support Vector Machine |
| TAR | Technology Assisted Review |
| TCA | Transfer Component Analysis |
| TF | Transfer Learning |
| TN | True Negative |
| TP | True Positive |
| tf-idf | term frequency - inverse document frequency |
| TrAdaBoost | Transfer Adaptive Boosting |
| TREC | Text REtrieval Conference |
| W2V | Word2Vec |
| XAI | Explainable Artificial Intelligence |

Table 1: List of Abbreviations

# Lists of Symbols

| Symbol | Definition |
|---|---|
| $G$ | Graph |
| $v$ | Vertex (node) |
| $\sigma_{st}$ | Shortest path between vertices $s$ and $t$ |
| $\sigma_{st}(v)$ | Shortest path between vertices $s$ and $t$ through node $v$ |
| $d(y, v)$ | Distance between vertices $y$ and $v$ |
| $deg(s)$ | Number of adjacent edges to node $s$ |

Table 2: Notations User Centrality

| Symbol | Definition |
|---|---|
| X | Feature matrix |
| Y | Label vector |
| $\mathcal{D}$ | Domain |
| $\mathcal{D}_S$ | Source domain |
| $\mathcal{D}_T$ | Target domain |
| $S$ | Source email dataset in domain $\mathcal{D}_S$ |
| $T$ | Target email dataset in domain $\mathcal{D}_T$ |
| $\mathcal{T}$ | Task |
| $\mathcal{T}_S$ | Source task |
| $\mathcal{T}_T$ | Target task |
| $X_S$, $Y_S$ | Data from source domain |
| $X_T$, $Y_T$ | Data from target domain |
| $h$ | Hypothesis, a possible classifier |
| $\mathcal{H}$ | All hypotheses (all possible classifiers) |
| $x$ | Possible outcome |
| $\mathcal{X}$ | All possible outcomes |
| $P_S(x)$ | Probability distribution of $x$ in $S$ |
| $P_T(x)$ | Probability distribution of $x$ in $T$ |
| $\epsilon_{\mathcal{D}_S}(h)$ | Error rate of classifier $h$ over $S$ |
| $\epsilon_{\mathcal{D}_T}(h)$ | Error rate of classifier $h$ over $T$ |

Table 3: Notations Transfer Learning

# 1 Introduction

Fraud is a costly affair which may concern billions of dollars [83]. Once there might be a suspicion of fraud, possible evidence should be found and the case should be evaluated. An investigation requires a lot of effort and a vast amount of time. This is mainly caused by the digitalisation of society, wherein we send an enormous number of emails and e-messages. It is even stated that we send around one million emails during our working life, which will likely only increase [99]. Finding evidence for a case of suspected fraud, can thus sometimes be compared to finding a needle in a haystack. It might even be worse since one does not know what this figurative needle looks like. This means that one has to find evidence, while neither knowing exactly which documents one is looking for, nor if any documents are present, and in case they are present, how many there are. When one has to filter certain documents out of a set, this can be done manually. However, by exploring algorithms such as predictive coding, which try to find the documents relevant to the case of fraud in a large unstructured collection of documents by themselves, the process can be speed up immensely. Without these techniques, more than 70% of the total costs of an investigation may be spent on reviewing the documents [83]. This makes the need for automated review apparent.

## 1.1 Introduction to E-Discovery

When there is a suspicion of fraud, evidence or other relevant documents need to be found in large, unstructured datasets. This process is called e-discovery, short for electronic discovery, also called Electronic Data Discovery [31]. One of the best-known cases of fraud is the Enron case, where emails served as evidence in the conviction of the CEO [68]. The company went bankrupt after an accountancy scandal and in the aftermath, the gathered emails were made public for research purposes. To find out about the fraud, all mailboxes of executive managers were reviewed. Due to the enormous number of emails, over half a million [81], doing this manually takes an excessive number of hours. To limit these hours, the process of finding documents that might contain evidence can be automated. An algorithm can be used to predict which documents might be relevant. This is called Technology Assisted Review (TAR) or predictive coding. Predictive coding can help improve the performance of the review, as well as decrease the time and workload needed. The way this technique is implemented can vary from searching with keywords to self-learning algorithms.

## 1.2 How Machine Learning Can Help

Currently, e-discovery is performed by identifying people of interest and using search terms. It is checked whether the document is sent to or by persons that are suspected of being involved in fraudulent activities, and if this document was published in the relevant time frame the fraud or its motive might have been discussed. With document retrieval and machine learning techniques getting more sophisticated and maturing, it could be an opportunity to find relevant documents by developing a model using those techniques [8]. This is why it is of great importance to find out which characteristics of a document may predict its relevance. If a model can be trained on those features, it could speed up the review process immensely. To extract features from emails, the text of the email can be vectorised [111]. On top of that, the metadata of the email can be extracted and used [64].

Machine Learning is a subfield of Artificial Intelligence (AI), in which algorithms can identify and learn patterns out of data. Therefore, it offers the possibility to create a model that can differentiate between a document that is relevant to the case and one that is not. The disadvantages of creating an e-discovery model are the small number of datasets that are publicly available for training, the small size of these datasets which makes it harder to train a stable model and the opacity of the model, for which it is not known upon which characteristics the document classification is based.

## 1.3 Transferring Knowledge Between Cases: Transfer Learning

Once a model performs well on a case, it remains a question whether this model with its characteristics just fits this case well, or if the model can be generalised over multiple cases [47]. When humans are learning a new skill, they make use of the skills they already acquired, to learn the new skill more quickly [75]. This could also be applied to an e-discovery algorithm. The knowledge gathered in one case might be useful for the training process in another case, since they may share some characteristics [84]. E-discovery cases are often imbalanced, only a small part of the emails is considered relevant. Since a model needs sufficient training data of all classes to perform well, transfer learning can be especially helpful. Deep learning models have the opportunity to learn complex patterns, but they are need a lot of data to make sense of its latent patterns [103]. E-discovery cases are considered small in that aspect, thus the ability to use more cases in the training might result in a better performing model [65]. Transfer learning could eliminate the need to manually label each new case in its entirety since the ability to use data of other cases could minimize the effort of manual labelling, which is costly [37].

## 1.4 Insights Into the Black Box of Machine Learning: Explainable Artificial Intelligence

Despite the promises of machine learning models, these models might not be trusted, especially in sensitive domains, due to their lack of transparency [90]. In the legal domain, there is a certain reluctance to use AI, due to a lack of understanding of how these models operate [24].
In addition to that, models may rely on undesirable features for classification. An example is the Amazon hiring algorithm, which appeared to negatively score all-female schools or female language on resumes. This was caused by the bias of the training set, which contained the resumes of the current IT staff that were almost all male [36]. If the predictions of an algorithm lead to decisions that can influence lives, such as assessing someone's risk of recidivism, the need for transparency and interpretability become even more apparent [2].
Due to an increasing interest in transparency, the subject of explainable Artificial Intelligence (XAI) is becoming more popular, trying to explain the black box called machine learning.

## 1.5 Problem Statement

Once fraud is suspected, an e-discovery process can be started. Currently, this process starts with the scoping the case, whereby filters and search terms are identified to limit the search results. These results are then manually reviewed on their relevance to the case, which is expensive due to the number of hours people needed for review.
Machine learning has shown potential for this task, whereby both the content and metadata of communication seem to help the classification. However, not much research has been done on the use of machine learning in e-discovery. Ideally, a classification model is trained, in a way that requires no further input, except the emails themselves. Apart from limited available research into the most optimal machine learning model, there are two other major limitations to the use of machine learning in e-discovery.
One is that the datasets are relatively small, thus some patterns may not be picked up on. Knowledge about finding relevant documents may be transferred between different fraud cases. By using the knowledge learnt from each case and applying transfer learning, the knowledge can be strengthened and findings can be solidified.
Another limitation is that the machine learning model may act as a black box, which is inconvenient for tasks that focus on creating transparency, such as fraud investigations. This limitation

is further discussed and possible solutions are considered, such as the implications of using XAI in e-discovery.

This aim of this thesis is **to optimise a classification model and create more insight into the transferability between cases while considering the implications of applying explainable AI in the legal domain**. To support this research aim, research questions are specified.

## 1.6 Research Questions

The following research questions have been formulated, to support the aim defined in the previous subsection. The performance mentioned in the questions is measured by the $F_2$-score, which is explained in Section 4.4.1.

*RQ1. To what extent do word embeddings increase the performance of an e-discovery model compared to the use of tf-idf vectors?*

*RQ2. Which machine learning model that is currently used in e-discovery performs best on the provided cases?*

*RQ3. How can the performance of the model be improved using the available metadata?*

*RQ4. To what extent is the knowledge between the provided e-discovery cases transferable?*
*SQ4.1 Is the knowledge between the cases transferable?*
*SQ4.2 To what extent can an unlabelled dataset benefit from a form of transfer learning?*
*SQ4.3 To what extent can the performance of the model be increased using a form of transfer learning?*

*RQ5. To what extent could explainable AI serve the use of machine learning in legal technology?*

## 1.7 Structure

This thesis contains a background chapter, the approach for the study, its results, the discussion and conclusion. The **background** in Chapter 2 highlights the given case and the current way of working in e-discovery, the **related work** in Chapter 3 notes the current state of the art in e-discovery, text classification, transfer learning and explainable AI. The **approach** for this study can be found in Chapter 4. The implications for the use of **explainable AI** in legal technology are discussed in Chapter 6. It is followed by the **results** in Chapter 5, of which the **discussion** is in Chapter 7, alongside the limitations and recommendations for future work. The last chapter, Chapter 8, contains the **conclusion**.

# 2 Background

The Forensic Technology team of KPMG is concerned with the discovery, investigation and prevention of fraudulent activities, corruption and cyber incidents. They can secure data from computers, phones or other digital sources, to analyse and find the facts. This department has a lot of experience with working with unstructured data, using their data-driven solution to create insights. Their way of working is transparent so that it could be used as evidence in litigation. They have multiples areas of expertise, among others;

- **Forensic Data Analysis**, which focuses on detecting fraud by analysing administrative and financial data;
- **Cyber Incident Response**, which involves responding to cyber incidents and supporting an organisation in their response; and
- **Computer Forensics & E-Discovery**, which centres on securing and investigating electronic data, such as emails.

This study is conducted for the Forensic Technology department to assist the last-mentioned area, with a focus on e-discovery. The goal in an e-discovery case is to find documents that are relevant for a fraud investigation. This could either be used in litigation or used leading up to a dismissal or a settlement. There are different sorts of investigations that the Forensic Technology team assists in, whereby the solution can benefit from e-discovery. These are, among others:

- Regulatory requests,
- Internal audit,
- Mergers and acquisitions (beforehand knowledge),
- Deduplication,
- Internal signals, which can be of:
  - Fraud, or
  - Non-compliance (internal or external rules).

In most cases, a case starts when an organisation suspects fraudulent activities. Hereby a case is defined as the entire process following this inquiry. Another inquiry of this organisation will result in a new case. During a case, a document can be a lot of things, examples being an email, a spreadsheet, an image, a PDF or a chat history such as a Slack conversation. These examples highlight the multi-modality of documents, meaning they can contain textual and visual data in all sorts of forms. Another characteristic of e-discovery is that documents are usually unstructured, which means that they are neither stored in a relational database nor have pre-defined data types. This makes it harder to search them.

## 2.1 Processing an E-Discovery Case

For each case, the data first needs to be secured. This could be an email box, containing all emails with attachments, or an entire hard drive, or one or more external disks. It is important to secure data as soon as possible, so there is no time for anyone to change or delete evidence. Once the data is safely secured, the scope of the case can be identified.
The scope of the e-discovery case states which documents should be included in the case and which should not. The most common type of e-discovery case is textual communication data [30]. More than 50% of e-discovery documents are said to be emails [31]. Therefore, the scope of this

study is on textual email data. This means that other email data, such as a PDF attachment, is ignored to limit the scope of the project.
With the scope limited to emails, the only data properties are those of an email. The exception is the BCC field of an email, which is not stored for any of the available datasets. An email consists of the following properties:

```
Date: {Timestamp email sent}
From: {Sender's address}
To: {List of receiving addresses}
Cc: {List of receiving addresses}
Subject: {Textual subject}
Content: {Email body}
Attachments: {Attached documents}
Priority: {Importance level}
```

As an example, an email out of the public Enron set is shown in Figure 1.

```
From: Kitchen Louise [mailto:Louise.Kitchen@ENRON.com]
Sent: Tuesday, January 22, 2002 22:51
To: Belden Tim <Tim.Belden@ENRON.com>;
 Milnthorp Rob <Rob.Milnthorp@ENRON.com>
Cc: Schoppe Tammie <Tammie.Schoppe@ENRON.com>
Subject: Trip to Stamford

We are scheduled to meet with Mike Hutchins and John Costas in
Stamford on Friday. Tammie is co-ordinating the trip but essentially
there is no current plan for Thursday night but we will need to be
there in time for an early start on Friday. On Friday each of us
will meet with John and Mike, have lunch, tour the facilities etc
then depart.

That's all I know - Tammie is trying to get more details.

Louise

***********
EDRM Enron Email Data Set has been produced in EML, PST and NSF
format by ZL Technologies, Inc. This Data Set is licensed under
a Creative Commons Attribution 3.0 United States License
<http://creativecommons.org/licenses/by/3.0/us/> . To provide
attribution, please cite to "ZL Technologies, Inc.
(http://www.zlti.com)."
***********
```

Figure 1: Example email

The scope of an e-discovery case is limited by specifying search terms and filters. These filters could indicate which people or email addresses should be present as either sender or receiver. Furthermore, a filter can be applied on the time stamp to filter emails sent either before or after one point in time or between two points in time. These filters are combined until a sufficient number of emails are filtered, so the manual review of these emails can be started. The emails that are returned by the filters and search terms are called 'tagged' emails. The process of finding and reviewing them is called Technology-Assisted Review (TAR).

## 2.2 Tagging the Emails

The decision on which class a document belongs to can be seen as either a binary classification problem or an information retrieval problem. Both methods have been used in e-discovery tasks [107, 81, 96]. The main difference is that classification algorithms can classify documents without any additional input, while information retrieval methods need an input query to rank the relevance of the documents. An information retrieval method always needs this point of reference, while a classification method does not. This enables an information retrieval method to work without labelled training data, as it focuses on the distance between a document and the query. A classification method determines the class a document belongs to, based upon features that are present in that document. When used unsupervised, the method usually appoints classes based upon clusters. In case labelled data is available, the classifier can learn from these labels and predict the label of new input. Since the model should be able to classify emails without any prior knowledge of the case, it is considered a classification problem.

The documents that are relevant for litigation are called 'responsive' [107]. When looking for responsive documents, the privileged ones should be excluded. Privileged documents contain (personal) information that should not be used in court. Examples hereof are an email containing private information about someone's health or an email conversation with your lawyer. Also, mails containing sensitive personal information, such as social security numbers, may either be excluded or censured. In the cases used in this study, the privileged emails have already been excluded.

The current approach is to start an investigation by defining the scope. First, it should be stated which documents should be included in the investigations, and which not. This can concern specifying document types and identifying document locations or individual documents. Then the filters that should be applied to the data are determined. An exploratory search is done with search terms, which are (combinations of) words that should occur in the document. It can be that a specific combination of words should occur in the document, or that at least one of a specified list of words should occur. This specification is called a Boolean Operator. Once the search terms are defined, additional filters can be added. The documents can be filtered based upon persons included (as owner, sender or receiver of the document) or upon time span.

The union of the documents that are returned by these filters are seen as 'tagged'. These tagged documents are then returned as the list of possibly responsive documents, which needs to be reviewed. The reviewing process of the tagged documents based on their relevance is done manually. This may take up a lot of time, depending on the number of tagged documents and how clear it is whether they are relevant. Therefore, it should be noted that the filters and search terms should neither be too broad nor too narrow. When they are too broad, a lot of documents need to be reviewed manually, leading to high costs. When they are too narrow, the responsive documents may be missed, which is considered as more negative. For every case, the queries should be formulated and the filters should be redefined. All these factors are influencing the scoping process, which can be quite time-consuming because of it.

The number of documents being responsive out of all the tagged documents is called the precision. The number of responsive documents returned, considering all responsive documents, is called the recall. During litigation, the recall is of greater importance and outweighs the precision. Rather to manually review a couple more documents than miss an important one.

## 2.3 Enron: the Best Known Example

An example case is Enron, a company that went bankrupt in 2001 [68]. Just before the company went bankrupt, the faults in their accounting were uncovered. These entailed how they artificially created profit and hid their debts. This created a huge scandal, that quickly led to the end of a company with more than 21,000 employees and the largest file for Chapter 11 bankruptcy up until that moment [62].

Back then, all the email boxes of executive managers were secured for the investigation. After the investigation, the emails were made public and after that, the emails were sold to a researcher. Since then, the Enron case has become a public dataset which supported research into e-discovery. The emails gathered for the legal investigation have been cleaned before the dataset was made public. The cleaning process entailed removing the privileged documents and duplicate emails. Furthermore, computer-generated folders were disregarded. For preparation purposes, invalid email addresses were altered to valid fictional ones, and the to-field of emails sent without recipients were filled with 'no_address@enron.com'. Alongside that, some email boxes were deleted due to personal requests. Moreover, emails containing personal data, such as birth dates, social security numbers and credit cards numbers, were excluded from the dataset. The original dataset contained 619,446 messages, while the cleaned one contains around 500,000 messages and is distributed by EDRM [45]. It is the largest available 'real' dataset for e-discovery online, creating the first standard for e-discovery studies [71]. Some of the emails in this dataset have been used in the lawsuit, enabling the evaluation of the dataset [105].

As a walk-through example of an e-discovery case, imagine that the Enron case would be redone. The Department of Justice is assigned to the case. Hereby they suspect that fraud is committed in Enron's bookkeeping. However, their official documentation does not support these claims. Thus an e-discovery case is started. When applying the current way of working, they should start with securing the data. All data that might be relevant should be secured, whereby thought is given to what may be relevant to not gather too much. The data gathered in this example will be the email boxes of all employees that are in positions that could have known about it or have anything to do with it. Alongside their email boxes, their hard disks could also be secured. However, this is left out of the example to fit the scope of this study, which is limited to textual email data.

Once the data is stored and safe, the scoping begins. During the scoping process, it is discussed what data should be investigated, and, just as important, what data should not be investigated. In this case, it is decided that the email boxes of all executive managers are used, which were around 150 people. The reason for this is that these people might have known of the fraudulent issues, or have been involved in them. Due to the large scope of the suspected fraud, concerning billions of dollars, chances that more than one employee is responsible, or that the employee(s) involved would have influential functions, are high.

After filtering the mailboxes to the executive managers, other filters are defined, considering the current number of tagged (filtered) emails is around half a million. A time filter might be applied when it is certain that a fraudulent situation had not started before a specific point in time. In this case, revenue increased by more than 750% between 1996 and 2000. This can be an indicator that the situation started after 1996, five years before the bankruptcy. For this example, the time filter is thus set to after the first of January, 1996.

When these two filters are set, the number of emails that should be investigated would still be enormous. Next, search terms are defined. The case concerns financial fraud, leading to bankruptcy after the stock price fell to less than a dollar. About six months prior, the stock price was still above 90$. This shows the stock price was probably kept high artificially. Search terms that can be applied are 'budget', 'revenue' and 'buyback'. Alongside that, the current CEO Jeffrey Skilling and his predecessor Kenneth Lay were suspected of having a large influence and sometimes doing obscure things. Therefore, their names might also hit some responsive emails. Applying this combination of filters and search terms would result in a list of emails that have been tagged as possibly responsive. Based on this list, the filters can be updated iteratively, to get to a list of tagged documents that is neither too large nor too small. For instance, when these search terms and filters give limited results, other variants for Jeffrey Skilling and Kenneth Lay can be added, Jeff Skilling and Ken Lay. Or when there are too many results, the time filter can be adjusted to start a year later. Once it is assumed that all responsive emails are in the dataset, while there are not too many documents to review, the list is finalised. All the emails contained in this list are then manually reviewed based on their relevance for the case, thus whether they can be used as evidence. This is then considered the result of the case.

# 3 Related Work

Predictive coding intends to split the tasks between humans and computers in the best way possible, which should give the best result according to Barnett and Godjevac [9]. The semantic knowledge of humans is mirrored by creating useful features, which can be created by language processing. The speed, precision and efficiency of computers are used in the form of algorithms and machine learning models.

## 3.1 Feature Engineering

To extract the semantic information from the email, the words in the emails are processed to create useful features. The most used method in e-discovery to vectorise email content is term frequency-inverse document frequency (tf-idf) weighting, which serves as a baseline for this study. The tf-idf score is calculated per individual term, showing how characterizing this term is for the document it occurs in [67]. It consists of two parts, the term frequency and the inverse document frequency. The term frequency is the number of times the term occurs. The inverse document frequency (idf) is defined as the total number of documents in the collection divided by the number of documents in which the term occurs. Since the total number of documents is often a large number, usually the log is taken of the idf. With term $i$, the total number of documents in the collection $N$ and the number of documents containing the term $n_i$, the idf of term $i$ is defined as: $idf_i = log(\frac{N}{n_i})$. For each term $i$ in document $j$, the weight $w_{i,j}$ is defined as $tf_{i,j} \times idf_i$, whereby $tf_{i,j}$ is the number of times term $i$ occurs in document $j$. This term assigns higher weights to words that occur a lot in one document, but rarely show up in other documents.
Another method of vectorisation is creating word embeddings. These have not yet been tested in e-discovery, but have been used successfully in transfer learning on email classification by Azarbonyad *et al.* [6] and Liu *et al.* [74]. This indicates that there is a possibility that e-discovery on email data might benefit from using word embeddings, due to the similarity in the sort of data. There are multiple methods of creating word embeddings. This vectorisation method creates a multidimensional space, to which words are mapped. Its concept builds on the idea that similar words, with a similar meaning and function, should be mapped close to each other in the embedded space [29]. Word embeddings sparked particular interest after the introduction of the Word2Vec (W2V) model by Mikolov *et al.* [78]. This model bases its mapping upon either the Continuous Bag of Words (CBOW) method or Skip-grams. Subsequently, more complex models such as Bidirectional Encoder Representations from Transformers (BERT), Embeddings from Language Models (ELMo), Global Vectors (GloVe) and Flair were introduced [74, 4]. When comparing the W2V models to these more advanced word embedding models, the W2V models need less data to be trained. The W2V model can be initialised and trained independently per dataset, which is why this study considers both mapping methods.
For e-discovery, it is important that local patterns in the data are discovered. If the embedded space is trained upon just the data itself instead of on a pre-trained larger set, local patterns may become clearer. An example is that the word 'bank' would have a different position when the embedded space is trained on emails of a financial institution than on emails of interior designers. If a pre-trained model is used, the position of this word should be similar, because of training data having already influenced the embedding. Since the focus is on local patterns and out of vocabulary words are not occurring, GloVe and W2V are suitable algorithms for this study. The difference between W2V and GloVe is that W2V is a predictive algorithm using a neural network, for example predicting that the word 'money' might occur based on the word 'bank' occurring, while GloVe saves the co-occurrences of each term in a large co-occurrence matrix [88]. This means that W2V only contains implicit information about the co-occurrence between terms in its representation, while GloVe uses matrix factorisation and thereby keeps this co-occurrence information. Furthermore, Camacho-Collados *et al.* [25] found out that W2V

with CBOW performed slightly better on the detection of outlying terms than GloVe when they wanted to find the terms that did not belong to a group.
Apart from these models that create a new embedded space for the dataset, some models are pre-trained on large datasets, such as Wikipedia. These models can be considered more stable, due to more training data being available. Zalando's Flair framework [4] is considered, which contains pre-trained word embeddings for the Dutch language. This aligns with the primary language of both datasets used in this study.
Different ways of vectorising the text may request different pre-processing steps. When using word embeddings, it is advised by Mikolov to usually remove punctuation and sometimes lowercase the text [77]. It is also a possibility to substitute numbers with a pre-specified token (such as '<NUM>').

### 3.1.1 Inclusion of Metadata

The content of the email is the most obvious input data. Apart from that, the metadata of an email, such as the sender, receiver and subject may also contain information [107]. Jones *et al.* [64] investigated how this metadata could be of value for the prediction. They mentioned the length of the email, number of attachments, attachments names, file types, recipients count, copy count and combined recipient count. These variables are also available in our case. After establishing the metadata that should be added, they looked into how to add the metadata for an optimal effect. Hereby an approach concatenating the metadata with its header worked best for their datasets, thus an email with subject 'lunch' would get 'SUBJECT_lunch' concatenated to its body. However, these results proved not to be significant for each dataset. This is the only study done to the use of email metadata in e-discovery, thus it has not been reproduced and there is no consensus over if and how email metadata should be used to improve the performance of an e-discovery process.
Other values could be included as metadata, apart from the part that is already available. Vinjumur [107] suggest exploiting the language with POS tags, named entity tags and the likelihood that a sentence makes sense. Also, the role of a sender or receiver in the network can be taken into account [61]. In this way, e-discovery can be applied based upon analysis of the social network.
When considering the mentioned metadata metrics, the ones that consistently improve performance should be included in the model. The process of finding out which features do so is called feature selection. There are three main methods to apply feature selection: filtering, wrapping and embedding [56]. Filter methods are based upon statistical measures to score the importance of each feature independently, based upon its influence on the performance of the model. This is a fast method that only takes into account the correlation of features to the label, which is the class they belong to. Wrapping methods consider combinations of features, working either forwards by continuing to add the best performing feature while starting from none, or backwards by eliminating features after having started with using all of them. To evaluate the feature combination, a predictive model needs to be used, which has to be defined beforehand. Embedded methods look at the contribution of the features during the process of model creation, such as the feature importance in a decision tree-based model or the coefficient in a linear SVM.

## 3.2 Machine Learning

There are many machine learning models available to classify documents. The Naive Bayes (NB) model is used by many e-discovery vendors, as well as in text classification, and is thus considered a baseline model [10]. Besides the Naive Bayes model, the Support Vector Machine (SVM) is being used in a lot of email classification systems [96]. Scholtes *et al.* mentioned the SVM as the leading technique in the field of text classification. Therefore, it is considered a baseline as well.

Because of time constraints, not all available machine learning classification models are tested. The dissertation of Vinjumur [107] is used to indicate which models are promising enough. The reason for using her dissertation is that her study, about e-discovery on privileged emails, is the one most similar to this study, out of the found studies. She indicated that Logistic Regression and Naive Bayes have a very similar performance to standard Support Vector Machines. Furthermore, she proposed using an Artificial Neural Network and two ensemble methods: the Random Forest and Gradient Boosting Machine.
Aside from that study, Yang *et al.* [111] looked into the effectiveness of popular e-discovery algorithms. They identified Linear Regression, Support Vector Machines and Gradient Boosting as well-performing algorithms. Within the domain of Artificial Neural Networks, they highlighted the Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM) and Recurrent Convolutional Neural Network (RCNN).

### 3.2.1 Neural Networks

A model that is able to use adaptive functions, is the neural network [16]. A neural network consists of one input and one output layer and at least one hidden layer in between those. Each layer contains one or more neurons. The last layer determines the predicted class for each sample. There exist multiple sorts of neural networks, each of them with a different design. When a neural network has multiple hidden layers, it is considered a deep neural network.
The Multilayer Perceptron (MLP) uses one or more fully-connected layers, in which all the neurons are connected to all nodes of the previous layer. Where one layer can learn a either a linear or a non-linear function, multiple layers can learn more complex functions [29]. Each node has an activation function, typically all nodes in a layer use the same activation function. In addition to fully-connected layers, there are other types of layers available. One of them is the convolutional layer, which outputs a weighted sum of the values in a convolutional window of a specified size. The weights are shared between the different windows in a layer. Apart from convolutional layers, there are also recurrent layers. Examples being the 'vanilla' recurrent layer and the LSTM layer. The characteristic of a recurrent layer is that it contains a state, which contains information about previously inputted data [29]. An LSTM layer is a kind of recurrent layer, which can carry information across observations [29]. This is to try to solve the vanishing gradient problem, which can occur among others when using a saturating activation function on a lot of sequence steps, causing the gradient to approximate zero. An LSTM layer can be bidirectional, which means that it actually consists of two layers, one forward and one backward layer. This increases performance, but also the computation time [29]. Another sort of layer that an RCNN can contain is a pooling layer, of which the most common types are the average pooling layer and the maximum pooling (also max-pooling) layer. The max-pooling layer only returns the maximum value of each pooling window, while the average pooling returns the average of that window.
Next to the sort of layers that are implemented, the number of neurons for each layer should be determined. The neural nets can be subject to either under or overfitting, meaning that respectively there are not enough neurons to model the complexity of the data, or the model fits too tightly around the training data [59].
Possible values for the number of nodes can be found based upon exploratory analysis and research, taking into account the work of Stathakis *et al.* [101] and Heaton *et al.* [59] on the number of nodes and layers a neural network should have.
Next to the layers and nodes, the performance is dependent on the loss function used, as well as on the optimisation function and its learning rate. The loss in a binary classification problem is usually defined by the binary cross-entropy [29]. The most commonly used optimisation functions with learning rates centred around their default value are considered [29].
Furthermore, the performance can benefit from dropout, which drops a number of randomly selected neurons per run to avoid overfitting [100]. The fraction of neurons that should be

dropped is defined by the dropout ratio.

### 3.2.2 Model Optimisation

To find the optimal model architecture, the hyperparameters of each model need to be tuned. For each model, different hyperparameters are considered. They influence the performance in such a way that there is an optimal set of hyperparameters for each specific model per dataset, which may have a significantly better performance than other sets of hyperparameters. There are multiple ways of finding the optimal hyperparameters for a model [15]. One of them is manual search, during which multiple combinations are tried, to find one that works for the case. This gives some insight into the influence of the hyperparameters. However, it could be that the optimum is not found, due to some values or combinations that have not been tried. The hyperparameters might overfit on the dataset, resulting in a set of hyperparameters that will only work on that dataset or a set found in a local optimum. An automated approach is more thorough and has an increased chance of getting an optimal solution [15]. Examples of automated approaches are:

- **Grid search**, which is considered the most basic and complete variant of hyperparameter optimisation [15]. It uses a list of parameters and their probable values as a grid. Then, all possible combinations are tried, to find the best performing one;
- **Random search**, which uses a statistical distribution per hyperparameter, instead of a discrete set [15]. The motivation for this is that not all hyperparameters are equally important for the performance of the model. Therefore, not all set-ups have to be tested, but it tries more variations of the important parameters. Furthermore, it is said to be more efficient than grid search [14].
- **Optimization algorithms**, which models a loss function while considering the model a black box [15]. These seem to have an advantage when being used for computationally expensive models. Among others, the Bayesian Optimisation with Gaussian Process (BO-GP) is an optimization algorithm, which uses the Bayesian function as an estimator for modelling the loss and thereby tries to find the optimum of that function.

## 3.3 Model Evaluation

To measure whether improvements have been made, the performance of the classification model is measured. The evaluation of a classifier is often quantified by a confusion matrix [46]. In case of a binary classification problem, it is a 2x2 matrix, which is defined in Table 4.

| | **Responsive** | **Non-responsive** |
|---|---|---|
| Responsive | True Positive (TP) | False Positive (FP) |
| Non-responsive | False Negative (FN) | True Negative (TN) |

Table 4: Confusion matrix (columns are true labels, rows are predictions)

The precision is defined by the number of TP divided by the number of TP and FP together. Thus, in the case of finding responsive documents, the precision is the proportion of responsive documents over the found documents. The recall is defined as the sum of TP divided by the number of TP and FN. Thus the recall is defined as the proportion of the responsive documents found. Aside from precision and recall, another popular metric is the accuracy, which measures the percentage of correctly classified documents. Their respective formulas are $P = \frac{TP}{TP + FP}$ for precision, $R = \frac{TP}{TP + FN}$ for recall, and $A = \frac{TP + TN}{TP + TN + FP + FN}$ for accuracy.

To quantify the performance of retrieval algorithms the $F_1$-score is often used, which is the harmonic mean of the recall and precision [60]. The $F_1$-score is an instance of the $F_\beta$-score with $\beta = 1$, which is shown in Equation 1.

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \times recall}{(\beta^2 \cdot precision) + recall} \tag{1}$$

Rijsbergen [92] stated that the $F$-measure was derived to measure the effectiveness of retrieval for users that find recall $\beta$ times as important as precision. The $F_1$-score is the harmonic mean of recall and precision, the $F_{0.5}$-score assigns a higher weight to precision than to recall and the $F_2$-score weights recall over precision.
The $F_\beta$-score, precision, recall and accuracy are considered threshold scores since they are based upon a specific count of correct classifications and incorrect ones. Another way of measuring performance is using rank scores, which are able to evaluate rankings by showing their performance over different thresholds, such as the Receiver Operator Characteristic (ROC) Curve and its Area Under Curve (AUC) [38, 20]. Instead of using precision and recall, this metric uses sensitivity and specificity. Sensitivity uses the same formula as recall, while specificity is formulated as $SPC = \frac{TN}{TN + FP}$. Similar to the ROC is the Precision Recall Curve, which uses the Area Under Precision-Recall Curve (AUPRC) [20].
When choosing the performance metric, the skew of the dataset should be considered [63]. The skew is defined by Equation 2.

$$Skew = \frac{negative\ examples}{positive\ examples} \tag{2}$$

When the data is highly skewed, the data is imbalanced. In general, e-discovery datasets are imbalanced, since only a few emails are responsive out of the entire set of emails. Jeni *et al.* [63] calculated these scores on the same datasets, to show the influence of skew. Hereby it became clear that most scores were influenced by skew, though the $F_1$-score and AUPRC were only affected in one direction (Skew $>$ 1.0) and the AUC score seemed to remain consistent. The other scores that were measured included the accuracy and the ROC curve.

## 3.4 Transfer Learning

To transfer the knowledge from one dataset to another, transfer learning can be applied. Transfer learning has not yet been applied to e-discovery. Whether the domain and task of the two models are the same affects the choice of transfer learning setting. The domain concerns the features that are used in the classification process, while the task is the classification that should be made. When both the domain and task are the same, traditional machine learning can be applied [84]. When the task is different, either inductive or unsupervised transfer learning can be applied. Hereby Pan *et al.* [84] describe unsupervised transfer learning as a situation in which no labels are needed, such as clustering or dimensionality reduction. When the task is the same, but the domain is different but related, transductive transfer learning can be applied. This corresponds to the case of e-discovery. The transductive transfer learning setting focuses on improving the target classifier, based upon the knowledge gathered from the source data, whereby the source domain is not the same as the target domain. In e-discovery, this domain gap can be seen as the different vocabulary and case subject of each dataset. The task in both e-discovery cases is still the same, namely to predict which emails are indicative of fraudulent behaviour. In the transductive setting, domain adaptation methods can be applied to decrease any gap between the source and target domain. Domain methods can be approached in the following ways [84, 110, 39]:

- **instance-based**, which utilises the source instances, though their weight is adjusted based upon their similarity to the target domain;

- **feature-representation-based** (either symmetric or asymmetric), which focuses on finding a feature representation that works for both domains. This could be asymmetric, by re-weighting the source features to resemble the target features, or symmetric, by creating a shared latent representation across the domains;
- **parameter-based**, which aims for finding shared parameters or priors between the respective models, to enhance the knowledge transfer; and
- **relational-knowledge-based**, which tightly couples the source and target domain based upon a pre-defined relationship.

Since the goal is transductive transfer learning, only instance-based and feature-representation-based methods apply out of the initial four [84]. Aside from these approaches, hybrid methods are also a possibility [110, 39].
In case the optimal model is a deep neural network, deep domain adaptation methods could also be applied. In the case of deep domain adaptation, three additional approaches are mentioned [103]:

- **mapping-based**, which can map instances of both domains into a new space, which increases the similarity between the domains;
- **network-based**, which concerns the partial re-use of the network that is trained on the source domain; and
- **adversarial-based**, which tries to find transferable representations of the source and target domain, based on adversarial technologies.

Mapping-based methods are similar to symmetric feature-representation-based methods since they both focus on creating a shared representation of the data in a new space. The network-based approach is often used in transfer learning with neural networks, by copying the first $n$ layers of the source model to the target model [112]. Then they can either be frozen or fine-tuned by the training on the target task. The remaining layers can be randomly initialised, or copied but not frozen. During the copying and freezing phase, many variations are possible. The thought behind freezing the first layers is that the first layers create general representations, while the later layers are more specific for the task. The adversarial methods are often focused on image data, such as Adversarial Discriminative Domain Adaptation (ADDA) [106].
Apart from these mentioned methods, there are many more domain adaptation methods, such as EasyAdapt (EA) [37], Structural Correspondence Learning [18], Transfer Component Analysis (TCA) [85], TrAdaBoost [35], Subspace Alignment (SA) [48] and a Domain-Adversarial Neural Network (DANN) [3].
Domain adaptation has been applied to email data before in two studies of Azarbonyad *et al.* [7, 6]. In the first study, Azarbonyad *et al.* [7] focus on computing the semantic shifts in discourse. In the second study, Azarbonyad *et al.* [6] look into domain adaptation for commitment detection in the Enron and Avocado email datasets. In this paper they focused on characterising the differences between email sets, to later perform transfer learning. The methods of transfer learning used were feature-level adaptation, sample-level adaptation and an auto-encoder to leverage both.

## 3.5 Explainable Artificial Intelligence

In current Artificial Intelligence (AI) practice, the model is often treated as some sort of black box [95]. The model has input and output, but what happens inside of it remains unknown or is not interpretable. Since recent years this lack of transparency is being addressed more, leading to a rise of interest in Explainable Artificial Intelligence (XAI) [2]. This field focuses on explaining the decision-making process of machine learning models, which is often hard to interpret.

Interpretability centres on the ‘why’ of a prediction, when the prediction can be interpreted, it is possible to answer the ‘why-question’ of why this prediction has been made [79]. Especially when a model can influence someone’s life, the need for interpretability is high [80]. A model predicting whether a person prefers books or movies based upon their tweets would need less interpretability than a model deciding whether someone is eligible for a loan. In the last case, just a prediction is not enough. Aside from the prediction, one would want to know which factors need to be changed in what way, to get a different outcome. Doshi-Velez and Kim [43] argue that the prediction itself is not enough and the need for interpretability comes from the problem formalisation being incomplete. This means that for tasks and problems only a what (prediction) is not satisfactory and a why (explanation) is needed as well. Rüping [94] defines the three goals of interpretability as a model being accurate, understandable and efficient.

Apart from interpretability, a model also needs transparency to be explainable [108]. Transparency covers simulatability, decomposability and algorithmic transparency [72]. The issue at hand for simulatability is that a person should be able to classify something manually, based upon the model. This means that the model should be understandable enough for a human to be read and understood so that people can infer the model’s decisions correctly. Decomposability entails that each part of the model has an intuitive explanation. In case of a decision tree, this means that each split contains an explanation on what it does exactly. Each part of the model should be accounted for, it should be explainable on its own. When a model has algorithmic transparency, it can be proven that they can get to a unique solution on an unseen dataset. For example, a linear classifier improves by fitting a line between those classes. The training process is transparent, for a new dataset it can be reasoned whether if the linear classifier can divide the classes. For neural networks, especially deep neural networks, it cannot be known from the start whether they will be able to solve a problem when one is presented.

There are more reasons for wanting to explain a model, such as to detect and reduce bias in a model or to find out ways the model could fail. An example is the commotion caused by the Amazon hiring process in 2018 when it became clear that their algorithm was trained for hiring men and indicated negative hiring advice when an all women’s school or more ‘female’ language was used [36]. Another example is the tank classification set, which is a well-known tale of AI gone wrong. In this example, images were classified as to whether they were showing a tank [26]. However, according to the tale, it appeared that the model was not good at classifying tanks, but rather at identifying whether the weather was sunny [28]. Supposedly, all pictures in the training set containing a tank showed sunny weather, while pictures not containing a tank showed gloomier weather. The tale zooms in on the fact that a model can learn unexpected patterns, which we may not always realize. It is an example of a situation where XAI could have helped verify the model to see if it is trained on the correct features. In turn, this could also uncover possible biases or other weaknesses in the model. Furthermore, it could lead to further insights by exploring the relations between features within the model and possible causality.

When looking into safety measures, being able to explain the model might help to find the loopholes [80]. An example would be when you are looking into self-driving cars and how they recognise bikers. The car should know where the bikers are to avoid dangerous collisions. If the model shows that it focuses on the two wheels touching the ground for classifying something as a bike, it can be questioned whether a three-wheeler would also be recognised. Once these matters become apparent, they can be fixed before any accidents occur. When the safety of the method is ensured, the models may be used in more critical cases.

Doshi-Velez and Kim [43] argue that a model is more likely to have the following traits, once it is explainable:

- **Fairness**, meaning that the prediction has no implicit nor explicit bias;
- **Privacy**, ensuring that no privileged data is used;
- **Reliability**, meaning that small changes in the input have no major influence on the prediction;
- **Causality**, ensuring that causal relationships are apparent; and
- **Trust**, checking whether humans can trust its decisions, compared to those of a black box.

To create an explainable model, interpretation methods can be used. At the moment, there seems to be no consensus on which traits or properties an explainable model needs to have, though many papers are overlapping on these traits. An important property of an explainable model is its fidelity. The fidelity of a model should describe how well the explanation approximates the prediction of the machine learning model [80].
In addition to these traits, Adadi *et al.* [2] mention the principles they think AI should adhere to, namely the A.R.T. principles:

- **Accountability**, the need to be able to explain and justify the decisions and actions of the AI model;
- **Responsibility**, considering the role of people and the capabilities of AI models; and
- **Transparency**, the need to be able to describe, inspect and reproduce the output of the mechanism.

## 3.6 Interpretation Methods

To overcome the principle of some machine learning models being considered 'a black box', interpretation methods can be used. XAI tries to either make this black box explainable or work around it [95]. Whether a model can be made explainable and the effort that it takes depends on the model, whereby a decision tree is intrinsically suited for interpretability, while a neural network is not [94]. The relation between models and their explainability is shown in Figure 2. Here it is visible that classification rules, which could also be described as if-then logic, regression algorithms and decision trees are more explainable than other models. These models can be considered white models since they are explainable by itself.
Below some of the most established techniques [27] are highlighted:

- *Salient map* [98] computes the gradient of the classification of an input entry with respect to the input vector itself. This gradient can then be used to mask the input, creating a heatmap visualisation of the importance of the features in the vector [98]. In case the input does not have a classification, a Parzen window approximation can be used [27]. A Parzen window can estimate a probability density function for a certain feature when the distribution is unknown. This enables the creation of a salient map on unlabelled data. The advantage of salient maps is that they should be able to explain the decision of any classifier [27].
- *Integrated Gradient* [102] is an explanation technique based upon the axioms Sensitivity and Implementation Variance. The sensitivity ensures that when one feature is different between two input vectors and these vectors result in different predictions, then this feature should have a non-zero attribution to the gradient. With the axiom of implementation

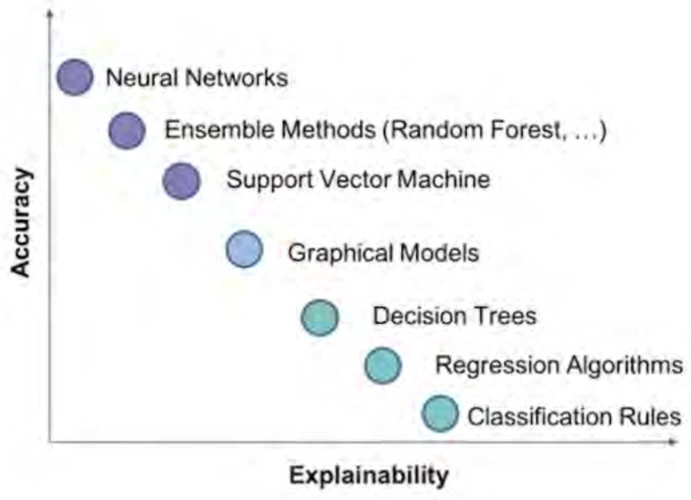


Figure 2: Explainability and accuracy of different machine learning classifiers, from Gandhi [50]

variance, two models are considered functionally equivalent when both their input vectors result in the same output vectors. In this way, the contribution of each feature to the gradient should be the same. The result of the Integrated Gradient is an additive model, created by cumulating the gradients between the inputs and their outputs.

- *Layer-wise Relevance Propagation (LRP)* [5] estimates which elements are most important for the classification decision. These can then be highlighted, to gain more insight into the decision process.

- *DeepLIFT*, [97] is a method that is specifically designed for usage in neural networks. It decomposes and back-propagates the output to trace the value of each input feature.

- *Local Interpretable Model-agnostic Explanation (LIME)* [90], is an explanation technique that creates an interpretable model, which is then trained to fit the prediction. Based on the original input, LIME generates slightly different samples. These generated samples are then inputted to the original model. The differences in predictions are used to evaluate the importance of the features. This new interpretable model should then be a good approximation locally. However, over the global model, this representation might be less accurate. Therefore, it is considered to have local fidelity. This sort of model, training a shallow model to mimic the function of a deeper one, is also called a proxy model [52].

- *Anchored LIME (aLIME)* [91] is a variation of LIME. It creates anchors, which are if-then rules that have clear coverage. They state the conditions for which the explanation is valid so that the prediction of the anchor is not influenced by changes in other features.

- *Shapley Additive Explanations (SHAP)* [76] is a method developed in game theory to explain the division of the pay-out. SHAP uses Shapley values, which are approximated by a kernel SHAP, to quantify the importance of certain features out of the input. This is a computing-intensive process, though the theory is solid and not an approximation.

- *Learning to Explain (L2X)* [27] is a technique based upon instance-wise feature selection. This means that based upon a prediction of a model, the instance-wise feature selection explores the importance of each input feature. With L2X a neural network is created, to map the feature importance as weights and return the set of the features that have the largest weights. Once the L2X model is trained, only a single pass is required. According to Chen *et al.* [27], L2X seems to outperform the other mentioned XAI methods (Salient map, LRP, DeepLIFT, SHAP and LIME) on four cases, based on clock time and their ability to identify the most influential features.

Some methods have been designed to explain specific classifiers, such as the DeepLIFT method that can explain neural networks. These are model-specific XAI methods. The benefit of a model-specific XAI method is that their performance might be increased, due to their fit to the method. A model-agnostic technique works around the model by using the prediction and the input, thus considering the model as a black box. Examples of model-agnostic techniques are LIME, SHAP and L2X.

# 4 Research Method

This chapter discusses the approach for this study, which consists of multiple stages. The result of each stage is needed to proceed to the next one. The datasets used in this study are introduced in section 4.1. Using these datasets, the following steps are taken:

1. **Vectorising the emails**. The data is first pre-processed, which is described in section 4.2. Then the emails are vectorised, which is discussed in section 4.3.

2. **Tuning and Choosing the Classification Model**. The considered classification models are discussed in section 4.4 and their evaluation metric in section 4.4.1. The approach for tuning the hyperparameters of the models is discussed in section 4.5.

3. **Evaluating of Metadata**. The metadata features are discussed in section 4.6. The methods used for selecting the metadata features that are contributing to the performance are described in section 4.6.1.

4. **Testing the Learning Bounds**. The learning bounds, which indicate if a positive knowledge transfer is expected, are explained in section 4.7.

5. **Applying Domain Adaptation Methods**. The transfer learning approaches and the domain adaptation methods are discussed in section 4.8.

## 4.1 Datasets

For this study, three datasets are available. The Enron set is used for examples since this dataset is publicly available. This dataset has been limited to the mailboxes of six employees, of whom emails could be found that were used in court. The emails used in court make up the responsive part of the emails. Two other datasets, dataset A and dataset B, are anonymized datasets provided by KPMG. The majority of these emails are Dutch. Due to their confidential nature, further specifics on these datasets cannot be shared.
Due to the extreme imbalance of the Enron set and its sort of fraud being unique, as well as that the set is English and gathered and filtered differently than most sets, this set is not included in the evaluation of the classifiers, metadata and transfer methods. For these purposes, only dataset A and dataset B are considered, since these share their collection, filtering and pre-processing methods. Future datasets are more likely to be similar to dataset A or dataset B than to the Enron dataset.
The distribution between responsive and non-responsive emails in the datasets can be seen in Table 5.

| **Dataset** | **Enron** | **Dataset A** | **Dataset B** |
|---|---|---|---|
| Responsive | 22 (1.0%) | 136 (6.8%) | 205 (21.2%) |
| Non-responsive | 22074 (99.0%) | 1863 (93.2%) | 761 (78.8%) |
| Total | 22096 (99.0%) | 1999 (100%) | 966 (100%) |

Table 5: Distribution of relevant emails per dataset

## 4.2 Pre-processing

In Algorithm 1 the pseudocode for the pre-processing phase is shown, which cleans the emails and increase their usefulness. The pre-processing steps are inspired by Scholtes *et al.* [96]. First, the footer, punctuation and multi-spaces are removed. Then the text is put to lowercase and the tokens are created. The tokens are then filtered on alphanumeric tokens that contain at least 3 characters. The numeric tokens are replaced by '<NUM>' [77].

**Algorithm 1** Pre-processing of email body

```
Input: Email body
Output: List of tokens
SET content AS email body
REMOVE(footer)
REMOVE(non-alphanumeric characters)
LOWERCASE(content)
SET tokens as TOKENIZE(content)
SET processed tokens AS new list
for all token IN tokens do
  if token IS numeric then
    REPLACE token WITH '<NUM>'
    ADD token TO processed tokens
  end if
  if token IS alphanumeric AND length(token) $> 2$ then
    ADD token TO processed tokens
  end if
end for
return processed tokens
```

## 4.3 Vectorisation

After the pre-processing of the text, the outputted tokens should be vectorised. Based upon the related literature, four vectorisation methods are evaluated:

- **Term-frequency inverse-document frequency (tf-idf)** values;
- **Word2Vec (W2V)**, based upon Continuous Bag of Words (CBOW);
- **Word2Vec (W2V)**, based upon Skip-grams (SKIP); and
- **Flair**, (Dutch) pre-trained word embeddings.

To calculate the tf-idf values, an email is considered as a document. The vocabulary of a dataset entails the unique set of words for all the emails in that dataset. This vocabulary is present during the processing of the case. However, due to the confidential matter of e-discovery datasets, we assume that the vocabulary of a case may not be saved after its processing, to protect the original emails. For each email, a vector is initialised, with its length equal to the size of the vocabulary. The tf-idf values of each token in the email are added to this vector, corresponding to their place in the vocabulary. To reduce the feature space, the features with the lowest tf-idf scores are removed from the document vectors, since they contribute the least to the classification decision. This cut-off is applied to speed up the model. The threshold for the scores to be included is calculated as *min + (max - min) * percentage*, whereby the lowest and highest values of all document features are used [96]. The percentage should be a low value not to exclude too much. In this study, a percentage of 1% is chosen. The cut-off is performed on the raw tf-idf values. After removing the features that are below the threshold, the values are normalised per vector. Thus for each email, each tf-idf value is divided by the maximum tf-idf value of that email. Since this may result in very small values, the base 10 logarithm is taken of each value [96]. In this way, the models may pick up on trends easier.
Another way of vectorising the emails is by using word embeddings. For this, two W2V models are used, one using CBOW and one using Skip-grams. Aside from the self-trained word embedding models, a pre-trained word embedding model is used, namely Flair. These embedding models are not further trained on the data since the pre-trained Dutch models are already bidirectional and trained on much larger datasets.

The models need two-dimensional input data, except for the LSTM and RCNN. These require sequential data, thus three-dimensional input data. A two-dimensional matrix consists of all emails as the first dimension, with their features as the second dimension. Sequential input requires that for each email the features are put in per observation, instead of all at once. In case of an email, this means that each token is vectorised and put in subsequently. In this way, the classifier gains information about the order of the tokens as well. This could be advantageous for the performance, however, this also increases computational load.
To enable sequential input, the vectorisation of the emails has to be redone. Hereby the model input is transformed from two-dimensional (emails $\times$ features) to three-dimensional (emails $\times$ observations $\times$ features).
For the tf-idf sequential input, the tf-idf vector should be restructured, since one tf-idf vector per observation does not fit into memory either. Instead of using the tf-idf vector, only the tf-idf value of the token itself is used as input. This means that information about the vocabulary is lost. However, by only creating one feature dimension, the sequence becomes processable. Due to limited resources, Flair embeddings are not evaluated on the sequential models, since their sequential form exceeds the available memory.
When processing the emails, each email is considered a sample and each token an observation. To process the input, all samples that are inputted at the same time need to have the same number of observations. This is a restriction of the implementation of the neural networks, for which Tensorflow v2.1.0 API for Python is used [1]. To process the sample in batches of more than one, the samples with fewer observations are padded with zeros to create input vectors of the same size. However, if the longest sequence is taken as the dimension for the observations, this can cause the input matrix to be increasingly sparse and large. Since the available resources are limited and the sequential models are computationally expensive, two methods are considered to improve the memory usage and processing time. The performance might thus be more optimal when these methods are not applied, but they are needed to make the process manageable in terms of computing power and time.
One method is to maximise the dimension of the observations to a certain percentile so that the longest sequences are cut off. Combined with the need for padding, shortening the longest sequence can have a significant effect on memory use and processing time. When choosing the percentile, one needs to keep in mind that the higher the percentile, the less information is lost and the less the training time is decreased.
Besides that, mini-batching can be applied, which is also called sequence bucketing [13, 42]. This method constructs so-called buckets of a pre-defined number of samples, which can be inputted subsequently. Each bucket is padded to the maximum sequence length within that bucket, which is almost always less than the overall maximum sequence length. Therefore, less redundant padding is inputted. To find the optimal batch size, the time for training one model can be measured for each bucket size. It should have a positive effect on the runtime when the bucket size is a power of two [55]. The considered bucket sizes are $2^n$ with $n$ ranging from one to seven, creating values between 2 and 128.

## 4.4 Classification Models

A problem similar to e-discovery is spam detection, for which the Naive Bayes algorithm can be considered as baseline [67]. For e-discovery, the current baseline model is the Support Vector Machine [96, 11].
Not much research has been found to the use of different machine learning models in e-discovery. Based on Yang [111] and Vinjumur [107], more models can be identified that may be effective for e-discovery. Both the baselines and the models mentioned by these papers are considered. Therefore, aside from these two baseline models, two ensemble methods and one neural network are considered. All these models process the data per email. Additionally, two sequential neural networks are tested, which process the email per token instead of as a whole.

The models that are evaluated, are:

- Naive Bayes (NB),
- Support Vector Machine (SVM),
- Random Forest (RF),
- Gradient Boosting Machine (GBM),
- Multilayer Perceptron (MLP),
- Long Short-Term Memory (LSTM), and
- Recurrent Convolutional Neural Network (RCNN).

When using the Scikit-learn package, the SVM can be implemented in two manners. The original SVM does not support partial or incremental learning, as it can only learn in one instance as implemented in Scikit-learn, therefore all parameters are immutable for this model. The SVM can also be implemented with a linear kernel and trained with Stochastic Gradient Descent (SGD). In Scikit-learn, this is implemented through their so-called SGD classifier, which does support partial learning. Therefore, this classifier is also included in the model evaluation, defined as SGD.

Several design choices have been made for the neural networks, thus the MLP, LSTM and RCNN. In these neural networks, the last layer uses the sigmoid activation function, since e-discovery concerns a binary classification problem.

In the LSTM model, a dropout layer is added after each LSTM layer to reduce the risk of overfitting. On top of this, a fully-connected layer is stacked.

The Recurrent Convolutional Neural Network (RCNN) for text classification is designed according to the paper of Lai *et al.* [70]. It consists of several layers, as depicted in Figure 3. It starts with a bidirectional vanilla recurrent layer, containing 3 nodes per layer. Then a convolutional layer with a kernel size of 1 is added. In this design, a maximum pooling layer is stacked on top of this. Next, a fully-connected layer is added, which uses the tanh function as the activation function. The output layer in the study of Lai *et al.* [70] uses the softmax function with as many nodes as there are unique classification labels. However, this study uses a layer with the sigmoid function with only one node, since e-discovery is a binary classification problem. The optimisation function for the RCNN is the stochastic gradient descent function.

Due to this setup containing a max-pooling layer, the only node that is learning from back-propagation is the node with the highest value per email. Due to the limited number of training samples, this could negatively impact the performance. To get more feedback from the training process, additional max-pooling layers with a smaller pooling size can be used, combined with convolutional layers. This is depicted in Figure 4 and results in more elaborate back-propagation. The pooling size of these additional max-pooling layers depends on the number of extra layers that need to be added. To ensure a symmetric design, we assume that all non-global pooling layers have the same pooling size $k$. This pooling size can then be calculated using the number $n$ of pooling layers and the sequence length $S$ per sample. A pooling layer with pooling size $k$ reduces the dimensionality of the sequence length $S$ with $S \times (\frac{1}{k})^n$. When solving this equation, this leads to a maximum pooling size $k$ of $\sqrt[n]{S}$.

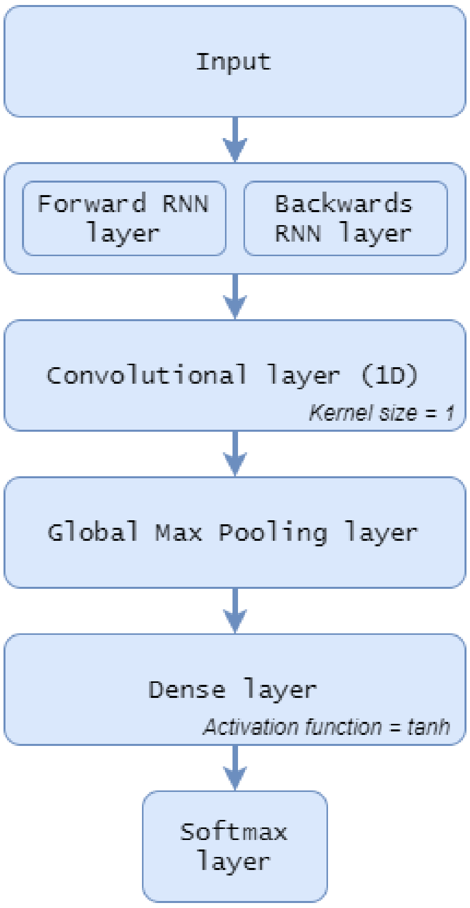


Figure 3: Layers of an RCNN, based on Lai *et al.* [70].

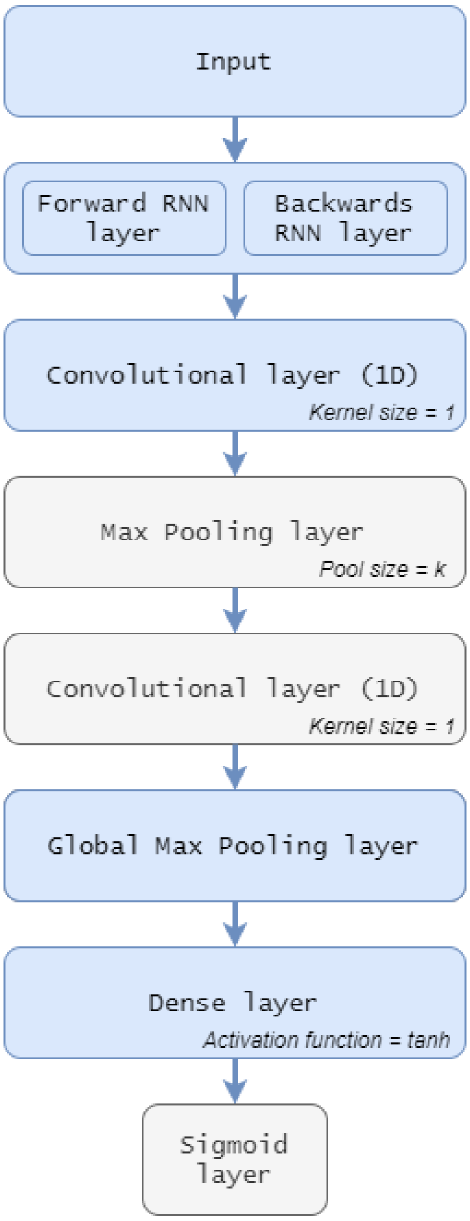


Figure 4: Layers of the implemented RCNN with an extra pooling layer. The altrications are shown in white.

### 4.4.1 Model Evaluation

The chosen evaluation metric for this study is the $F_2$-score since both recall and precision need to be considered for e-discovery, whereby more emphasis is needed on the recall. The AUPRC weights the precision and recall equally. Therefore, the $F_2$-score is used as the principal metric for evaluation, since a higher $F_2$-score indicates a more desirable model.
When training a model, it can either overfit or underfit. To ensure the performance is depicted accurately, several steps can be taken [16]. First of all, the dataset can be split into the train and test set. Hereby a random state needs to be specified, so the created split is reproducible. A common ratio between the train data and the test data is four to one so that they contain relatively 80% and 20% of the data. Due to the imbalance of the data, the folds are stratified, so that each fold contains at least a couple of responsive emails. For five-fold cross-validation, the model is initialised five times, whereby each time a different one of the five folds, thus 20% of

the data, is used as the test set. Hereby the model is using the optimal architecture as defined below. The performance of a model is measured by averaging the $F_2$-scores of the five runs.
To determine the best performing model, each model is evaluated twice, with the random states 0 and 1, which increases the stability of the performance estimate. The final performance is then defined as the average of the performance of both random states. When altering the random state, among others the initialisation of models and the split of the training and test set are influenced.
Per dataset, the mean and standard deviation of the $F_2$-scores for each model and vectorisation combination are noted. These are then compared using the Friedman test [40], which ranks the models for each dataset separately. The optimal model and vectorisation method are chosen, based upon the average ranking for each model and vectorisation method combination. With the Friedman Rank, it can be tested whether the model-vectorisation combinations perform significantly different. If the $p$-value is lower than a certain confidence interval $\alpha$, it can be assumed that their performances are significantly different.

## 4.5 Model Tuning

To tune the models, a hyperparameter optimisation method should be chosen. As primary hyperparameter optimisation method, the grid search is chosen, due to its accuracy, reproducibility and completeness. The grid per model consists of all its hyperparameters and their possible values, which can be seen in Table 6. The to be tuned parameters, together with their possible values, have been inspired by manual exploration, the book *'Deep Learning'* of Goodfellow *et al.* [54] and the study of Olsen *et al.* [82] to hyperparameters, the latter which can be consulted for explanations on their function.
For every combination of parameters, cross-validation is used to compute a mean $F_2$-score and its standard deviation. This is done with five folds, which is the default number of folds. If more folds are used, the folds can become relatively small thus decreasing the pattern matching capabilities of the model, while also being more computing-intensive. In addition to that, fewer folds might be less stable in terms of results [23]. Neural networks are trained with five epochs during the optimisation process.
There are two sorts of parameters, those that can be changed after initialisation, such as the learning rate, and those that cannot be changed due to model restrictions, such as the SVM kernel. Since all models, except for the neural networks, are implemented using their Scikit-learn implementation [86], the parameters are based upon the restriction of that package. Due to the sensitivity of the performance to the model parameters, both parameter types require a different approach.
For the mutable parameters, the optimal value for that model, vectorisation method and dataset can be used. This is the set of hyperparameters that results in the best performance for this set-up. However, since the overarching goal of this study is to transfer knowledge, the immutable parameters should be optimal for all sets. The datasets were trained, optimised and evaluated separately.
Neural networks have a larger number of variations in their hyperparameters. When performing a grid search along these, the limited available resources would be exceeded. Therefore, Bayesian Optimisation with Gaussian Processes (BO-GP) is applied [58]. When using this method, the hyperparameters that need to be tuned are considered as the dimensions that have to be optimised with regards to the $F_2$-score. This is done by modelling a loss score, whereby the classifier is searching for the minimum value of the $F_2$-score. The optimizer is called a maximum of 500 times for the MLP model, which has two-dimensional input data. For the models handling three-dimensional input data, the LSTM and the RCNN, the optimizer is called fewer times to keep the computing time feasible. These sequential models are both called 100 times. For two-dimensional data an initial batch size of one is used, while for three-dimensional data the larger batch size of 64 is used, due to time constraints, to reduce computation time further.

In case this proves not to be sufficient, a percentile cut-off and mini-batching can be applied. For this, multiple percentiles and various batch sizes can be evaluated, to see which method, or combination of methods, reduces the computing time to a considerable amount.
For the parameters that are not optimised, their default values are used, as implemented in the Tensorflow v2.1.0 API for Python [1]. This means that among others, the LSTM layers use the tanh activation function and that the weights for the fully-connected and LSTM layers are initialised using a Xavier normal initialiser [53].

| **Model** | **Parameter** |
|---|---|
| NB | Priors |
| | Var. smoothing |
| SVM | Class weights* |
| | C* |
| | Kernel* |
| | Coefficient* |
| | Gamma* ($\gamma$) |
| | Degree* |
| SGD | Class weights |
| | Learning rate ($\eta$) |
| | Initial learning rate ($\eta_0$) |
| | Exponent for inverse scaling learning rate |
| | Penalty* |
| | Loss |
| | Alpha ($\alpha$) |
| GBM | Number of estimators |
| | Maximum features |
| | Maximum depth |
| | Loss |
| | Learning rate |
| RF | Number of estimators |
| | Maximum features |
| | Maximum depth |
| | Class weights |
| | Criterion |
| *Neural networks* | Threshold |
| | Optimizer* |
| | Learning rate* |
| | Epochs* |
| | Batch size* |
| MLP | Number of input nodes |
| | Number of dense nodes |
| | Number of dense layers |
| | L2 regularizer |
| | Activation function |
| | Dropout |
| LSTM | Number of dense nodes |
| | Number of LSTM nodes |
| | Bidirectionality LSTM layer |
| | L2 regularizer |
| | Activation function dense layer |
| | Dropout |

| | |
|---|---|
| RCNN | Number of pooling layers |
| | Number of convolutional nodes |
| | Number of dense nodes |

Table 6: The hyperparameters per model, the asterisk (*) indicating model restrictions for altering after training

Once the optimised parameters are known, the learning rate, batch size and epochs are defined by visualising the training and validation loss and $F_2$-score. For this, a visual is created for each batch size and learning rate. These plots can be found in Appendix 8. The performance and loss are measured over either 200 epochs, for the MLP, or 100 epochs, for the LSTM and RCNN. The distinction in epochs is because of training time, which is much longer for sequential models. The batch sizes (1, 2, 4, 8, 16, 32 and 64) are all powers of twp, which should decrease the run time [55] Hereby the upper bound of 64 is chosen, so the batch sizes are neither too large nor too generalizing for the relatively small datasets. The learning rates that are considered are $10^{-n}$ with $n$ an integer ranging from two to six [55, 12], with an upper bound of the optimal learning rate found during the optimisation over the original five epochs. The reason for defining the learning rate, batch size and number of epochs after the initial optimization, is since using more epochs in this optimisation phase is not feasible in terms in of time. However, the batch size and learning rate are two hyperparameters that have a large, direct influence on the training process of the model. Therefore, it makes more sense to determine them at the same time.
Based upon the produced visuals, a combination of hyperparameters is chosen whereby the model is learning in the most stable manner possible, while not overfitting. This would mean that the loss goes down of both the training and validation sets, while the performance increases. The number of epochs is defined by the least loss over the validation set. This should prevent overfitting while ensuring that the model does learn. The optimal trade-off is made manually, based upon the produced plots.

## 4.6 Metadata

Aside from the text in the email body, more information is available or can be extracted per email. Both the number of attachments and the priority level of an email are metadata features that are available for each email. The sender, receiver(s), timestamp and subject have to be processed to be used as input to the model.

**Subject** Similarly to the body of the email, the subject of the email can be taken into account. This can be done by using the same method that is applied for the vectorisation of the body of the email. For the values of the terms to be correct, the tokens of both the body and subject of the mail are used when creating them. This is to ensure that the tokens of the subject are not out of vocabulary. To process all tokens at the same time, the tokens of the subject are concatenated to the tokens of the body.

**Sender and Receivers** More information might be gained by looking at the sender and receiver(s) of the email. This can be done by observing the number of receivers, or by looking at their relevant importance among all senders and receivers. When considering the importance of someone, the centrality can be used as a measure. To use the centrality, the communication network should be considered as a graph, whereby the senders and receivers are the nodes and the emails the directed links between them [61]. The centrality indicates the structural importance of a node [93], which is measured in the following aspects:

- **betweenness**, quantifying for how many nodes this node functions as a bridge for the shortest path between two other nodes;

- **closeness**, calculating the average length of the shortest path between this node and all other nodes in the graph; and
- **degree**, showing the number of vertices a node has, thus how many other people someone has sent emails to.

The centrality is measured using the igraph package [34]. Based upon a graph $G$, including all mail traffic as the edges $E$ and mail addresses involved as the vertices $V$, the metrics can be extracted. The betweenness is defined in Equation 3 based on [21], where $\sigma_{st}$ is the total number of shortest paths from node $s$ to node $t$ with $\sigma_{st}(v)$ as the number of those paths that pass through node $v$. The closeness is defined in Equation 4 as in [49], where $d(y, v)$ is the distance between vertices $y$ and $v$. The degree is defined in Equation 5 as in [21], where $deg(v)$ is the number of its adjacent edges.

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{3}$$

$$C_C(v) = \frac{1}{\sum_y d(y, v)} \tag{4}$$

$$C_D(v) = deg(v) \tag{5}$$

The values are averaged over all recipients in the 'to' and 'cc' fields together, so each email has the same vector length for the receivers. This results in six vectors, namely the three centrality values for both the sender and its receivers.

**Timestamp** Another feature that might be of importance, is the time an email is sent. This time can be split into a date and the time of day. When processing this information, it should be taken into account that the main information that is to be extracted should be a generic indicator for fraud. In this study, the following features are extracted from the email timestamp:

- Local time of sending,
- Day of the week,
- Position in timeline.

For the local time of sending, the hour and minutes are noted in one variable, by creating a continuous variable that shows the minutes as the percentage of an hour. This means that for the timestamp 11:45, the featured value would be 11.75.
The day of the week is a categorical variable, ranging from Monday to Sunday, respectively zero to six.
Lastly, the position in time relative to the investigation is added as a metadata feature. For this, two metrics are evaluated that use a relative data instead of an absolute date. By using a relative data, the created feature is more generic, so it solves problems with different cases having non-overlapping date ranges. Two methods are tested to implement a relative date. The first method is to number the days, starting on the day of the first email of the set [61, 113]. This variation is called 'numbered all days'. Sometimes systems emails are present in the dataset, which are dated on an initiation date, such as 01-01-1980. This date is an outlier compared to the rest of the emails. To circumvent the influence of these system emails on this metric, it is proposed to number only the days on which at least one email is sent, creating a sequence that ignores gaps in communication. This variation is called 'numbered mail days'. Both 'numbered all days' and 'numbered mail days' are evaluated.

Another possibility to show relative time is by sorting the timestamps and dividing them by a number, thus creating time-blocks. This number by which it is divided should be large enough to span less than a couple of months in time and small enough to represent (more than) a day. The chosen numbers are 10, 50, 100, 250 and 500. The reason for defining the time-blocks as having 1/X of the dates, instead of blocks containing 1/X of the time delta between the oldest and newest email, is again to circumvent the influence of outliers. Since the datasets only contain 966 and 1999 records, a maximum of 500 steps is chosen, since a higher number would result in the 'numbered mail day' variable. In case the datasets were larger, time-blocks containing more than 500 emails could be added as features.
Before using these metrics as features for the classifier, they are normalised by dividing all values per feature over its maximum value. This is to help the classification process [17]. When combining the previously introduced features, it results in the following list:

- Number of attachments,
- Importance,
- Time of day,
- Weekday,
- Day *(numbered all days)*,
- Day *(numbered only mail days)*,
- Timestep *(steps=10)*,
- Timestep *(steps=50)*,
- Timestep *(steps=100)*,
- Timestep *(steps=250)*,
- Timestep *(steps=500)*,
- Number of receivers,
- *Centrality sender*,
  - Betweenness,
  - Closeness,
  - Degree,
- *Centrality receiver(s)*,
  - Betweenness,
  - Closeness,
  - Degree.

### 4.6.1 Metadata Evaluation

Before using all the metadata features in the classification process, it has to be evaluated whether they have a positive effect on the classification. To explore the data, the mutual information between the metadata features and their target can be calculated. This might indicate which features may be relevant. However, two features that may not contain much information by themselves may lead to a significant improvement when combined. Since the designed metadata features may be dependent on each other for their usefulness, the selection process can benefit from an approach that takes the interdependence of features into account [56]. Therefore, the wrapping method is used. Apart from interdependences within the metadata features, they may also prove to be more helpful when combined with the token features. Therefore, the feature selection should take all interdependencies into account, while being computationally manageable.

To apply the wrapper method to both the metadata and vectorised tokens at the same time exceeds the computational and time resources. Apart from that, it is intended that the token features are kept and that only the metadata features are eliminated that are not contributing to the performance.

The approach of adding features is gradual. First, it is evaluated whether adding the subject as text to the body would improve the performance. For this, the features are generated in the same manner as with the original optimised model. However, this time the tokens of the subject are added to the tokens. Again, five-fold cross-validation is applied. Over the results of these folds, McNemar's test is applied to check whether the models perform equally [41]. If the $F_2$-score is higher when the subject is added, McNemar's test can be used to see whether the difference in misclassified emails is significant. The reason for using McNemar's test is that it is the recommended test when training and evaluating a model is expensive [89, 22]. A paired t-test cannot be applied in this case, due to the use of k-fold cross-validation. Since the dataset is not large enough to create a separate training and test set for each fold, leave-one-out cross-validation is used. Therefore, there is no independence between samples, which violates the assumption of independence of the paired t-test. Based on the significance scores, it is decided to either include the subject or not.

When adding the other metadata features, feature selection is applied to find the most optimal subset of features. The direction of the selection should depend on the number of relevant features with regards to the total number of features [73]. Since this is not known, both the forward and backward feature selection methods are applied. Thereby each search method trains the model on the original tokens and possibly the subject, depending on whether it significantly improved the performance. The performance of a model with its included subset of metadata features is defined by the $F_2$-score on the test set. Hereby the initialization seed, as well as the random state used in dividing the train and test sets, are set to zero. The train set and the test set is 20% of the total set. Due to limited time and resources, the performance is not cross-validated.

The forward selection method starts with training the optimal model using only the tokens. After noting the initial result, thus the model performance on the test set without any metadata, each next best-performing metadata feature is found by trying to concatenate each feature to the existing features and scoring the model. The feature that achieves the highest $F_2$-score is added to the dataset. Then, each remaining metadata feature is again concatenated to the dataset, now including the previous best-performing feature, again noting the performances. This continues for as many times as there are metadata features, resulting in a list containing the combinations that are performing best for each number of features included. The pseudocode for this forward feature selection can be seen in Algorithm 2.

**Algorithm 2** Pseudocode for forward feature selection

```
 Input: All tokens and all metadata features
 Output: Optimal subsets of metadata features
SET results AS new list
SET features AS new list
SET remaining features AS list of all metadata features
for all no. of metadata features do
    SET best $F_2$-score AS 0.
    for all feature IN remaining features do
        TRAIN model WITH (tokens + features + feature) RESULT $F_2$-score
        if $F_2$-score > best $F_2$-score then
            SET best $F_2$-score AS $F_2$-score
            SET best feature AS feature
        end if
    end for
    ADD best feature TO features
    REMOVE best feature FROM remaining features
    ADD (features, best $F_2$-score) TO results
end for
return results
```

The backward selection method operates similarly, though it works the other way around and starts by including all metadata features. The pseudocode for this search can be found in Algorithm 3. It iterates over all the remaining metadata features and eliminates the feature that is being left out during the run with the best $F_2$-score since that shows that its contribution to the performance in that specific case is the least. The metadata is pruned this way until only one metadata feature is left.

**Algorithm 3** Pseudocode for backward feature selection

```
 Input: All tokens and all metadata features
 Output: Optimal subsets of metadata features
SET results AS new list
SET features AS list of all metadata features
for all no. of metadata features do
    SET best $F_2$-score AS 0.
    for all feature IN features do
        TRAIN model WITH (tokens + features - feature) RESULT $F_2$-score
        if $F_2$-score > best $F_2$-score then
            SET best $F_2$-score AS $F_2$-score
            SET least feature AS feature
        end if
    end for
    REMOVE least feature FROM features
    ADD (features, best $F_2$-score) TO results
end for
return results
```

For each dataset, the forward and backward feature selection methods are applied, resulting in the four lists of metadata features that performed best during each approach. To find the optimal subset of features for both datasets, the optimal forward and backward subset for each dataset are applied to both datasets, as well as the union of the forward and backward subsets.

The union is taken both per subset and over both subsets. The resulting subset lists are thus the results for the forward search, the backward search and the union of those two searches, for each of the two datasets separately and the union of these datasets. These nine options are tested, this time with five-fold cross-validation. Hereby the subset showing the most significant increase in performance for both datasets, compared to the performance using only the tokens, is chosen to be included in the optimal model, by concatenating it to the padded tokens. When the optimal model is sequential, an extra sequence containing the metadata is added. In case no subset shows a significant improvement, no metadata features are added in the optimal model.

## 4.7 Learning Bounds

Once the optimal model has been chosen, the possible advantages of transfer learning can be considered. First, a look is taken whether the learning bounds defined by Blitzer [19] hold. The learning bounds are designed to test the similarity between datasets [19]. Below, its variables are defined:

$S$ as the source email dataset in domain $\mathcal{D}_S$
$T$ as the target email dataset in domain $\mathcal{D}_T$
$x$ as a label of the datasets
$\mathcal{X}$ as all possible labels of the datasets
$P_S(x)$ as the probability distribution of $x$ in $S$
$P_T(x)$ as the probability distribution of $x$ in $T$
$h$ as a hypothesis, a possible classifier
$\mathcal{H}$ as all hypotheses, which are all possible classifiers
$\epsilon_{\mathcal{D}_S}(h)$ as the error rate of classifier h over $S$
$\epsilon_{\mathcal{D}_T}(h)$ as the error rate of classifier h over $T$

To define the learning bound, it is important to know how similar the domains are. For this, Blitzer *et al.* [19] state that the distribution distance $d_{\mathcal{H}}$ should be used. He states this distribution distance holds the advantage of being able to work with a finite unlabelled sample, compared to the Kullback-Leibler divergence. However, since our study uses two labelled datasets, the Kullback-Leibler divergence $D_{KL}$ is used [67], as shown in Equation 6. The log is taken to base two. When Equation 8 holds, the target error is considered as within learning bounds. This indicates that there is a reasonable chance of a positive transfer from the source data to the target data since it can be assumed there is a classifier that performs well for both. The mathematical proof for the learning bounds can be found in the dissertation of Blitzer on domain adaptation [17]. The classifier $h$ is the best performing classifier as chosen utilising the Friedman Rank in the previous chapter.

$$D_{KL}(P_S||P_T) = \sum_{x\in\mathcal{X}} P_S(x) log\left(\frac{P_S(x)}{P_T(x)}\right) \tag{6}$$

$$\kappa = \min_{h\in\mathcal{H}}[\epsilon_{\mathcal{D}_S}(h) + \epsilon_{\mathcal{D}_T}(h)] \tag{7}$$

$$\epsilon_{\mathcal{D}_T}(h) \leq \epsilon_{\mathcal{D}_S}(h) + \kappa + D_{KL}(P_S||P_T) \tag{8}$$

with $\kappa$ as defined in Equation 7,
$D_{KL}$ as defined in Equation 6 and
$h$ as some classifier with low error on both $\mathcal{D}_S$ and $\mathcal{D}_T$.

## 4.8 Domain Adaptation

There are a lot of manners to transfer knowledge from one domain to the other, ranging from basic to complex. Before introducing the more complex transfer methods, some basic transfer methods are highlighted, based upon Daumé III *et al.* [37].

*TgtOnly* Only the target data is used to train and test a model, with the hyperparameters optimised on the target dataset.

*All* Both datasets are used to train and test the model. This means that the data of the two datasets is concatenated to be used to evaluate the model. The hyperparameters are either the set that causes the least loss for both sets, or a concatenation of the optimal layers, depending on whether the model is a neural network. To test this, all architectures are evaluated separately.

*Pred* The predictions of the source model on the target data are used as an extra feature [37]. For this, a model that is pre-trained on the source data predicts the class of each sample in the target set, after which this prediction is concatenated to the features of this sample. The target model is then trained and tested on the target set, including these predictions. The predictions are considered as an extra feature.

These methods require a source and target dataset, together with a classifier. Apart from these methods, domain adaptation methods can be applied. Based on related work, two categories of domain adaptation methods apply to our study, namely instance-based and feature-representation-based. In case the optimal classifier is a neural network, three additional categories are mentioned, which are mapping-based, network-based and adversarial-based.
For each of these approaches, one method is used, based upon Pan *et al.* [84], Weiss *et al.* [110] and Tan *et al.* [103]. Based on the similarity between (symmetric) feature-representation-based and mapping-based transfer methods, both focus on mapping the instances to a new shared (latent) space, only one method is considered for both of these approaches. The network-based and adversarial-based methods only apply in case a deep neural network is the most optimal classification model.

*Transfer Adaptive Boosting (TrAdaBoost)* For instance-based domain adaptation, *TrAdaBoost* is chosen [84, 110, 103]. This method adapts the sample weight of each instance, by predicting the class of each weighted instance [35]. In case it classifies a source instance incorrectly, it decreases the weight of that sample, while it increases the weights of misclassified target instances. A disadvantage of this method is that it needs the source data during the transfer process. This method is iterative, thus adapting the sample weights for a certain number of iterations. This number needs to be set beforehand. It is determined by running the model for a large number of iterations while noting the $F_2$-score per iteration. The iteration that causes the least relative loss for either target dataset is selected as the number of iterations. The implementation of the method is based on Zhuang *et al.* [114].

*Transfer Component Analysis (TCA)* For (symmetric) feature-representation-based domain adaptation, this study uses *TCA* [85, 110]. This method focuses on creating a latent subspace by identifying so-called transfer components, which are common features that are distributed in a similar way over the datasets [85]. These transfer components are learned by a kernel, which uses Maximum Mean Discrepancy (MMD) to minimize the distance between the data distributions from both domains. The code is based on its implementation in the libTLDA package [69]. Hereby the gamma ($\gamma$) is set to 0.5, conforming to the paper of Pan *et al.* [85]. The gamma is a trade-off parameter, balancing the dependence between features and labels. The linear, Laplacian and RBF kernels are considered, with either 10, 20 or 30 dimensions. The number of dimensions indicates how many transfer components are identified. Since the datasets used by Pan *et al.* [85] in their original paper contain more records than the ones used in this study, additional dimensions are added to the experiment, to test whether a larger dimension can represent the complexity and specifics of the datasets used in this study better. The added dimensions are 50, 100, 150 and 200. The transfer is evaluated with the optimised MLP models and a $k$-Nearest Neighbors ($k$-NN) model with $k = 1$ since this is the model used in the original paper. For each of these models, the optimal kernel and dimensions are determined.

*Layer-copying* The approach that is part of network-based domain adaptation, focuses on re-using pre-trained layers of a source model to improve the performance of the target model. It does not have an explicit name in literature. The most basic way of doing so would be to completely copy the layers of a model pre-trained on the source data and use it as the prior of the target training and prediction process. This means all the weights would be copied from this source model to the target model. In addition to copying all the source weights, experiments can be done by partially copying the weights of the source network, with the option of freezing them. Yosinski *et al.* [112] mention that deep layers should transition from extracting general knowledge to more domain-specific features. These layers could either be frozen, to preserve the knowledge, or only copied, so the knowledge can be adapted to better fit the target domain. In the same study, they experimented by copying from both the same and a different network, and either freezing the layers or letting them maintain their learning ability. Hereby they showed that copying the layers from a different network while letting them be fine-tuned by the new dataset performed best. This experiment was done on a computer vision experiment, so the results of applying this method to text could be different. However, since it is not the aim to fully optimise this method, but merely to test whether it can improve the classification process in e-discovery, the method is implemented according to their best-performing one. This means that first all layers are trained on the source data. When training on the target set, the weights are copied and not frozen, so they are fine-tuned by the target set. The copying of all pre-trained layers to the target model is included in the transfer method comparison, whereby the architecture of the model is either optimised for the source data, defined as *SrcPrior*, or for the target data, defined as *TgtPrior*. In both cases, the model is trained on the source data and consecutively fine-tuned and tested on the target data.

*Domain-Adaptive Neural Network (DANN)* This approach makes use of adversarial technology, to create a latent space for both domains, in the form of a neural network. This model contains two classifiers, whereby one classifies the class of the data point, while the other one classifies the data point as being from either source or target domain. The loss of the first classifier is minimised, while the loss of the latter is maximised. Therefore, the first layers create a latent mapping, after which the source and target data should be indistinguishable. Then, the class classifier should be able to determine the class of each data point, while the domain classifier should be unable to detect the domain the data point is from.

For this study, *DANN* is preferred over other adversarial-based domain adaptation methods, such as ADDA (Adversarial Discriminative Domain Adaptation) [106], which is mentioned as an example by Tan *et al.* [103]. The reason for choosing *DANN* is that its authors have provided the implementation online [51].

The method makes use of early stopping, whereby 10% of the source data set is used as the validation for the early stopping criteria. Initially, the method has a hidden layer using the sigmoid activation function 25 nodes. If the optimal model is a neural network, its optimal activation function can be evaluated as well. The number of nodes can also be adjusted to fit the datasets, whereby the range for the number of nodes in the paper is (1, 5, 12, 25, 75, 100, 150, 200). In addition to that, the learning rate of *DANN* can be adjusted, for which the authors consider values between $10^{-4}$ and $10^{-1}$. Lastly, the lambda ($\lambda$) adaptation rate can be adjusted, which weights the domain adaptation regularisation term. For this parameter, five values ranging from $10^{-2}$ to 1 and equally spaced in log-space are used. For each transfer, thus from dataset A to dataset B and vice versa, these parameter ranges are evaluated in an exhaustive search. The parameter values generating the least relative loss over both transfers, as compared with the optimal parameters, are the ones used for the final evaluation. The final evaluation is done over two states (zero and one), whereby the state only influences the formation of the early stopping validation set.

Some methods introduced in the previous section are dependent on the availability of the target labels. The availability of these labels indicates the sort of transfer method.

When the target labels are present, it is considered a case of supervised transfer learning. Thereby the aim is to improve the performance on the target dataset using the knowledge gathered on the source dataset. The baseline for supervised transfer learning is the *TgtOnly* method since this is the best achievable performance without transferring any knowledge. The supervised methods are evaluated by averaging the $F_2$-scores of five-fold cross-validation over two random states.

When only a part of the target labels is available, it is considered semi-supervised transfer learning. The same goal and methods apply as with supervised learning, however, there is less knowledge about the target domain. In case of semi-supervised transfer learning, the part that is labelled should be sufficient to have an impact while fine-tuning the model, while it should still be doable to label this seed manually. Therefore, a ratio of 0.2 is chosen. In this way it can be tested with five-fold cross-validation, making sure each data point is included in the training set once, thereby creating a stable overview of its performance and simplifying the comparison with fully supervised transfer learning. The results of these approaches can indicate how dependent the performance is on the provision of target labels. The evaluation of the semi-supervised methods is the same as with supervised methods.

In case of unsupervised transfer learning, no target labels are available. The goal is to transfer knowledge from one labelled case to an unlabelled one, to predict the labels of the target set without any prior knowledge. In current practice, this could help create training seeds for new cases, so it does not need to be coded. Since no initial model can be trained on only the target features, without any labels, the baseline is the *SrcPrior* method. The knowledge gathered of the source dataset, namely the optimal model pre-trained on the source data, can be used to provide initial labels for the target dataset, which serves as the initial performance on the target. To evaluate the impact of the model architecture, the *TgtPrior* method can be applied, which uses the model architecture optimised for the target dataset. All unsupervised methods are evaluated by taking the average $F_2$-score over two random states, except for *TCA* using $k$-NN, which does not make use of random states. If the model is deterministic without any random states, only the one $F_2$-score is reported.

Apart from whether the target labels are available, the choice of method can be dependent on whether the source data is available. Since e-discovery can be a confidential matter, in some cases the source data may not be available for use. Hereby it could be that only the vocabulary is unavailable, thus that the vectorised emails can be used, since without the vocabulary the original emails cannot be recovered. In cases where the vectorised source data is also unavailable, the pre-trained models are the only knowledge that can be transferred. Of the considered methods, *TCA*, *DANN*, *TrAdaBoost* and the *All* method need the vectorised source data. The *Pred*, *SrcPrior* and *TgtPrior* methods do not need source data, only the pre-trained models.

A schema containing the methods and baseline per supervision level can be seen in Table 7.

| *Approach* | **Unsupervised** | **Semi-supervised** | **Supervised** |
|---|---|---|---|
| Target labels available | No | Partly (20%) | Yes |
| Baseline | SrcPrior | TgtOnly | TgtOnly |
| Methods | TCA | Pred | Pred |
| | DANN | TrAdaBoost | TrAdaBoost |
| | | SrcPrior | SrcPrior |
| | | TgtPrior | TgtPrior |
| | | | All |

Table 7: Transfer approaches with their baseline and included methods

# 5 Results

## 5.1 Pre-processing

The result of the pre-processing pipeline can be seen in Table 8, which shows how the data is stored after the pre-processing. The original email comes from the Enron set and can be seen in Figure 5.

| | |
|---|---|
| **TIMESTAMP** | 22-01-2002, 22:51 |
| **FROM** | ‘louise.kitchen@enron.com’ |
| **TO** | (‘tim.belden@enron.com’, ‘To’), (‘rob.milnthorp@enron.com’, ‘To’), (‘tammie.schoppie@enron.com’, ‘Cc’) |
| **ATTACHMENTS** | 0 |
| **IMPORTANCE** | 0 |
| **SUBJECT** | ‘trip’, ‘stamford’ |
| **TOKENS** | ‘scheduled’, ‘meet’, ‘with’, ‘mike’, ‘hutchins’, ‘john’, ‘costas’, ‘stamford’, ‘friday’, ‘tammie’, ‘ordinating’, ‘trip’, ‘essentially’, ‘current’, ‘plan’, ‘thursday’, ‘night’, ‘need’, ‘time’, ‘early’, ‘start’, ‘friday’, ‘friday’, ‘meet’, ‘with’, ‘john’, ‘mike’, ‘lunch’, ‘tour’, ‘facilities’, ‘etc’, ‘depart’, ‘know’, ‘tammie’, ‘trying’, ‘get’, ‘details’, ‘louise’ |
| **RESPONSIVE** | 0 |

Table 8: Data entry of an email (original email shown in Figure 5)

The dimensionality of the input data is determined by the vectorisation method and, in case of a sequential input, the number of tokens per email. The distribution of the number of tokens per email is shown per dataset in Figure 6.
The W2V vectorisation has a dimensionality of 300. This results in an input shape of (1966, 300) and (966, 300) for respectively dataset A and dataset B. The first number hereby indicating the number of emails that each set contains. In the sequential variant, the dimensions are either (1966, 4774, 300) or (966, 2485, 300), for respectively dataset A and dataset B. After being maximised to the $90^{th}$ quartile, the dimensions are (1999, 923, 300) and (966, 665, 300) for respectively dataset A and dataset B.
The tf-idf scores have different dimensions per dataset. Dataset A has a dimensionality of (1966, 17213) for the tf-idf scores. With a threshold of 1%, the cut-value ratio is 0.0599, which reduces the dimensionality to (1996, 5119). The sequential variant has an original dimensionality of (1966, 4774, 1), which is reduced to (1966, 923, 1) when maximised to the $90^{th}$ quartile. For dataset B the original dimension of the tf-idf tokens is (966, 13225). With the same threshold, the cut-off ratio of this dataset is 0.0274, which reduces its dimensionality to (966, 5549). The sequential variant has an initial dimensionality of (966, 2485, 1), which is reduced to (966, 665, 1) when maximised to the $90^{th}$ quartile.

## 5.2 Hyperparameter optimisation

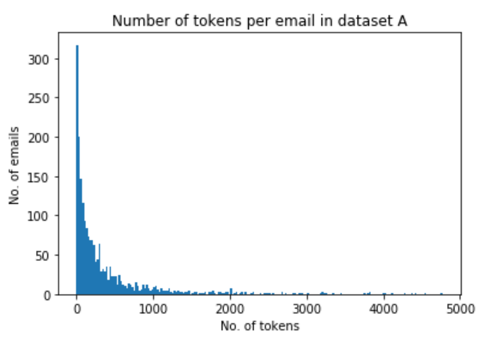
To find the most optimal model, hyperparameter optimisation is applied. The Naive Bayes, Support Vector Machine, Support Vector Machine with SGD training, Random Forest and Gradient Boosting Machine are optimized by grid searches. The values that make up the grid can be found in Table 9. For explanations of the parameters, Olson *et al.* [82] can be consulted. After testing each combination of values in five-fold cross-validation, the most optimal values are determined, which can be found in Table 10. With these optimal values, their $F_2$-score is determined by another five-fold cross-validation on two random states. The average score is taken as the final score of each model, after which all scores are ranked per dataset.

```
From: Kitchen Louise [mailto:Louise.Kitchen@ENRON.com]
Sent: Tuesday, January 22, 2002 22:51
To: Belden Tim <Tim.Belden@ENRON.com>; Milnthorp Rob <Rob.Milnthorp@ENRON.com>
Cc: Schoppe Tammie <Tammie.Schoppe@ENRON.com>
Subject: Trip to Stamford

We are scheduled to meet with Mike Hutchins and John Costas in Stamford on Friday.
Tammie is co-ordinating the trip but essentially there is no current plan for
Thursday night but we will need to be there in time for an early start on Friday.
On Friday each of us will meet with John and Mike, have lunch, tour the facilities
etc then depart.

That's all I know - Tammie is trying to get more details.

Louise

***********
EDRM Enron Email Data Set has been produced in EML, PST and NSF format by ZL
Technologies, Inc. This Data Set is licensed under a Creative Commons Attribution
3.0 United States License <http://creativecommons.org/licenses/by/3.0/us/> . To
provide attribution, please cite to "ZL Technologies, Inc. (http://www.zlti.com)."
***********
```

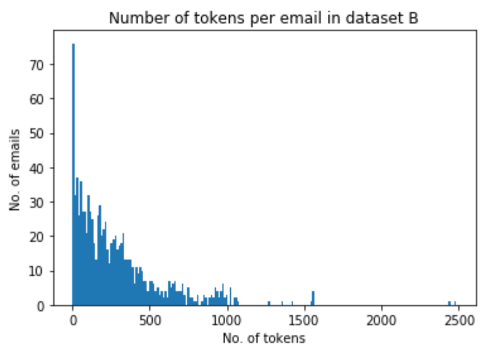
Figure 5: Example email, out of the Enron dataset

| *Parameters* | *Values* | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Naive Bayes** | | | | | | | | | |
| Priors | None | Class weights | | | | | | | |
| Var. smoothing | $10^{-11}$ | $10^{-10}$ | $10^{-9}$ | $10^{-8}$ | $10^{-7}$ | | | | |
| **Support Vector Machine** | | | | | | | | | |
| Class weights* | TRUE | FALSE | | | | | | | |
| Kernel* | RBF | Linear | Poly | Sigmoid | | | | | |
| C* | 0.1 | 1 | 10 | 100 | | | | | |
| Degree* | 1 | 2 | 3 | 4 | 5 | | | | |
| Gamma* | Auto | Scale | | | | | | | |
| Coefficient* | -0.5 | 0 | 0.5 | 1 | | | | | |
| **Support Vector Machine with Stochastic Gradient Descent Training** | | | | | | | | | |
| Class weights | TRUE | FALSE | | | | | | | |
| Loss | Hinge | Squared hinge | | | | | | | |
| Penalty* | L1 | L2 | | | | | | | |
| Alpha | $10^{-1}$ | $10^{-2}$ | $10^{-3}$ | $10^{-4}$ | $10^{-5}$ | | | | |
| Learning rate | Constant | Optimal | Invscaling | Adaptive | | | | | |
| Eta0 | $10^{-1}$ | $10^{-2}$ | $10^{-3}$ | $10^{-4}$ | $10^{-5}$ | | | | |
| Power t | 0 | 0.25 | 0.5 | 0.75 | 1 | | | | |
| **Gradient Boosting Machine** | | | | | | | | | |
| Loss | Deviance | Exponential | | | | | | | |
| Max. features | None | Squared | $log_2$ | | | | | | |
| Max. depth | None | 2 | 4 | 8 | 16 | 32 | 64 | 100 | |
| Number of estimators | 2 | 4 | 8 | 16 | 32 | 64 | 100 | 150 | 200 |
| Learning rate | 0.05 | 0.1 | 0.2 | 0.3 | | | | | |
| **Random Forest** | | | | | | | | | |
| Class weights | TRUE | FALSE | | | | | | | |
| Criterion | Gini | Entropy | | | | | | | |
| Max. features | None | Squared | $log_2$ | | | | | | |
| Max. depth | None | 2 | 4 | 8 | 16 | 32 | 64 | 100 | |
| Number of estimators | 2 | 4 | 8 | 16 | 32 | 64 | 100 | 150 | 200 |

Table 9: Hyperparameters used for grid optimisation, the asterisk(*) indicating that the value cannot be altered after training



(a) Distribution of number of tokens in dataset A



(b) Distribution of number of tokens in dataset B

Figure 6: Distribution of the number of tokens per dataset

| *Dataset* | *Vectorisation* | *Parameters* | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Naive Bayes** | | *Priors* | *Var. smoothing* | | | | | |
| Dataset A | W2V-CBOW | TRUE | $10^{-9}$ | | | | | |
| Dataset B | W2V-CBOW | FALSE | $10^{-9}$ | | | | | |
| Dataset A | W2V-SKIP | FALSE | $10^{-9}$ | | | | | |
| Dataset B | W2V-SKIP | FALSE | $10^{-9}$ | | | | | |
| Dataset A | tf-idf | FALSE | $10^{-9}$ | | | | | |
| Dataset B | tf-idf | FALSE | $10^{-9}$ | | | | | |
| Dataset A | Flair | FALSE | $10^{-9}$ | | | | | |
| Dataset B | Flair | FALSE | $10^{-9}$ | | | | | |
| **Support Vector Machine** | | *Class weights* | *Kernel* | *C* | *Degree* | *Gamma* | *Coefficient* | |
| Dataset A | W2V-CBOW | TRUE | Poly | 100 | 5 | Scale | 0.5 | |
| Dataset B | W2V-CBOW | TRUE | Poly | 100 | 5 | Scale | 0.5 | |
| Dataset A | W2V-SKIP | TRUE | Poly | 10 | 5 | Scale | 1 | |
| Dataset B | W2V-SKIP | TRUE | Poly | 10 | 5 | Scale | 1 | |
| Dataset A | tf-idf | TRUE | Sigmoid | 100 | 3 | Scale | 1 | |
| Dataset B | tf-idf | TRUE | Sigmoid | 100 | 3 | Scale | 1 | |
| Dataset A | Flair | TRUE | Poly | 10 | 5 | Scale | 0 | |
| Dataset B | Flair | TRUE | Poly | 10 | 5 | Scale | 0 | |
| **SVM with SGD Training** | | *Class weights* | *Loss* | *Penalty* | *Alpha* | *Learning rate* | *Eta0* | *Power t* |
| Dataset A | W2V-CBOW | TRUE | Hinge | L2 | $10^{-5}$ | Adaptive | $10^{-2}$ | 0 |
| Dataset B | W2V-CBOW | TRUE | Squared hinge | L2 | $10^{-5}$ | Adaptive | $10^{-3}$ | 0 |
| Dataset A | W2V-SKIP | TRUE | Hinge | L1 | $10^{-4}$ | Adaptive | $10^{-1}$ | 0 |
| Dataset B | W2V-SKIP | TRUE | Squared hinge | L1 | $10^{-2}$ | Adaptive | $10^{-3}$ | 0 |
| Dataset A | tf-idf | TRUE | Squared hinge | L1 | $10^{-3}$ | Adaptive | $10^{-1}$ | 0 |
| Dataset B | tf-idf | TRUE | Hinge | L1 | $10^{-5}$ | Invscaling | $10^{-1}$ | 0 |
| Dataset A | Flair | TRUE | Squared hinge | L2 | $10^{-2}$ | Adaptive | $10^{-2}$ | 0 |
| Dataset B | Flair | TRUE | Hinge | L2 | $10^{-3}$ | Adaptive | $10^{-2}$ | 0 |
| **Gradient Boosting Machine** | | *Loss* | *Max depth* | *Max features* | *Number of estimators* | *Learning rate* | | |
| Dataset A | W2V-CBOW | Deviance | None | Squared | 4 | 0.30 | | |
| Dataset B | W2V-CBOW | Deviance | None | None | 100 | 0.10 | | |
| Dataset A | W2V-SKIP | Deviance | 16 | $log_2$ | 2 | 0.20 | | |
| Dataset B | W2V-SKIP | Deviance | 4 | Squared | 100 | 0.30 | | |
| Dataset A | tf-idf | Deviance | 16 | None | 4 | 0.20 | | |
| Dataset B | tf-idf | Exponential | 16 | None | 200 | 0.05 | | |
| Dataset A | Flair | Deviance | 16 | None | 4 | 0.30 | | |
| Dataset B | Flair | Deviance | 2 | None | 200 | 0.20 | | |
| **Random Forest** | | *Class weights* | *Max depth* | *Max features* | *Number of estimators* | *Criterion* | | |
| Dataset A | W2V-CBOW | TRUE | 4 | Squared | 200 | Entropy | | |
| Dataset B | W2V-CBOW | TRUE | 2 | $log_2$ | 32 | Gini | | |
| Dataset A | W2V-SKIP | TRUE | 2 | $log_2$ | 32 | Entropy | | |
| Dataset B | W2V-SKIP | TRUE | 2 | $log_2$ | 200 | Gini | | |
| Dataset A | tf-idf | TRUE | 2 | None | 16 | Gini | | |
| Dataset B | tf-idf | TRUE | 4 | $log_2$ | 150 | Gini | | |
| Dataset A | Flair | TRUE | 2 | None | 200 | Entropy | | |
| Dataset B | Flair | TRUE | 2 | Squared | 64 | Entropy | | |

Table 10: Optimal hyperparameters found using grid search per dataset and vectorisation method

The categories and ranges that are used for the BO-GP optimisation process for the parameters of the models are noted in Table 11.

| *Parameters* | *Values* | | | | |
|---|---|---|---|---|---|
| **Multilayer Perceptron** | *low* | *high* | | | |
| Number of layers | 1 | 5 | | | |
| Number of input nodes | 1 | 200 | | | |
| Number of dense nodes | 1 | 200 | | | |
| L2 regularizer | $10^{-10}$ | $10^{-1}$ | | | |
| Dropout ratio | 0.0 | 0.8 | | | |
| Threshold | 0.1 | 0.9 | | | |
| Learning rate* | $10^{-6}$ | $10^{-2}$ | ReLU | | |
| Optimizer* | Adam | Adadelta | Adagrad | SGD | RMSprop |
| Activation function | Tanh | Sigmoid | ReLU | | |
| **LSTM** | *low* | *high* | | | |
| Number of LSTM nodes | 1 | 100 | | | |
| Number of dense nodes | 1 | 200 | | | |
| L2 regularizer | $10^{-10}$ | $10^{-1}$ | | | |
| Dropout ratio | 0.0 | 0.8 | | | |
| Threshold | 0.1 | 0.9 | | | |
| Learning rate* | $10^{-4}$ | $10^{-2}$ | | | |
| Optimizer* | Adam | Adadelta | Adagrad | SGD | RMSprop |
| Activation function | Tanh | Sigmoid | ReLU | | |
| **RCNN** | *low* | *high* | | | |
| Number of convolutional layers | 0 | 10 | | | |
| Number of convolutional nodes | 1 | 100 | | | |
| Number of dense nodes | 1 | 200 | | | |
| Learning rate* | $10^{-4}$ | $10^{-2}$ | | | |
| Threshold | 0.1 | 0.9 | | | |

Table 11: Hyperparameters used for Bayesion Optimisation with Gaussian Processes

Due to limited resources, the number of parameters and their ranges for the LSTM models are narrowed, such as the number of (LSTM) layers. Since the optimisation still exceeds computation resources, sequence bucketing is used in combination with a cut-off to further speed up the model. The cut-off is tried on both the $90^{th}$ and the $99^{th}$ quartiles of the longest sequence length. The bucket size is determined by timing the training of a simple LSTM model for five epochs with different bucket sizes, of which the fastest is chosen. This simple model consists of an input and an output layer with one mono-directional LSTM layer in between, this LSTM layer has 64 nodes and uses the tanh activation function. The model is compiled with the Adam optimizer. The training durations are shown in Figure 7, which shows that a bucket size of 64 is fastest.

During the hyperparameter optimisation, the initial batch size used for the MLP is 1. The LSTM implements buckets with a batch size of 64 and the RCNN works with a batch size of 64. The RCNN does not use bucketing, as it runs faster than the LSTM and uses the sequence length to determine the nodes in its pooling layers, which therefore should be the same in each sequence. Both sequential methods make use of a cut-off at the $90^{th}$ percentile.

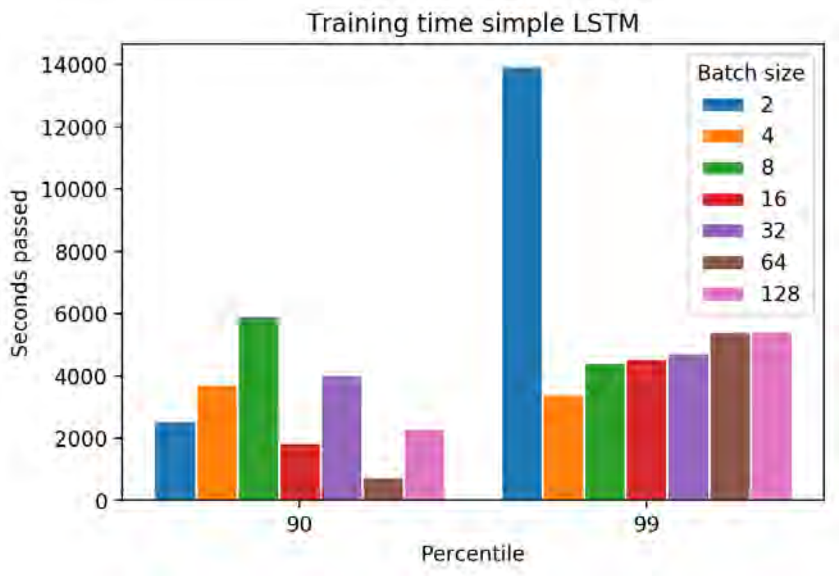


Figure 7: Training times in seconds for a simple LSTM model on five epochs shown for the cut-off on the 90th and 99th percentile

The optimised values for the MLP, LSTM and RCNN can be seen in respectively Table 12, 13, 14. The optimal learning rate, batch size and epochs are explored using these optimised hyperparameters. They are defined by visualising the training and validation loss and $F_2$-score. For this, a visual is created for each batch size (2, 4, 8, 16, 32 and 64) and learning rate, over either 500 epochs, for the MLP, or 200 epochs, for the LSTM and RCNN. The range of learning rates that is tried is $10^{-n}$ with $n$ an integer between six and two [54], with the learning rate optimised over the original 5 epochs as the upper bound.

The visuals showing the chosen curves are depicted in Appendix 8, in Figure 13, 14 and 15 for respectively the MLP, LSTM and RCNN. The performance of the model is determined by taking the average of five-fold cross-validation on the same two random states as with the grid searches. The MLP models show an overall better performance using the newly deduced learning rate than with the learning rate that was optimised with BO-GP over 5 epochs.

| **Dataset** | *Dataset A* | *Dataset B* | *Dataset A* | *Dataset B* | *Dataset A* | *Dataset B* | *Dataset A* | *Dataset B* |
|---|---|---|---|---|---|---|---|---|
| **Vectorisation** | *W2V-CBOW* | *W2V-CBOW* | *W2V-SKIP* | *W2V-SKIP* | *tf-idf* | *tf-idf* | *Flair* | *Flair* |
| **Parameters** | | | | | | | | |
| Number of input nodes | 200 | 200 | 200 | 103 | 136 | 194 | 199 | 1 |
| Number of dense nodes | 111 | 200 | 82 | 31 | 49 | 200 | 32 | 122 |
| Number of dense layers | 1 | 1 | 3 | 2 | 2 | 3 | 1 | 5 |
| Dropout ratio | 0 | 0.1 | 0.2 | 0.4 | 0.3 | 0.4 | 0.2 | 0 |
| L2-regularizer | $10^{-10}$ | $10^{-10}$ | $10^{-7}$ | $10^{-7}$ | $2 \times 10^{-5}$ | $10^{-8}$ | $5 \times 10^{-9}$ | $10^{-10}$ |
| Threshold | 0.1 | 0.4 | 0.6 | 0.5 | 0.1 | 0.1 | 0.1 | 0.3 |
| Activation function | ReLU | ReLU | ReLU | ReLU | Tanh | ReLU | Tanh | ReLU |
| Optimizer | Adam | Adam | Adam | Adam | Adam | Adam | Adam | Adam |
| Learning rate | $10^{-5}$ | $10^{-5}$ | $10^{-5}$ | $10^{-5}$ | $10^{-5}$ | $10^{-5}$ | $10^{-5}$ | $10^{-5}$ |
| Batch size | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Epochs | 50 | 50 | 100 | 100 | 25 | 25 | 75 | 75 |

Table 12: Hyperparameters Multilayer Perceptrons

| **Dataset** | *Dataset A* | *Dataset B* | *Dataset A* | *Dataset B* | *Dataset A* | *Dataset B* |
|---|---|---|---|---|---|---|
| **Vectorisation** | *W2V-CBOW* | *W2V-CBOW* | *W2V-SKIP* | *W2V-SKIP* | *tf-idf* | *tf-idf* |
| **Parameters** | | | | | | |
| Number of LSTM nodes | 76 | 4 | 44 | 53 | 46 | 48 |
| Number of dense nodes | 166 | 146 | 155 | 58 | 41 | 39 |
| Bidirectional | FALSE | FALSE | TRUE | TRUE | TRUE | TRUE |
| Dropout ratio | 0.3 | 0.01 | 0.4 | 0.6 | 0.39 | 0.33 |
| L2-regularizer | $1.9 \times 10^{-9}$ | $4.3 \times 10^{-8}$ | $2.2 \times 10^{-5}$ | $3.3 \times 10^{-7}$ | $1.6 \times 10^{-10}$ | $8.8 \times 10^{-7}$ |
| Threshold | 0.2 | 0.16 | 0.31 | 0.14 | 0.24 | 0.45 |
| Activation function | ReLU | Sigmoid | Tanh | ReLU | Tanh | Tanh |
| Optimizer | RMSprop | RMSprop | Adam | Adam | SGD | SGD |
| Learning rate | $10^{-4}$ | $10^{-4}$ | $10^{-2}$ | $10^{-2}$ | $10^{-2}$ | $10^{-2}$ |
| Batch size | 8 | 8 | 16 | 16 | 4 | 4 |
| Epochs | 75 | 75 | 20 | 20 | 30 | 30 |

Table 13: Hyperparameters LSTM

| **Dataset** | *Dataset A* | *Dataset B* | *Dataset A* | *Dataset B* | *Dataset A* | *Dataset B* |
|---|---|---|---|---|---|---|
| **Vectorisation** | *W2V-CBOW* | *W2V-CBOW* | *W2V-SKIP* | *W2V-SKIP* | *tf-idf* | *tf-idf* |
| **Parameters** | | | | | | |
| Number of dense nodes | 195 | 42 | 1 | 57 | 200 | 100 |
| Number of convolutional nodes | 58 | 9 | 1 | 65 | 50 | 1 |
| Number of convolutional layers | 3 | 3 | 10 | 7 | 10 | 3 |
| Threshold | 0.44 | 0.49 | 0.23 | 0.5 | 0.28 | 0.5 |
| Learning rate | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-2}$ | $10^{-2}$ |
| Batch size | 1 | 1 | 1 | 1 | 4 | 4 |
| Epochs | 40 | 40 | 50 | 50 | 25 | 25 |

Table 14: Hyperparameters RCNN

## 5.3 Classifier Performance

In Table 15 the $F_2$-scores can be found per model and vectorisation method for each dataset. The scores noted are the average of five-fold cross-validation on two random states, along with their standard deviation. When looking at the $F_2$-scores, it shows that the first dataset is more imbalanced, which results in lower $F_2$-scores on average. Per dataset, the averaged $F_2$-scores are ranked, with the possibility of ties and using half ranks. The last column in the table notes the average rank over both datasets. Based on the ranks the Friedman Rank calculation is made.

| | | *Dataset A* | | | *Dataset B* | | | |
|---|---|---|---|---|---|---|---|---|
| **Model** | **Vectorisation** | **$F_2$-score** | **St. Dev** | **Rank** | **$F_2$-score** | **St. Dev** | **Rank** | **Avg Rank** |
| NB | W2V-CBOW | 0.305 | *0.03* | 19 | 0.578 | *0.04* | 18 | **18.5** |
| NB | W2V-SKIP | 0.306 | *0.03* | 18 | 0.607 | *0.04* | 16 | **17** |
| NB | tf-idf | 0.197 | *0.06* | 30 | 0.555 | *0.05* | 26 | **28** |
| NB | Flair | 0.297 | *0.02* | 21 | 0.568 | *0.04* | 23 | **22** |
| SVM | W2V-CBOW | 0.308 | *0.09* | 17 | 0.646 | *0.07* | 12 | **14.5** |
| SVM | W2V-SKIP | 0.374 | *0.07* | 8 | 0.671 | *0.05* | 7 | **7.5** |
| SVM | tf-idf | 0.339 | *0.09* | 12 | 0.636 | *0.05* | 14 | **13** |
| SVM | Flair | 0.360 | *0.08* | 10 | 0.658 | *0.06* | 8 | **9** |
| SGD | W2V-CBOW | 0.364 | *0.06* | 9 | 0.638 | *0.06* | 13 | **13** |
| SGD | W2V-SKIP | 0.411 | *0.04* | 5 | 0.634 | *0.05* | 15 | **15** |
| SGD | tf-idf | 0.398 | *0.06* | 6 | 0.694 | *0.06* | 4 | **5** |
| SGD | Flair | 0.380 | *0.06* | 7 | 0.657 | *0.06* | 9 | **8** |
| GBM | W2V-CBOW | 0.215 | *0.03* | 28 | 0.445 | *0.06* | 29 | **28.5** |
| GBM | W2V-SKIP | 0.202 | *0.06* | 29 | 0.550 | *0.05* | 27 | **28** |
| GBM | tf-idf | 0.292 | *0.08* | 22 | 0.545 | *0.07* | 28 | **25** |
| GBM | Flair | 0.239 | *0.04* | 27 | 0.564 | *0.05* | 24 | **25.5** |
| RF | W2V-CBOW | 0.348 | *0.08* | 11 | 0.598 | *0.05* | 17 | **14** |
| RF | W2V-SKIP | 0.327 | *0.05* | 13 | 0.649 | *0.06* | 11 | **12** |
| RF | tf-idf | 0.298 | *0.05* | 20 | 0.563 | *0.07* | 25 | **22.5** |
| RF | Flair | 0.314 | *0.07* | 15 | 0.651 | *0.04* | 10 | **12.5** |
| MLP | W2V-CBOW | 0.320 | *0.05* | 14 | 0.673 | *0.05* | 6 | **10** |
| MLP | W2V-SKIP | 0.533 | *0.15* | 2 | 0.696 | *0.03* | 3 | **2.5** |
| MLP | tf-idf | 0.581 | *0.15* | 1 | 0.882 | *0.10* | 1 | **1** |
| MLP | Flair | 0.522 | *0.19* | 3 | 0.800 | *0.07* | 2 | **2.5** |
| LSTM | W2V-CBOW | 0.267 | *0.00* | 26 | 0.574 | *0.00* | 21 | **23.5** |
| LSTM | W2V-SKIP | 0.267 | *0.00* | 23.5 | 0.574 | *0.00* | 21 | **22.25** |
| LSTM | tf-idf | 0.267 | *0.00* | 25 | 0.574 | *0.00* | 21 | **23** |
| RCNN | W2V-CBOW | 0.464 | *0.16* | 4 | 0.575 | *0.04* | 19 | **11.5** |
| RCNN | W2V-SKIP | 0.313 | *0.07* | 16 | 0.683 | *0.09* | 5 | **10.5** |
| RCNN | tf-idf | 0.267 | *0.00* | 23.5 | 0.392 | *0.27* | 30 | **26.75** |

Table 15: Friedman Ranks per dataset and averaged

This problem has 29 degrees of freedom, noted as $df$, which is the number of possible combinations between models and vectorisation methods, which is 30 and noted as $k$, minus one. The number of datasets is two, noted as $n$. The $R$ is the summed rank per model-vectorisation combination, after which the $R^2$, the squared Friedman Rank, is calculated for each model-vectorisation combination. The Friedman statistic $Q$ is calculated as

$$Q = \frac{12n}{k(k+1)} \sum_{j=1}^{k} (R - \frac{k+1}{2})^2.$$

By taking the chi-squared distribution of $Q$ and $df$, the $p$-value is calculated. When $p$ is lower than $\alpha$, which is 1 minus the confidence level, the hypothesis that the mean model-vectorisation performances are the same can be rejected. With a confidence level of 99%, the value of $\alpha = 0.01$. Based upon the values given by Table 15, $Q$ and $R^2$ can be calculated, which are respectively 52.648 and 36991. This results in a $p$-value of 0.0046. Since this $p$-value is lower than $\alpha$, it can be assumed that the performance of the model-vectorisation combinations is significantly different.

Based upon the averaged Friedman Ranks, the chosen model is the Multilayer Perceptron, combined with tf-idf vectorisation. This is a neural network, whereby the optimized hyperparameters can be seen in Table 16. In further experiments, only this model is used.

| Parameter | Dataset A | Dataset B |
|---|---|---|
| Number of input nodes | 136 | 194 |
| Number of fully-connected hidden nodes | 49 | 200 |
| Number of fully-connected hidden layers | 2 | 3 |
| Dropout | 0.3 | 0.4 |
| L2-regularizer | $2 \times 10^{-5}$ | $10^{-8}$ |
| Activation function | Tanh | ReLU |
| Threshold | 0.1 | 0.1 |
| Optimization function | Adam | Adam |
| Learning rate | $10^{-5}$ | $10^{-5}$ |
| Batch size | 1 | 1 |
| Epochs | 25 | 25 |

Table 16: Optimal parameters for the MLP with tf-idf vectorisation per dataset

## 5.4 Inclusion of Metadata

The first step in adding the metadata is to inspect the effect of adding the subject to the tokens.

For each dataset, the performance of the classifier is evaluated twice. One time it is evaluated with only the tokens of the email body and one time including the tokens of the subject, to test whether adding the subject tokens improves the $F_2$-score and leads to significantly less misclassified instances. The confidence interval is defined as 99%, thus the alpha $\alpha$ is 0.01. The performance is measured by using five-fold cross-validation over two random states, thus creating ten observations per dataset. The average $F_2$-score over these ten observations is noted as the result for each condition, which can be seen in Table 17.

| Dataset | Without subject | With subject |
|---|---|---|
| Dataset A | 0.579 (0.138) | 0.626 (0.150) |
| Dataset B | 0.889 (0.097) | 0.888 (0.080) |

Table 17: $F_2$-scores and their standard deviation with and without the subject per dataset

For dataset A, the model performance improves when adding the subject. When the subject is added to the tokens of dataset B, the $F_2$-score is almost equal, except for a lower standard deviation. When applying McNemar's test to both, the $p$-values are $2.32 \times 10^{-10}$ and 0.68 for dataset A and dataset B respectively. For dataset A, this value is lower than the alpha $\alpha$ of 0.01, thus it can be said that with 99% confidence the performance is significantly different when the subject is added. This is not the case for dataset B, in which case the models perform roughly equally. However, due to the similar performance and lessened standard deviation, the addition of the subject is not considered a negative influence at all. Therefore, the subject is included in the next steps.

To explore the metadata features, the mutual information scores between each metadata feature and the class labels are calculated per dataset. This results in the values as can be seen in Table 18.

| **Metadata feature** | **Dataset A** | **Dataset B** |
|---|---|---|
| Number of attachments | 0.008 | 0.005 |
| Importance | 0.000 | 0.001 |
| Time of day | 0.136 | 0.308 |
| Weekday | 0.003 | 0.005 |
| Day *(all days)* | 0.145 | 0.289 |
| Day *(only mail days)* | 0.246 | 0.514 |
| Timestep *(steps=10)* | 0.003 | 0.060 |
| Timestep *(steps=50)* | 0.024 | 0.107 |
| Timestep *(steps=100)* | 0.039 | 0.143 |
| Timestep *(steps=250)* | 0.086 | 0.242 |
| Timestep *(steps=500)* | 0.120 | 0.356 |
| Number of receivers | 0.003 | 0.011 |
| Betweenness sender | 0.000 | 0.000 |
| Betweenness receiver(s) | 0.000 | 0.000 |
| Closeness sender | 0.000 | 0.000 |
| Closeness receiver(s) | 0.000 | 0.000 |
| Degree sender | 0.026 | 0.059 |
| Degree receiver(s) | 0.000 | 0.000 |

Table 18: Mutual information of metadata features per dataset

A forward and a backward feature selection search is applied per dataset after the classification model is pre-trained on the content and subject tokens. The best performing subset per search direction and dataset can be seen in Table 19. Of these four subsets, the union is taken per dataset and direction, which results in nine subsets of metadata features.

| **Dataset** | **$F_2$-score** | **No. of features** | **Features** |
|---|---|---|---|
| | | | *Forwards* |
| Dataset A | 0.5028 | 5 | Weekday, importance, no. of receivers, closeness (to), degree (to). |
| Dataset B | 0.7184 | 1 | No. of receivers |
| | | | *Backwards* |
| Dataset A | 0.4776 | 9 | Day numbered, day (only mail days), timestep (10), timestep (50), timestep (100), no. of receivers, closeness (to), degree (to), betweenness (to). |
| Dataset B | 0.6720 | 1 | Day (only mail days). |

Table 19: Best performing metadata subsets and their $F_2$-score, per search direction and dataset

Since the subsets of dataset B are included in the subsets of dataset A for both directions of the feature selection search, the three union subsets over the datasets are not included. They would contain the same subsets as the subsets found for just dataset A. To note the subsets, the first letter of the direction or directions of the search is combined with the number of the dataset the search was for. Thus the subsets that are evaluated are F-A, F-B, B-A, B-B, BF-A and BF-B. The evaluation of these subsets on both datasets is shown in Table 20, whereby they are evaluated together with the tokens of the content and subject, using the model optimised for that dataset.

| Subset | Dataset A | Dataset B |
|---|---|---|
| Without metadata | 0.579 (0.138) | 0.889 (0.097) |
| With subject | 0.626 (0.150) | 0.888 (0.080) |
| F-A | 0.584 (0.162) | 0.836 (0.152) |
| F-B | 0.621 (0.147) | 0.888 (0.085) |
| B-A | 0.583 (0.167) | 0.824 (0.145) |
| B-B | 0.615 (0.150) | 0.862 (0.124) |
| BF-A | 0.580 (0.169) | 0.822 (0.146) |
| BF-B | 0.616 (0.155) | 0.862 (0.124) |

Table 20: $F_2$-scores and their standard deviation for each metadata subset

This evaluation shows that none of the metadata subsets found performed consistently better than the baseline. Of the found subsets, the ones optimised for dataset B show the most potential, of which the F-B seems to increase the performance more, compared to B-B and BF-B. The metadata subsets optimised for dataset A show a decline in performance. This is true for F-A, B-A and BF-A on both dataset A and dataset B, where they all have $p$-values lower than 0.0001 on McNemar's test, meaning they misclassified significantly more emails.

The outcome of this evaluation is that none of the metadata features is added to the input data.

## 5.5 Transfer Learning

This section shows the results for testing the learning bounds and evaluating the domain adaptation methods. For reference, the methods that are evaluated have been summarized in Table 21.

| Method | Explanation |
|---|---|
| *TgtOnly* | Train and test the model on the target data |
| *All* | Train and test the model on all data |
| *Pred* | A pre-trained source model predicts the target labels, these predictions are added as an extra feature to the target dataset. Then the model is trained and test the target data, including the extra feature. |
| *SrcPrior* | Train the source model on the source data, then fine-tune and test this model on the target data. |
| *TgtPrior* | Train the target model on the source data, then fine-tune and test this model on the target data. |
| *TrAdaBoost* | Train and test a model on all data iteratively, updating the weights of samples based on whether they are misclassified. |
| *TCA* | Identify the transfer components on all data, then train and test a classifier on these components. This classifier is either an MLP or 1-NN. |
| *DANN* | A model that creates a latent subspace for all data, after which both the domain and the label of the data are predicted. The domain classification loss should be maximised and the label classification loss minimized. |

Table 21: Summary of domain adaptation methods

### 5.5.1 Learning Bounds

Before the actual transfer of knowledge, the learning bounds should be evaluated. The Kullback-Leibler divergence $D_{KL}$ between the domains is based on the probability distributions in Table 22.

| $x$ | **Non-responsive** | **Responsive** |
|---|---|---|
| Distribution $P_A(x)$ | 0.068 | 0.932 |
| Distribution $P_B(x)$ | 0.212 | 0.788 |

Table 22: Probability distributions of dataset A and dataset B

Next, the classifier with the highest performance for both datasets should be found. Therefore, the models optimised for dataset A and dataset B are tested on both datasets separately. Both models have a different input layer and separate hidden layers. The number of hidden layers along with its activation function differ per model. To combine this, the hidden layers are concatenated in a fully-connected manner. Apart from the dissimilarity in hidden layers, both models use the same output layer, optimization function, learning rate, epochs and batch size. It could be argued that by concatenating the hidden layers, this new model can pick up on both phenomena, due to its fit to both datasets and increased complexity. The layers can be concatenated in both directions, either starting with the three hidden ReLU layers or with the two hidden tanh layers. The hidden layers are fully-connected, the output layer with its one sigmoid node remains only at the end. The results for the different model architectures, evaluated per dataset, can be seen in Table 23.

| **Model architecture** | **Dataset A** | **Dataset B** |
|---|---|---|
| Dataset A | 0.623 (0.149) | 0.807 (0.117) |
| Dataset A - Dataset B | 0.630 (0.168) | 0.858 (0.057) |
| Dataset B - Dataset A | 0.693 (0.203) | 0.877 (0.095) |
| Dataset B | 0.694 (0.211) | 0.900 (0.087) |

Table 23: $F_2$-scores and their standard deviation for the different model architectures per dataset

Based on these results, the model architecture optimised for dataset B shows the best result. Therefore, this model is used as classifier $h$ in the learning bounds formulas. With this classifier $h$, the average error rates for five-fold cross-validation are noted as $\epsilon_{\mathcal{D}_1}(h)$ and $\epsilon_{\mathcal{D}_2}(h)$ for respectively dataset A and dataset B. The sum of the average error rates is $\kappa$. The values for the equations for the learning bounds are the following:

$$\begin{aligned}
D_{KL}(P_1||P_2) &= 0.114 \\
D_{KL}(P_2||P_1) &= 0.157 \\
\epsilon_{\mathcal{D}_1}(h) &= 0.131 \\
\epsilon_{\mathcal{D}_2}(h) &= 0.063 \\
\kappa &= 0.194 \\
0.063 = \epsilon_{\mathcal{D}_2}(h) &\leq \epsilon_{\mathcal{D}_1}(h) + \kappa + D_{KL}(P_1||P_2) = 0.439 \\
0.131 = \epsilon_{\mathcal{D}_1}(h) &\leq \epsilon_{\mathcal{D}_2}(h) + \kappa + D_{KL}(P_2||P_1) = 0.414
\end{aligned}$$

This shows that the equations hold for both transfers, thus domains appear similar enough for a positive knowledge transfer in either direction.

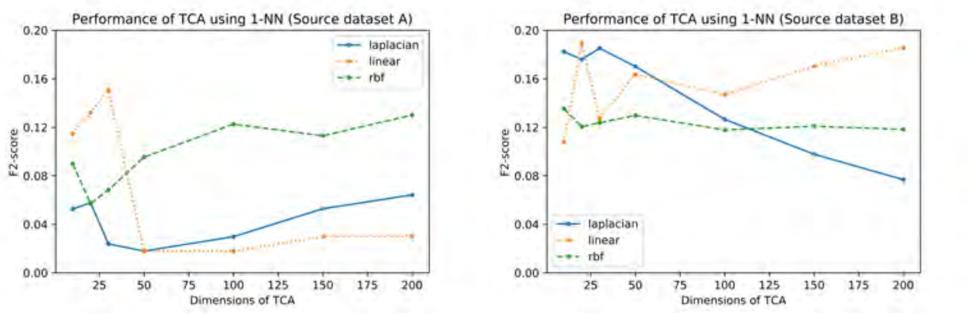
To enable an easy transfer between both domains in the domain adaptation experiments, the dimensionality of both datasets are made equal. Once the initial tf-idf cut-off has been applied, the dimensionality of a data point is 5119 in dataset A, while it is 5549 in dataset B. The data of dataset A is padded with zeros to 5549, so each model can be trained and tested on either dataset.

### 5.5.2 Unsupervised Approach

The baseline of the unsupervised method is the *SrcPrior* method, which trains the model, which is optimised on the source dataset, on the source data before evaluating it on the target dataset. This method is applied using both optimal model architecture, the one optimised for dataset A and dataset B.

To apply *TCA*, its kernel and number of dimensions need to be determined. This is done separately for the variation using 1-NN and the one using an MLP. The performance of *TCA* using 1-NN can be seen in Figure 8. Based on this graph, the parameters are determined as a linear kernel with 20 dimensions, which had the least relative loss.

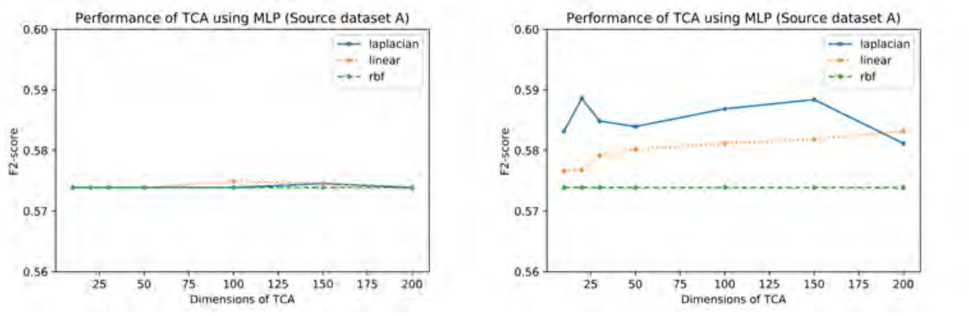
For the MLP variation, the performance for the transfer from dataset A to dataset B can be seen in Figure 9 and the performance for the transfer from dataset B to dataset A in Figure 10. The parameters for the MLP variation are determined as the Laplacian kernel with 150 dimensions.



(a) *TCA* using 1-NN from dataset A to dataset B

(b) *TCA* using 1-NN from dataset B to dataset A

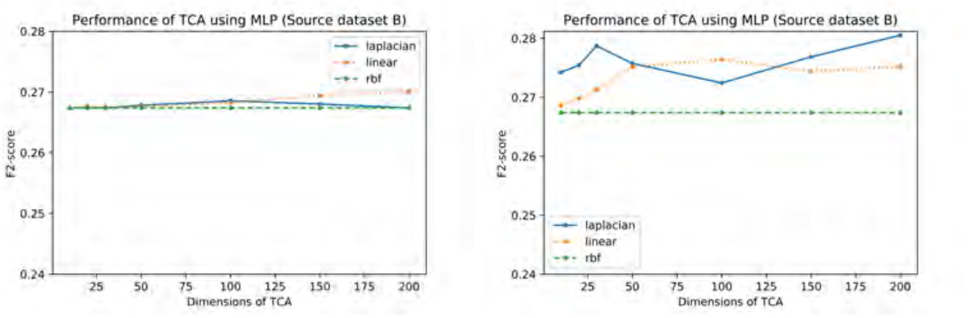
Figure 8: Performance of *TCA* with 1-NN



(a) *TCA* using MLP with model architecture dataset A

(b) *TCA* using MLP with model architecture dataset B

Figure 9: Performance of *TCA* with MLP from dataset A to dataset B



(a) *TCA* using MLP with model architecture dataset A (b) *TCA* using MLP with model architecture dataset B

Figure 10: Performance of *TCA* with MLP from dataset B to dataset A

To optimise DANN, the activation function, number of nodes, learning rate, adaptation rate and output function are tuned. The optimal parameters found per transfer and the parameters with the least relative loss, considered the most optimal for both, are shown in Table 24. The optimal $F_2$-score for the transfer from dataset A to dataset B is 0.5739, for the transfer from dataset B to dataset A the $F_2$-score is 0.2832. For the transfer from dataset A to dataset B, the best performance is achieved by classifying all emails as responsive, in most cases using the threshold and a sigmoid hidden layer, while the output layer also uses the sigmoid function. When all emails are classified as responsive, the $F_2$-score is 0.5739, which is noted 469 out of 1620 times during the optimisation of *DANN*. The parameter combination shown in the table is the first combination tried that achieved the highest $F_2$-score. After considering the relative loss for each parameter combination per transfer, the least relative loss is 0.0378. This loss is based on the $F_2$-score that is achieved with this parameter combination for the transfer from dataset B to dataset A, which is 0.2725. The parameters that are noted as optimal in the table are used to evaluate *DANN*.

| **Parameter** | **Dataset A → Dataset B** | **Dataset B → Dataset A** | **Optimal** |
|---|---|---|---|
| Hidden nodes | 1 | 1 | 100 |
| Activation function | Sigmoid | Sigmoid | Tanh |
| Learning rate | $10^{-1}$ | $10^{-1}$ | $10^{-1}$ |
| Adaptation rate | 1.00 | 1.00 | 0.3162 |
| Output function | Sigmoid | Softmax | Sigmoid |
| Threshold | True | True | True |

Table 24: Optimised parameters for *DANN*

The evaluated methods for the unsupervised transfer are shown in Table 25. Based on these results, a few observations can be made. Between the *SrcPrior* and *TgtPrior* method, *TgtPrior* shows the best results for both transfers.

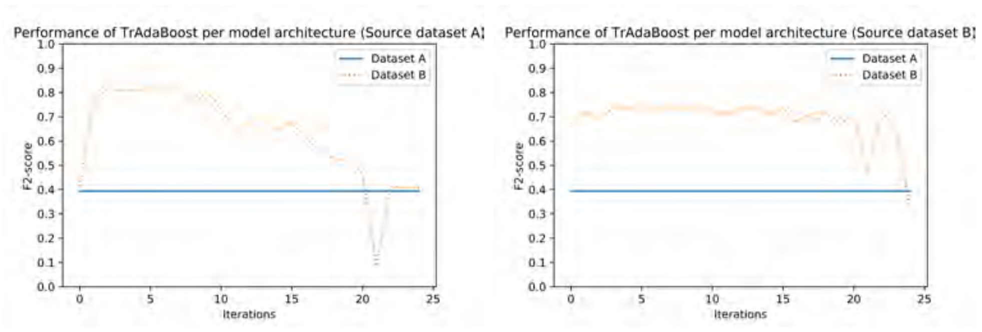
The *TCA* method using an MLP shows that for both transfers the model optimised for dataset B slightly outperforms the model optimised for dataset A. The latter model architecture classifies almost every email as responsive, while the first one puts more effort into classifying emails as both classes. This also reflects in their standard deviations. However, the differences in performance are only marginal. For the 1-NN variation, no random states are used, thus resulting in one value without a standard deviation.

| **Method** | **Model architecture** | **Dataset A → Dataset B** | **Dataset B → Dataset A** |
|---|---|---|---|
| SrcPrior | Src | 0.253 (0.041) | 0.104 (0.080) |
| TgtPrior | Tgt | 0.318 (0.196) | 0.164 (0.051) |
| TCA | 1-NN | 0.132 (-) | 0.189 (-) |
| TCA | MLP (Dataset A) | 0.574 (0.000) | 0.268 (0.000) |
| TCA | MLP (Dataset B) | 0.581 (0.007) | 0.270 (0.004) |
| DANN | - | 0.574 (0.000) | 0.270 (0.004) |

Table 25: $F_2$-scores and their standard deviation for the unsupervised transfer methods

### 5.5.3 Semi-supervised Approach

The only semi-supervised method of which a parameters needs to be optimised is the *TrAdaBoost* method, of which the number of iterations needs to be determined. The performance over the iterations is visualised for both transfers in Figure 11, whereby for each transfer both model architectures are evaluated in the same figure. The architecture optimised for dataset B showed actual results, while the architecture optimised for dataset A classified every email as responsive. This is reflected in the straight line, showing no improvement. Based on the results, the number of iterations is set as eight, since this gives the least relative loss.



(a) Performance of *TrAdaBoost* from dataset A to dataset B

(b) Performance of *TrAdaBoost* from dataset B to dataset A

Figure 11: Performance of *TrAdaBoost* using either model architecture

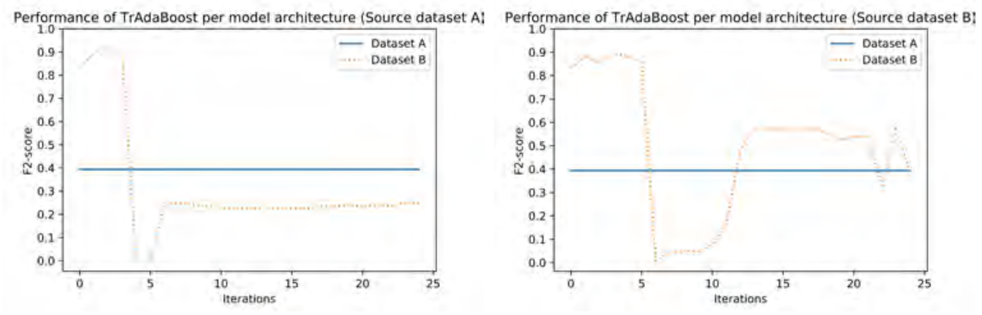
The performance of the methods that are evaluated for the semi-supervised transfer can be seen in Table 26. It shows that between *SrcPrior* and *TgtPrior*, the model architecture optimised for dataset B performs best in both cases. This is different from the unsupervised setting, where the *TgtPrior* method performs better in both cases. The *Pred* method does not perform well, though not worse than the baseline. The *TrAdaBoost* method only shows a learning curve with the model architecture optimised for dataset B, as with the one optimised for dataset A, the model returned every email as responsive. That also reflects in the standard deviation of both *TrAdaBoost* methods with the architecture optimised on dataset A.

| Method | Model architecture | Dataset A → Dataset B | Dataset B → Dataset A |
|---|---|---|---|
| TgtOnly | Tgt | 0.690 (0.105) | 0.356 (0.051) |
| SrcPrior | Src | 0.733 (0.076) | 0.622 (0.147) |
| TgtPrior | Tgt | 0.752 (0.090) | 0.535 (0.121) |
| Pred | Src - Tgt | 0.692 (0.109) | 0.390 (0.112) |
| TrAdaBoost | Dataset A | 0.394 (0.000) | 0.394 (0.000) |
| TrAdaBoost | Dataset B | 0.562 (0.217) | 0.685 (0.321) |

Table 26: $F_2$-scores and their standard deviation for the semi-supervised transfer methods

### 5.5.4 Supervised Approach

For the supervised *TrAdaBoost*, the number of iterations needs to be determined as well. In Figure 12 the $F_2$-scores over the iterations of *TrAdaBoost* are shown per transfer. Based on these results, the number of iterations is set at four. This means that the supervised variant of *TrAdaBoost* needs only half as many iterations as the semi-supervised variant.



(a) $F_2$-scores over the iterations of *TrAdaBoost* from dataset A to dataset B (b) $F_2$-scores over the iterations of *TrAdaBoost* from dataset B to dataset A

Figure 12: Performance of *TrAdaBoost* using either model architecture

The results of all methods can be seen in Table 27. Of the results for *TgtOnly*, for both transfers the architecture optimised for dataset A performs poorest, followed by the dataset A - dataset B concatenation. The dataset B architecture and the one starting with the hidden layers optimised for dataset B performed best and second-best for both transfers.

For the *SrcPrior* and *TgtPrior* methods, the one using the dataset B architecture performs best, which is similar to the semi-supervised result. The *Prior* method based upon the dataset B architecture performs best on the dataset A to dataset B transfer and second-best on the dataset B to dataset A transfer. The *Pred* method performs slightly poorer than the *Prior* method using the dataset B architecture but equal or better than the *Prior* method using the dataset A architecture.

The *TrAdaBoost* method classified all emails as responsive with model architecture A, while with model architecture B the method performance can be compared to the *All* method for the dataset A to dataset B transfer, while it is the top performance for the dataset B to dataset A transfer.

| **Method** | **Model architecture** | **Dataset A $\rightarrow$ Dataset B** | **Dataset B $\rightarrow$ Dataset A** |
|---|---|---|---|
| TgtOnly | Tgt | 0.884 (0.100) | 0.631 (0.137) |
| All | Tgt | 0.780 (0.139) | 0.750 (0.000) |
| All | Src - Tgt | 0.733 (0.122) | 0.765 (0.144) |
| SrcPrior | Src | 0.859 (0.087) | 0.777 (0.175) |
| TgtPrior | Tgt | 0.910 (0.091) | 0.684 (0.117) |
| Pred | Tgt - Src | 0.891 (0.079) | 0.684 (0.211) |
| TrAdaBoost | Dataset A | 0.394 (0.000) | 0.394 (0.000) |
| TrAdaBoost | Dataset B | 0.723 (0.260) | 0.806 (0.050) |

Table 27: $F_2$-scores and their standard deviation for the supervised transfer methods

# 6 Explainable AI

Explainable AI (XAI) can help to increase the transparency of the model and thereby the trust in the model. Especially once a task has to be done in compliance with legislation, models can only be used when they are transparent [95]. Explainability has potential in the legal domain, since algorithms should behave in an equitable, honest and non-discriminatory manner to be used in the process [2]. In addition, XAI may be used to prove algorithms behave in a non-discriminatory manner, so their outcomes can be used as evidence.

## 6.1 Implications for Legal Technology

One of the downsides of using machine learning or artificial intelligence in the legal domain is that many algorithms are not transparent. To some extent, there is a certain reluctance to use it, according to Butterfield *et al.* [24] this is because of a lack of understanding of Technology Assisted Review (TAR) and comfort with old patterns. This is partly caused by TAR not being completely accepted in courts in the USA, due to a lack of transparency in the methodology and process. This lack of transparency in the USA might be caused by the work-product doctrine, which states that information about the work that is done to handle a case is privileged. This is similar to attorney-client privilege, through which all communication between an attorney and their client is privileged and cannot be used in a case. However, opposed to attorney-client privilege, which only concerns an attorney and their client, the work-product doctrine entails all people that worked on the case. Hereby their work product would be their judgement over the relevance of the emails. The doctrine is used to prevent them from having to explain their way of working. In case they would use an e-discovery classification model, their way of working would include the creation of a seed set, which is needed for training the model. If this seed set would need to be disclosed, this could cause issues, since this would include having to disclose the non-responsive emails in the seed. This disclosure is outside of the scope of discovery obligations. However, there is also support for creating a transparent TAR process, such as the Cooperation Proclamation of the Sedona Conference, which 'urges parties to work in a cooperative rather than an adversarial manner to resolve discovery issues to stem the monetary costs of discovery disputes' [104]. It is argued that the costs of developing a transparent solution outweigh the money saved by employing TAR [24].
When the TAR models that make use of AI become more explainable, it can gain them more acceptance in court. There are several other reasons for introducing explainable AI (XAI). According to Adadi *et al.* [2], these are:

- Explain to **justify**, when the system makes an unexpected decision, it can be justified more easily once the model is explainable, also the system operating ethically and without bias can be checked;
- Explain to **control**, by improving the understanding of how the system behaves, potential weaknesses and errors can be identified sooner, leading to increased control over the system;
- Explain to **improve**, when a model can be explained and understood, it can also be improved more easily; and
- Explain to **discover**, since models might be able to outperform humans at a task, an explainable model lead to new insights.

These are confirmed by Samek *et al.* [95], who define the reasons for using XAI as being for compliance with legislation, verification of the system, improvement of the system and to learn from the system. Thereby their reasoning largely overlaps with the reasons of Adadi *et al.* [2].
Waltl *et al.* reported specifically on the need for explainability in machine learning in legal technology [108]. Due to the criticality of being able to explain how you gathered your information

to the court, the used algorithms need to be explainable. Another issue limiting the use of new technologies is the so-called ‘stare decisis’, which states that cases should be handled in the same way as precedent similar cases. In the Netherlands, this principle is handled less rigidly, though it exists and is sometimes considered the right way to act [44]. However, when using machine learning for e-discovery, it should be clear how the possible evidence was selected. This contributes to the current way of working, which still uses search terms and filters, remaining that way. It all boils down to a need to be interpretable and transparent, both to a client and to the court.

An example of the use of AI in legal technology can be seen in the USA, where they have an AI system called COMPAS that predicts how likely a convict is to re-offend [57]. This system should be fair and non-discriminatory and the only way this can be ensured is by explainability. There has been a case, Loomis vs. Wisconsin, in which the use of an AI system to decide over someone’s sentence was disputed [2]. The workings of COMPAS were not obvious to the Judge since the system is closed and its algorithm secret.

This example highlights the need for Automated Decision Making (ADM) to be explainable. Especially in critical domains, such as the legal system, there is not yet a lot of work done to make the ADM processes explainable [2]. Processes that are using ADM are occurring more and more often, with the possibility of affecting our life [108]. Waltl *et al.* [108] mention three reasons for the rise in ADM. First off, it is because of digital data becoming more easily available, which is needed to apply machine learning. Next, computing infrastructures can handle increasingly larger amounts of data. Lastly, the algorithms that are currently used can process increasingly complex tasks.

These ADM systems might need to conform to certain rules, to protect the consumers. For example, it may not discriminate based upon gender, race or age. In the 1990s, research showed that people already had expectations for ADM systems, legal expert systems in specific [108]. Two of these expectations were related to the current concept of explainability, namely that the ADM should be able to easily understand and follow the knowledge representation and that its decisions should be explainable and transparent. With XAI becoming more prominent, these expectations could be met, enabling a carefully automated process.

In recent years, some studies have been done to the use of interpretable machine learning in critical ADM systems, such as the proposal for simple rules for bail decisions of Jung *et al.* [66], Corbett-Davies *et al.* [32] discussing the fairness of algorithms that are used for making consequential decisions and Germany running a project, AlgorithmWatch [87], to check the fairness of among others their video surveillance system and credit scoring.

Predictions made by an AI model are in principle mathematical operations, which follow strict procedures. By making them more transparent, they would be more explainable. This could cause an increase in trust that these systems make fair decisions, causing the acceptance of these systems to grow [108]. In addition to that, by making the AI interpretable, new insights to optimise the systems could be gathered. The system might be able to explain how it classifies responsive emails, possibly uncovering hidden patterns indicating fraudulent behaviour that are not intuitive for people.

The roadmap for the use of XAI is thus not laid out yet. The use of AI within ADM seems to gain momentum, but there is resistance due to its complex nature and opacity. If explainable models would be applied to e-discovery, new hidden patterns in emails relating to their responsiveness might be uncovered. Next to that, there is more control over the process, due to an increased insight into the behaviour of the system. Thereby it can be checked whether the system operates fairly and without any bias. This could lead to a system being trusted more and gaining acceptance, which could increase its use and open the way for more development.

# 7 Discussion

In this chapter, the findings shown in the previous chapter, Chapter 5, are discussed. For this, the errors and their consistency are analysed. Furthermore, the results of the classifier comparison, the inclusion of the metadata and the transfer learning results are discussed.

## 7.1 Error Analysis

After manually inspecting the misclassified emails, there does not appear to be a reason for why they are misclassified. It is not because of their language, because both English and multi-lingual emails get correctly classified and misclassified to the same extent. The same is true for short and long emails, which also get equally misclassified, the number of tokens an email contains does not seem to play a role in this. Furthermore, the words occurring in the emails were considered. Some words are case-specific, such as specific proper nouns that are tied to the case. However, these words occurred as often in correctly classified emails as in misclassified ones, for both responsive and non-responsive emails. The topic of a message and whether the message contains a call to action do not seem to be reasons for the classifier to classify them incorrectly. The same is the case for emails centring around an attachment, a link to an attachment or a past or future meeting, as well as for emails containing tables and emails with lots of forwards.

Since the acquisition of the emails is already focused on only getting emails that have a chance of being responsive, the emails are effectively pre-filtered. This pre-filtering consists of the choices for whose email boxes are secured and if the entire email box is secured, or only after a certain date. It could be that because of this pre-filtering, the proper nouns and words associated with the case are seen as less important. Most of the responsive emails may revolve around the same subject, but a lot of the non-responsive emails do as well. In this case, the classifier might pick up on a pattern that is not intuitive for people. The classifier could be classifying emails as responsive based on 'common' words that occur more often in responsive emails, but that we would not consider informative or indicative as the source of responsiveness.

All in all, no clear characteristics are found for the misclassification during the manual review of the misclassified emails. By visualising the emails in a matrix, visualising per method which emails are misclassified, possible patterns might be found of some emails getting misclassified never or all the time. These matrices can be seen in Appendix 8, whereby each email is represented by a thin row. After inspecting these, it seems that there is some consistency across rows. However, when looking into these emails, no patterns can be found to trace the partial consistency back to common characteristics.

## 7.2 Vectorisation and Model Optimisation

The hyperparameters for the neural networks have been optimised with BO-GP. One of the disadvantages of this optimisation method is that it models the loss over the parameters that are optimised, whereby it could be that the most optimal setting is never found due to the loss function getting stuck in a local optimum. Due to limited resources, this could not be circumvented. An example of a local optimum in this study is that the MLP with the architecture optimised for dataset B has a better performance on dataset A than the architecture that is optimised on dataset A itself. The architecture optimised on dataset A does not combine well with dataset B, which could indicate that these parameters have overfitted on dataset A.

During the sequential vectorisation, an observation is defined as a token. Experiments can be done by defining an observation differently, for example as a sentence or an n-gram. For the sequential tf-idf vectors, only the tf-idf value for the token itself is noted. This erases a lot of information about the vocabulary that is available the entire tf-idf vector is used. An option to circumvent this would be to perform Principal Component Analysis (PCA) on the original tf-idf

vectors. PCA could also be applied to the non-sequential tf-idf and word embedding vectors, to possibly increase feature usability.
Furthermore, the sequential models have a weak performance compared to the other models. This could be due to a sequential model not being the best fit for an e-discovery problem, but it is more likely that it would need more complexity to correctly model the case. Due to limited resources, not all hyperparameters were fully explored. For the LSTM model, the number of both LSTM and fully-connected layers could be increased, as well as the number of neurons they contain. Both mono-directional and bidirectional LSTM layers can be alternated. Aside from that, the sequences could be maximised to a quartile higher than 90 and the processing could be tried without bucketing. For both the LSTM and the RCNN, the batch size could be adjusted more freely.
During the optimisation process, the batch size could have been included in the hyperparameter search. To do this and get to a stable result, the number of runs during the search should have been increased, to ensure an optimum is found.
Once the optimised values per model were found, it showed each vectorisation manner performed differently. The W2V-CBOW embeddings show an overall weak performance, for none of the models it is the best vectorisation manner and for five of the eight models it has the lowest performance. The Flair embeddings show an average performance, neither being the best nor the worst vectorisation method for each model. For all the models, the best performance is achieved using either W2V-SKIP embeddings or tf-idf vectors. Thereby the tf-idf vectors perform best for the SVM with SGD training, GBM and MLP, while the W2V-SKIP has the best performance for the NB, SVM, RF, LSTM and RCNN. It is worth noting that the results for the SVM and SGD implementations are quite different per vectorisation manner, showing that the training method of the SVM affects the results. Aside from that, the SVM with tf-idf values is a strong baseline to beat, since it would have the highest score if it was not for the MLP.

## 7.3 Inclusion of Metadata

The results show that the only metadata that improves the performance or performs equally well on either dataset is the subject. The rest of the metadata features decrease the performance of at least one model. A possible explanation for this is that the model parameters are optimised on only the token data. Therefore, the found architectures might not be beneficial for the added metadata, which can consist of other features than text vectorisation. This could also explain why adding the subject does improve the performance of the model.
The subsets of metadata that performed best for each dataset were defined based upon their performance on one iteration, instead of on multiple iterations using cross-validation or different random states. Therefore, it could be that other metadata sets might be equally or better suited when tested more thoroughly. The optimal sets found contained much more metadata features for dataset A than for dataset B. This is unexpected since the mutual information scores between the metadata features and the class labels are higher for dataset B. The metadata feature with the highest mutual formation on both datasets, the day (only mails days), only shows in the backwards feature selection. Other metadata features with higher scores, namely the time of day, timestep (steps=500) and day (all days) are not included in any of the optimal subsets. This shows that the mutual information is not very informative for whether the metadata features would improve the performance when combined with the token data. The subset of metadata features that performs best when combined with the tokens, namely the F-B subset, only contains the metadata feature number of receivers. This feature does not have a high mutual information score, but this seems true for all features found by forward feature selection. Thereby it is confirming that the mutual information scores are not indicative for improved performance on the model optimised on the tokens. In this case, they seem to generate more noise. The performance decreases, since a lot of non-responsive emails are classified as responsive.
In the current setting, the subject is concatenated to the tokens of the body as to form the

input data. Since the subject contains fewer tokens and thus is more specific, it is a possibility to create two input layers, one for the tokens and one for the metadata features, and form a multi-input model. Hereby the input layer for the subject is merged into the network a few layers in. This would allow the first layers to create a latent space for the features created of the email body, after which the subject features are concatenated and the data is processed further.

## 7.4 Transfer Learning

During the evaluation of the four possible model architectures on both datasets, the architecture optimised on dataset B performed best on both datasets. This indicates that the hyperparameter optimisation on the dataset A has not found this optimum. This is always a possibility with Bayesian Optimisation, where not all combinations are tested, unlike to the exhaustive grid optimisation. The architecture with the weakest performance on both datasets is the one concatenating the hidden layers of dataset A and dataset B. This concatenation starts with 2 hidden layers of 49 nodes with the tanh activation function, followed by 3 hidden layers of 200 nodes with the ReLU activation function. Normally, hidden layers have a decreasing number of nodes, instead of increasing. Therefore, it is expected that this concatenation would not perform optimally. The dataset B architecture and the one combining dataset B - dataset A show that larger hidden layers with a ReLU activation function seem to be more suitable to find general patterns in the data than smaller ones with a tanh function. This phenomenon can be observed on the results of the *All* method as well.

### 7.4.1 Unsupervised approach

For unsupervised transfer learning, the baseline methods *SrcPrior* and *TgtPrior* are used, of which *TgtPrior* always performs better. This indicates that the architecture optimised for each dataset increases the performance of that set, independent of whether the model was trained on that set. This is partly unexpected since dataset A performed better when trained and tested on the architecture optimised on dataset B, while in this case, the best architecture for the transfer from dataset B to dataset A is the one optimised for dataset A.
Furthermore, *TCA* showed consistently better results with the model architecture optimised for dataset B. When the model architecture optimised for dataset A is used, the method classifies every email as responsive. This performs better than when 1-NN is used to classify the transfer components. However, 1-NN does make a distinction in responsive and non-responsive emails, though not enough emails are classified as responsive for each transfer to get a competitive performance. This can be seen more clearly in the error matrix in Figures 20 and 21 in Appendix 8, where the 1-NN variant classifies fewer emails incorrectly when *TCA* uses an MLP for classification. The MLP has a higher dimensionality, which increases its ability to model more complex relations compared to using 1-NN. However, since the model is optimised on tf-idf values and not on transfer components, it could not have been expected that it would perform as well as it does for the tf-idf values.
The *DANN* method does not perform well. For both transfers, it classifies most emails as responsive. Its performance could be improved by creating a deep network, instead of the shallow one that is evaluated. For this, the implementation should be extended to include the possibility of multiple layers, whereby a division has to be made which layers form the latent space and which belong to the class classifier. The optimised hyperparameters indicate that the networks with only one node in its hidden layer perform best for the individual datasets. This could be caused by the one node making less of a division between the classes, thus classifying more emails as responsive. It is remarkable that for both datasets, the best result for each dataset individually has 1 hidden node that uses the sigmoid activation function, while the parameters with the least relative loss include 100 hidden nodes that use the tanh activation function. A reason for this is that the transfer from dataset A to dataset B seems to work better using the

sigmoid function in the output layer, though for this transfer many sets of parameters achieve the same optimal performance by classifying all emails as responsive. The transfer from dataset B to dataset A shows better results by using the softmax function in the output layer. When a different function is chosen in the output layer, the rest of the parameters is heavily influenced. Overall, the unsupervised methods seem to increase the performance of the baseline method, *SrcPrior*. This is the case for all methods, except for the *TCA* method using 1-NN in the dataset A to dataset B transfer. The method scoring best on both transfers is the *TCA* method using an MLP with a model architecture optimised on dataset B. However, this is only by a small margin, compared to the TCA method using the model architecture optimised on dataset A and the *DANN* method. The *TgtPrior* method also improved on the baseline, but this was less compared to the *TCA* method using an MLP and the *DANN* method.

### 7.4.2 Semi-supervised and supervised approaches

In the semi-supervised and supervised approaches, the performance for the transfer from dataset A to dataset B is higher overall, but the increase in performance is lower. Compared to the unsupervised approach, the influence of labelled target instances is visible, since the performance of the methods differs a lot between both transfers.
In both approaches, considering *SrcPrior* and *TgtPrior*, the *Prior* method using the model architecture based upon dataset B performs best. This is different from the results of *SrcPrior* and *TgtPrior* in the unsupervised approach, where *TgtPrior* performed better than *SrcPrior* for both transfers. An explanation for this is that in the unsupervised approach, the model is only trained on the source data, while in the (semi-)supervised approach, the model is fine-tuned on the target data. This could cause the model using the architecture optimised on dataset A to only perform well when fine-tuned on the dataset A. For the semi-supervised approach, these methods are the highest performing methods on the dataset A to dataset B transfer. On the other transfer they are only surpassed by the *TrAdaBoost* method using the dataset B model architecture.
For the *Pred* method, the results differed between the semi-supervised and supervised approach. For the semi-supervised approach, the results are only slightly better than the baseline *TgtOnly*, but worse than with either *SrcPrior* or *TgtPrior*. It could be that the weights connected to the prediction are not trained sufficiently, as they are only trained on the labelled part (20%) of the target data. In the supervised approach, the *Pred* method performs better than the *TgtOnly* and equal to or better than the *Prior* method based on the architecture of dataset A. This shows more potential than in the semi-supervised approach, possibly since to the weights connected to the predictions are fine-tuned on more data.
When considering *TrAdaBoost* in both the supervised and the semi-supervised approach, the optimal number of iterations for the supervised approach is only half of the optimal number of iterations for the semi-supervised approach. This could be caused by the increased number of target labels available for training in the supervised approach, which can increase the learning curve. *TrAdaBoost* only shows a learning curve when combined with the model architecture optimised on dataset B. This could be because of those layers using the ReLU activation function, instead of the tanh function the other architecture uses. When the architecture optimised on dataset A is used, all emails are classified as responsive each time. Both *TrAdaBoost* and *TCA* prove to be a bad combination with the model architecture optimised on dataset A. Most probably, this has to do with the tanh activation function and the hidden layers containing fewer nodes. When the architecture optimised on dataset B is used, the methods perform quite well. When applying *TrAdaBoost*, during the first few iterations it first starts classifying more emails as responsive, after which it starts classifying fewer non-responsive emails as responsive. This causes the performance to first increase. Then the performance drops when all emails are classified as non-responsive, after which the model starts making distinctions again. For the transfer from dataset B to dataset A, *TrAdaBoost* is the best performing method, while in the

other transfer it has the worst performance. This is most likely caused by *TrAdaBoost* being evaluated on both the source and the target data. Emails from dataset A are misclassified more often than from dataset B, thus the incorrectly classified source instances of dataset A decrease the performance in this case. For the transfer from dataset B to dataset A, the performance benefits from the instances of dataset B that are correctly classified more often.
Alike *TrAdaBoost*, the *All* method is evaluated on a combination of the source and target data. This is visible in the fact that their performance lies between the performances of dataset A and dataset B. The model architecture that is used seems to influence the performance, in a way that is in line with the other results. If the architecture starts with the hidden layers that were optimised for dataset B, namely 3 hidden layers with a ReLU activation function and 200 nodes, the results are highest. Adding the hidden layers optimised for dataset A, which are 2 hidden layers with a tanh activation function and 49 nodes, slightly decreases the performance. If the architecture only contains the tanh layers, the results are lower than when the model starts with the ReLU layers. The lowest performance is noted when the ReLU layers are added behind the tanh layers, in that case, the number of nodes in the hidden layers increases instead of decreases. This is uncommon since a latent space of a lower dimensionality is then transformed to a higher dimensionality, while no more data is inputted. Based on the results, the *All* method should only be used if the source set is less imbalanced than the target dataset, otherwise, it seems more likely to decrease the performance.
One of the most evident results of transfer learning is that the transfer from dataset B to dataset A shows more improvement than vice versa. The reason for this might be the source dataset. Since dataset B already showed better results when evaluated on its own, these results may strengthen the learning process on dataset A. This is most likely caused by dataset B being more balanced. It seems that when a more imbalanced set is used as the source set, the target set shows decreasing results. The only manner to increase the results of the more balanced target set is by using the source set to initialise the weights of the network.
The impact of having the actual source data available, which is needed for the *All* and *TrAdaBoost* methods, is limited. For the supervised transfer from dataset B to dataset A, the TrAdaBoost method combined with the model architecture optimised on dataset B gives the best results. However, the best results that are from the *SrcPrior* method, followed by the *All* methods. This shows that the *SrcPrior* method would be a valid choice to transfer knowledge, since no source data is needed, except for the pre-trained network. In addition to that, this method is only evaluated on the target set, instead of both sets, and gives an increase in performance for both transfers.
The difference between semi-supervised and supervised seems to be that in the semi-supervised method, the performance increase realised by the transfer methods is relatively larger than in the supervised approach. However, the overall performance in the supervised approach is higher. Overall, the *Prior* method of which the hyperparameters are optimised on dataset B and the *Pred* method show an increase over both transfers, for both the semi-supervised and supervised approach. All other methods do not improve on the baselines for both transfers. The advantage of these methods is that they also do not need the source data.

## 7.5 Limitations and Suggestions for Future Research

When using machine learning for text classification, a large and increasing number of models is available. Only a small number of models is tested in this study. More research can be done on the performance of other machine learning models on recall text classification problems, especially e-discovery. Besides that, the tuning of the (hyper)parameters of the models can be tested more thoroughly. Hereby computational power and resources should be taken into account. In case more resources would have been available, such as a GPU, the neural networks could be optimised more thoroughly, especially the sequential neural networks. The number of hyperparameters and their ranges could be extended, to allow for more complex modelling.

Also, the data might need to be compromised less. The cut-off ratios that are used in this study might be relaxed or made abundant if more memory is available. This study shows that rather simple sequential models do not perform better than an optimised MLP model. However, this could be because of the lower complexity of the tested sequential models. If resources allowed for further optimisation, they may outperform the MLP at one point. If the hyperparameters are evaluated more extensively, the optimal model might have better results. However, one should be cautious to not overfit the hyperparameters on the datasets.

If more cases would be available, specifically the same type of case, it can be evaluated whether this improves the transfer. By generalising over the same type of case, it could be that due to the similarity of the domain, the positive influence of transferred knowledge can be enlarged.

When preparing a dataset containing fraudulent cases for classification, there is an imbalanced class problem. To gather more data, resampling methods might be used. Many of the current methods focus on visual resampling or the resampling of numerical data. Experiments can be done by translating emails back and forth, as well as substituting some words for synonyms, to resample textual data. However, there is no defined approach to apply any of these techniques. Aside from the resampling methods, the responsive emails could be oversampled or the other class undersampled.

During this study, only two representative datasets were available. This means that the results are not generalizable outcomes for the e-discovery classification problem. The performance of the vectorisation methods, classifying algorithms and model architectures are indicators for these specific situations. They could be indicative of an optimal classifier for e-discovery. However, these findings would have to be reproduced with multiple different datasets to come to a possible optimal approach that might be a generic solution for the domain. The same is true for the transfer approach, the methods showing the most potential for these transfers might not be the same when other datasets are used.

The two datasets contained mainly Dutch emails. The results of having datasets that differ in language or form of fraud could also greatly influence the classification and transfer process. By developing a more elaborate overview of how the language and case of fraud influence the classifier settings and gathered knowledge, the chances of finding a generic e-discovery (transfer) model can be increased.

In this study, the available metadata of emails is considered in the classification process. Apart from this already available metadata, more metadata could be generated. Examples hereof are using POS-tags (part-of-speech-tags) or flagging based upon occurrences of certain terms or constructs. There is only limited research available on the creation of this extra data for e-discovery, to which additional research would be beneficiary. Alongside that, the parameters of the models could be optimised on the metadata, to show the possibilities of using the metadata for classifying the emails.

For some of the transfer learning methods, different parameters and settings could be evaluated, which have not been tested due to time restrictions. DANN could be tested with a deep network, instead of the shallow one used. Hereby the layers forming the latent space and the classifiers can be extended by one or more layers. Furthermore, TrAdaBoost can be evaluated using an SVM to classify the instances. The method was evaluated using this model in the original paper. Future research could be done if the type of classifier influences the effect of using TrAdaBoost.

Aside from the applied transfer learning methods, other transfer methods could be tested. This study evaluated only one method per domain adaptation approach. Therefore, more methods are available per approach, which may perform better. This could be older models, such as EasyAdapt [37] or Structural Correspondence Learning [18], or other recent methods that are more aligned to the state-of-the-art, such as Adversarial Discrimination Domain Adaptation (ADDA) [106]. For some domain adaptation methods, such as ADDA, no studies have been found where the method is applied to a text classification problem. This could also be interesting for future research. Aside from that, methods from other transfer learning fields, such as multi-

source learning, can be applied to e-discovery.
Furthermore, there could be opportunities for active learning, based on Technology-Assisted Review. Once a small subset is reviewed on relevance, this feedback could be implemented back into the model to gradually improve the performance. The Continuous Active Learning method of Cormack and Grossman [33] can used as the baseline. Aside from this baseline, there are not many other studies introducing active learning methods for e-discovery. The dissertation of Wang [109] on active transfer learning can be used. The use of active learning might create opportunities for improving both the performance of the classifier and the transfer method.
Considering XAI, no studies have yet explored the use of XAI in e-discovery. It can be interesting to find out if hidden patterns could be uncovered, gaining more insight into what makes the model classifies certain emails as responsive. Next to that, an experiment can be done as to how well a proxy model would perform compared to the more complex machine learning classifiers.

# 8 Conclusion

The aim of this thesis is **to optimise a classification model and create more insight into the transferability between cases while considering the implications of applying explainable AI in the legal domain**. This aim should be achieved by answering the research questions, based upon the results.

*RQ1. To what extent do word embeddings increase the performance of an e-discovery model compared to the use of tf-idf vectors?*
Three types of word embeddings are evaluated in this study, Word2Vec based upon Continuous Bag of Words, Word2Vec based on Skip-grams and bi-directional Flair embeddings. The optimal model makes use of tf-idf vectorisation, instead of word embeddings. Over the seven models that are evaluated, tf-idf vectorisation performs best on the Support Vector Machine with stochastic gradient descent training, Gradient Boosting Machine and Long Short-Term Memory. However, on the Naive Bayes, Support Vector Machine, Random Forest and Recurrent Convolutional Neural Network it is outperformed by word embeddings. On the last three, Word2Vec based on Skip-grams showed the best performance. Of the word embedding types, Word2Vec based on Skip-grams showed the most potential, followed by Flair. The Word2Vec embedding based on Continuous Bag of Words had an overall weak performance.

*RQ2. Which machine learning model that is currently used in e-discovery performs best on the provided cases?*
The Multilayer Perceptron combined with tf-idf vectorisation showed the best performance on both datasets. Furthermore, this model could also be combined with Word2Vec based on Skip-grams or Flair to perform well, since these were the second and third best options for both datasets. If the Multilayer Perceptron would not have been included, the Support Vector Machine with stochastic gradient descent training performs best on either model, which is the current baseline for e-discovery. The performance of the sequential neural networks, the Long Short-Term Memory and the Recurrent Convolutional Neural Network, was firmly influenced by the limited amount of resources available. Therefore, only a simple form of these networks has been evaluated.

*RQ3. How can the performance of the model be improved using the available metadata?*
The only metadata that showed positive results when added is the subject, which has a significant increase in performance for one dataset and similar results with a lower standard deviation for the other. All other metadata subsets found by forward and backward feature selection did not improve the performance. This could be caused by the parameters of the model, which have been optimised solely on tokens. To improve the performance of the e-discovery model using metadata features, further research has to be done on how the most usable features can be engineered and on the fine-tuning of a model to get suitable information of them.

*RQ4. To what extent is the knowledge between the provided e-discovery cases transferable?*
The answer to his questions is covered by the three sub-questions.

*SQ4.1 Is the knowledge between the cases transferable?*
The learning bounds hold, showing that there is a chance for positive transfer of knowledge. The results are symmetrical, thus the learning bounds hold in both directions.

*SQ4.2 To what extent can an unlabelled dataset benefit from a form of transfer learning?*
When the target set is unlabelled, it concerns an unsupervised transfer approach. The baseline for this approach is the *SrcPrior* method, which uses a pre-trained source model to classify the

target set. The domain adaptation methods *TgtPrior*, Transfer Component Analysis (*TCA*) and the Domain-Adversarial Neural Network (*DANN*) are applied in addition to this baseline method. Except for *TCA* using 1-Nearest Neighbours, all methods outperformed the baseline on both transfers. This shows that an unlabelled dataset can indeed benefit from a form of transfer learning. Hereby *TCA*, using a Multilayer Perceptron with hidden layers using the ReLU activation function, and *DANN* considerably increased the baseline results for both transfers. However, they did so by classify most emails as responsive. The *TgtPrior* method outperformed the *SrcPrior* method in both cases, but by less than the other methods did. The *TCA* using a Multilayer Perceptron and *DANN* show potential, but might need more fine-tuning to optimise its results.

*SQ4.3 To what extent can the performance of the model be increased using a form of transfer learning?*
When part of the target set is labelled, it concerns a semi-supervised approach. In this case, the goal is to optimise the performance using a transfer method. However, the method showing the best results is dependent on the transfer. The methods that are evaluated are *Pred*, which uses the prediction of the source model on the target data as an extra feature, Transfer Adaptive Boosting (*TrAdaBoost*), *SrcPrior*, *TgtPrior* and in case of the supervised approach, *All*, which trains and evaluates a model on the concatenation of both datasets.
When transferring from a less balanced set to a more balanced set, *TgtPrior* has the best performance and *SrcPrior* the second-best. In case the transfer is from the more balanced set to the less balanced set, the *TrAdaBoost* method with the source model architecture performs best, followed by the *SrcPrior* and *TgtPrior* methods. Hereby, the model architecture with more neurons and the ReLU activation function outperforms the one with fewer neurons and the tanh activation function in every case.
For the supervised setting, the results are similar. The performances in the supervised setting show a smaller increase than in the semi-supervised setting. The $F_2$-scores of the *All* method seem to be an average of the performances of *TgtOnly* for both datasets, showing a performance increase for the less balanced set and a decrease for the more balanced set. For the latter set, the $F_2$-scores are higher overall, due to a larger number of responsive emails in the dataset itself. Furthermore, the effect of the transfer seems to be dependent mostly on the quality of the source set. In case the source set is more balanced, the target set can benefit from this. If this is the other way around, the observed increases in performance are only marginal. The only methods outperforming the baselines for both transfers are the *Prior* method using the model architecture optimised on the more balanced dataset and the *Pred* method. These would be good baselines to use for future experiments, while they also do not need any source data, except for the pre-trained networks.

*RQ5. To what extent could interpretability serve the use of machine learning in legal technology?*
If a model becomes more explainable and its algorithms more transparent, multiple positive changes can occur. It can help verify that the system is not biased and complies with legislation, as well as help gain insight into hidden pattern and development possibilities. First, the possibilities of developing interpretable models for e-discovery should be evaluated. Currently, no studies can be found about applying XAI to e-discovery. Once the performance of interpretable models is comparable to complex, well-performing classifiers, the opportunities for using machine learning in more legal cases might arise. However, this will probably be in the long run.

## 8.1 Contributions

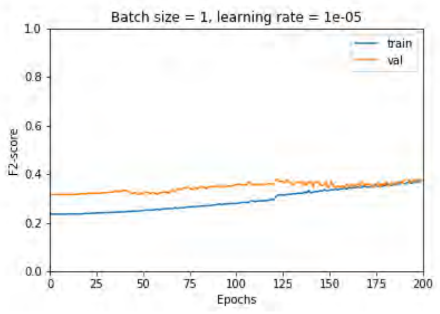
This thesis contains multiple contributions. First off, it adds to the existing research in the e-discovery domain. This thesis contributes by showing the results of four vectorisation methods and seven classification models. We show that combining a Multilayer Perceptron with tf-idf vectorisation is the best classifier for the two available datasets, beating the current baseline in e-discovery, the Support Vector Machine with tf-idf vectorisation.

Furthermore, no other study has been found that applies transfer learning to e-discovery. In this study, three approaches to transfer learning are evaluated: unsupervised, semi-supervised and supervised transfer learning. We show that applying transfer learning in e-discovery can be beneficial, independent of whether target labels are available.

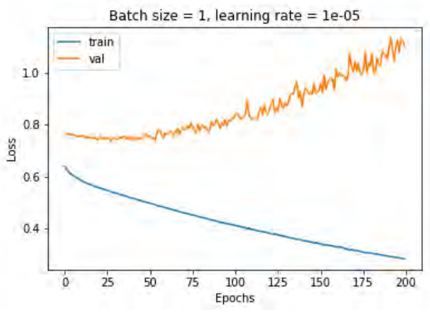
Lastly, we contribute by summarising the current possibilities of machine learning and artificial intelligence in legal technology, and how explainable AI might enlarge these opportunities.

# Appendices

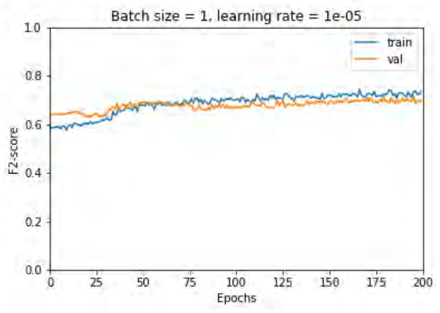
# Appendix A: Visuals epochs



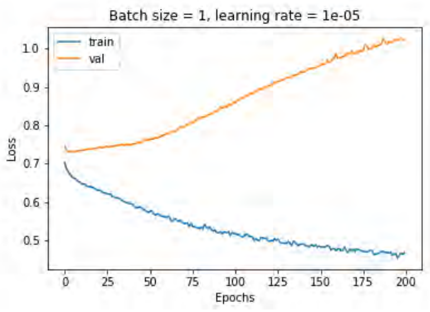
(a) Dataset A with W2V-CBOW performance



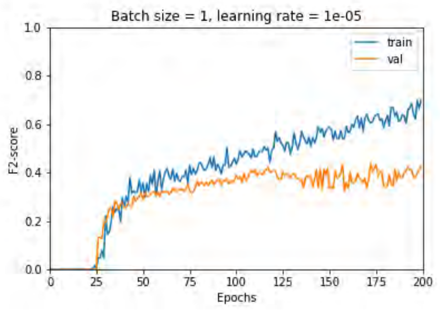
(b) Dataset A with W2V-CBOW loss



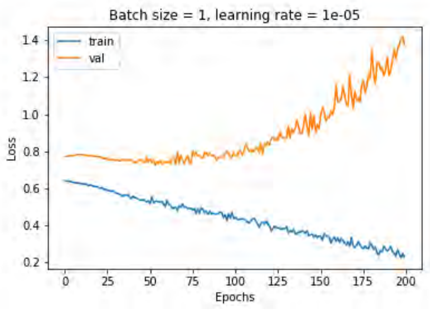
(c) Dataset B with W2V-CBOW performance



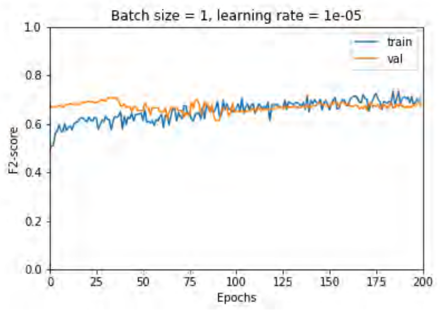
(d) Dataset B with W2V-CBOW loss



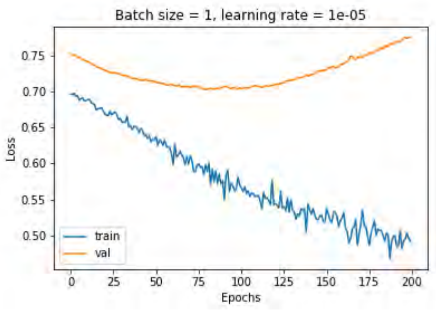
(e) Dataset A with W2V-SKIP performance



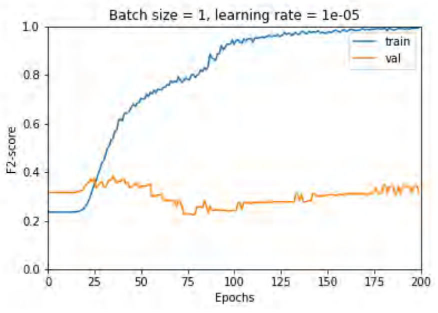
(f) Dataset A with W2V-SKIP loss



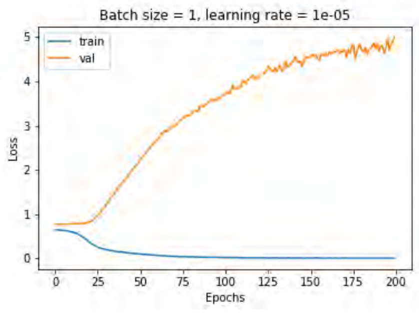
(g) Dataset B with W2V-SKIP performance



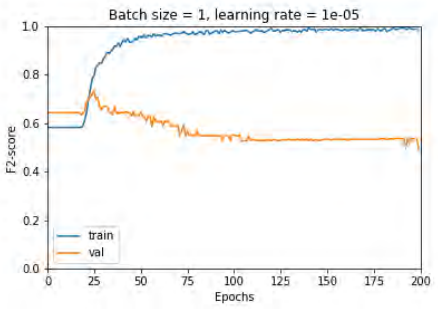
(h) Dataset B with W2V-SKIP loss



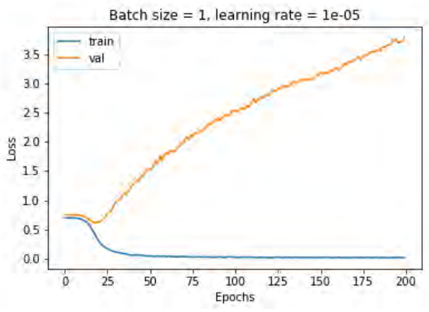
(i) Dataset A with tf-idf performance



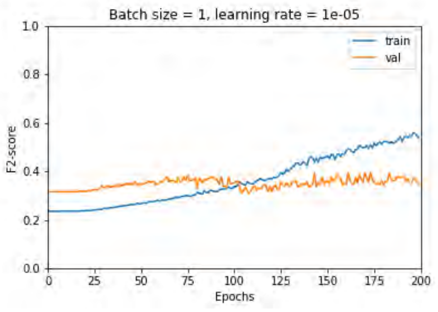
(j) Dataset A with tf-idf loss



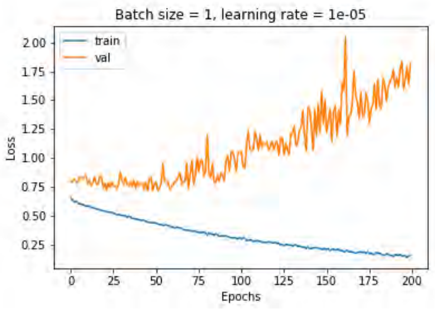
(k) Dataset B with tf-idf performance



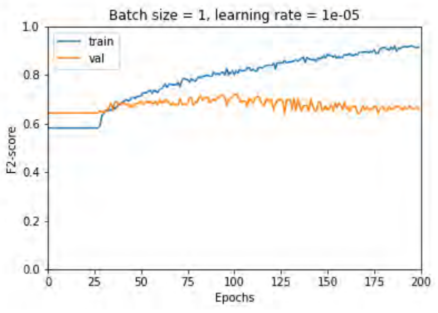
(l) Dataset B with tf-idf loss



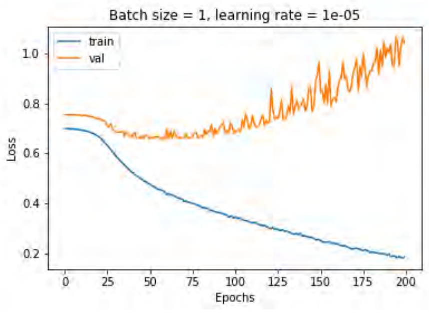
(m) Dataset A with FLAIR performance



(n) Dataset A with FLAIR loss

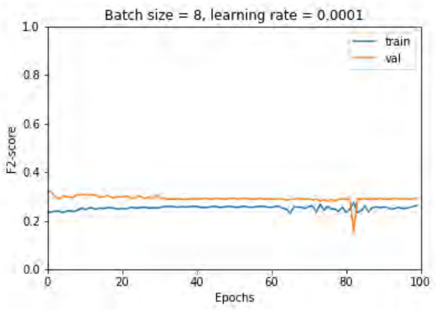


(o) Dataset B with FLAIR performance

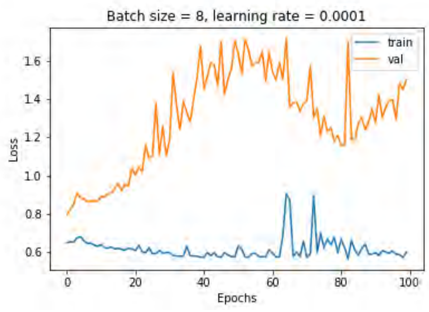


(p) Dataset B with FLAIR loss

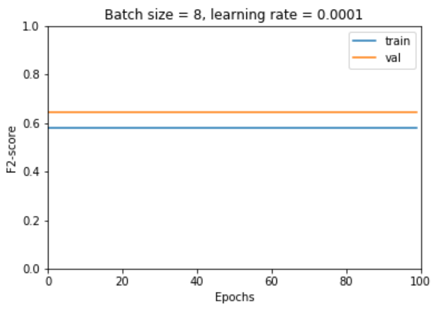
Figure 13: Visuals of training and validation performance and loss for the MLP variations.



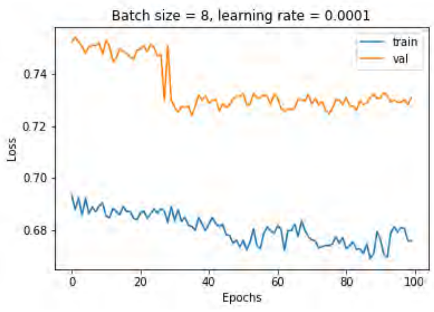
(a) Dataset A with W2V-CBOW performance



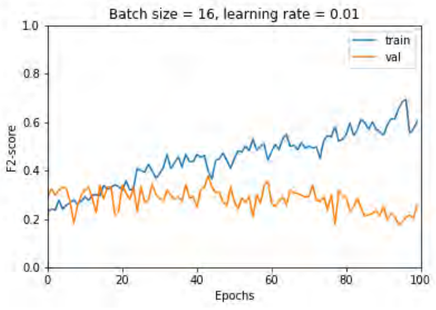
(b) Dataset A with W2V-CBOW loss



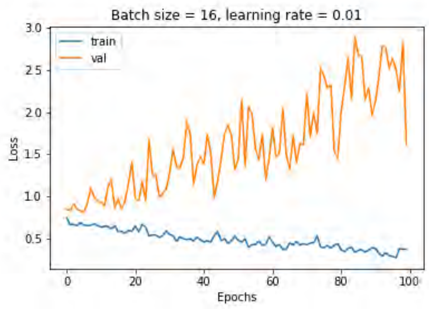
(c) Dataset B with W2V-CBOW performance



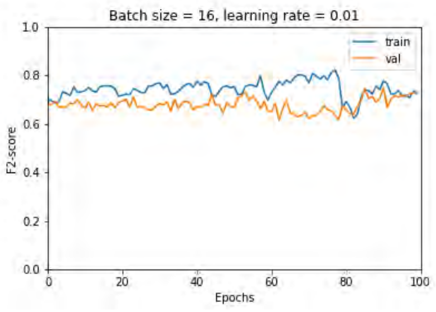
(d) Dataset B with W2V-CBOW loss



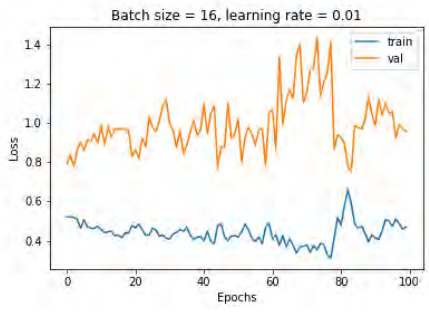
(e) Dataset A with W2V-SKIP performance



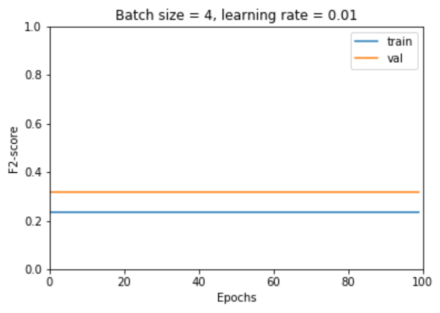
(f) Dataset A with W2V-SKIP loss



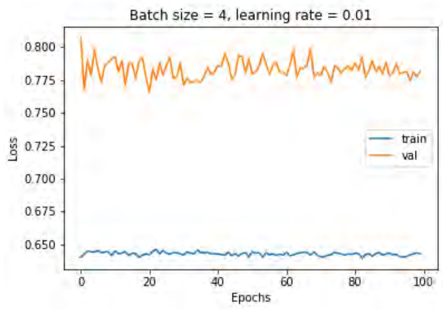
(g) Dataset B with W2V-SKIP performance



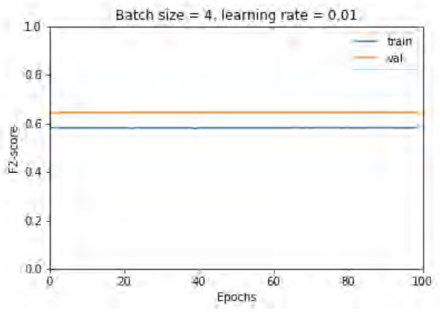
(h) Dataset B with W2V-SKIP loss



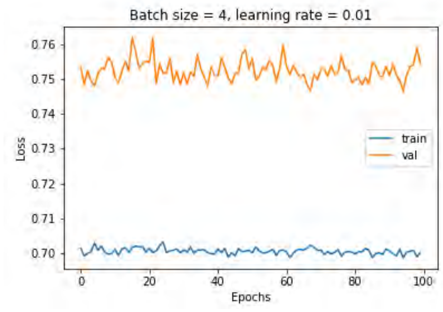
(i) Dataset A with tf-idf performance



(j) Dataset A with tf-idf loss

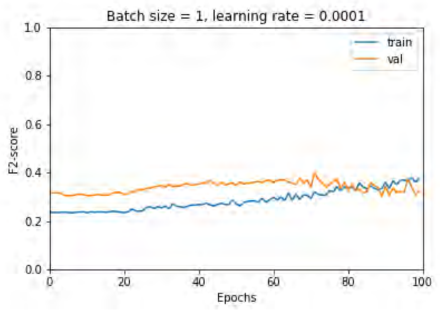


(k) Dataset B with tf-idf performance

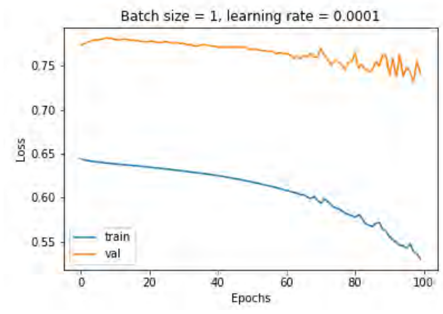


(l) Dataset B with tf-idf loss

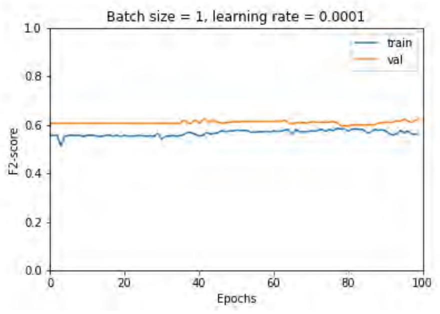
Figure 14: Visuals of training and validation performance and loss for the LSTM variations.



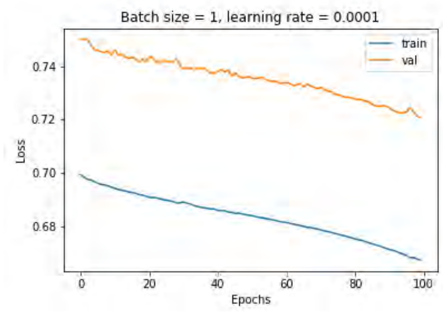
(a) Dataset A with W2V-CBOW performance



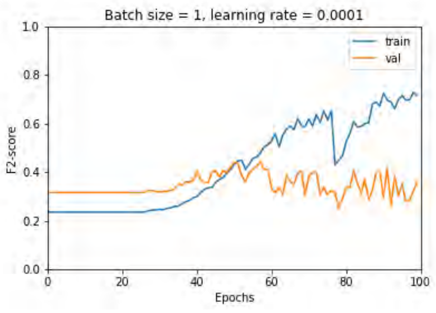
(b) Dataset A with W2V-CBOW loss



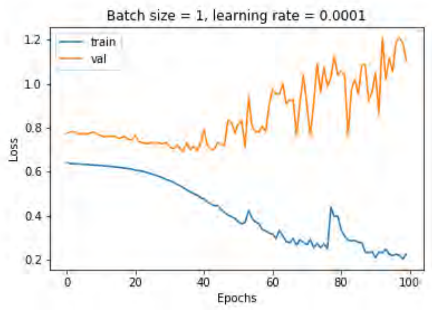
(c) Dataset B with W2V-CBOW performance



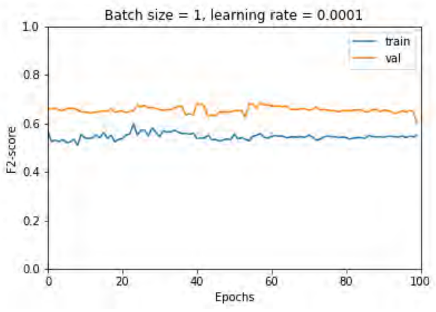
(d) Dataset B with W2V-CBOW loss



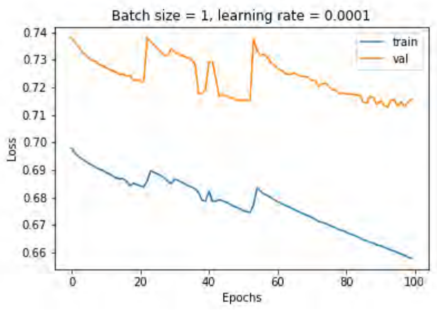
(e) Dataset A with W2V-SKIP performance



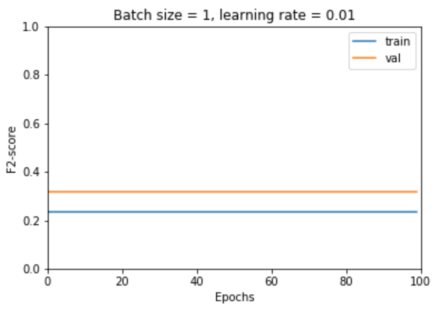
(f) Dataset A with W2V-SKIP loss



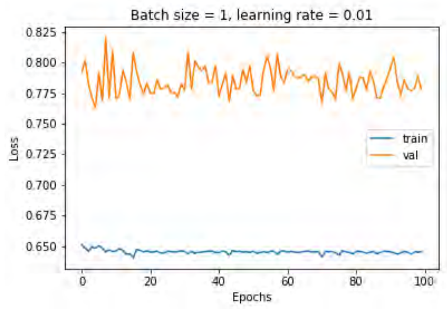
(g) Dataset B with W2V-SKIP performance



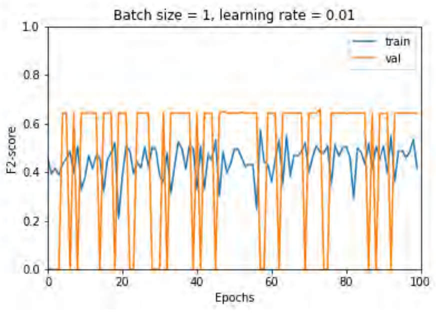
(h) Dataset B with W2V-SKIP loss



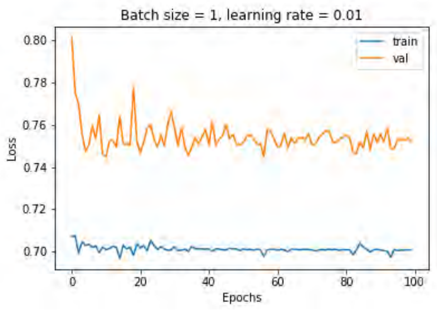
(i) Dataset A with tf-idf performance



(j) Dataset A with tf-idf loss



(k) Dataset B with tf-idf performance



(l) Dataset B with tf-idf loss

Figure 15: Visuals of training and validation performance and loss for the RCNN variations.

## Appendix B: Misclassification Matrices

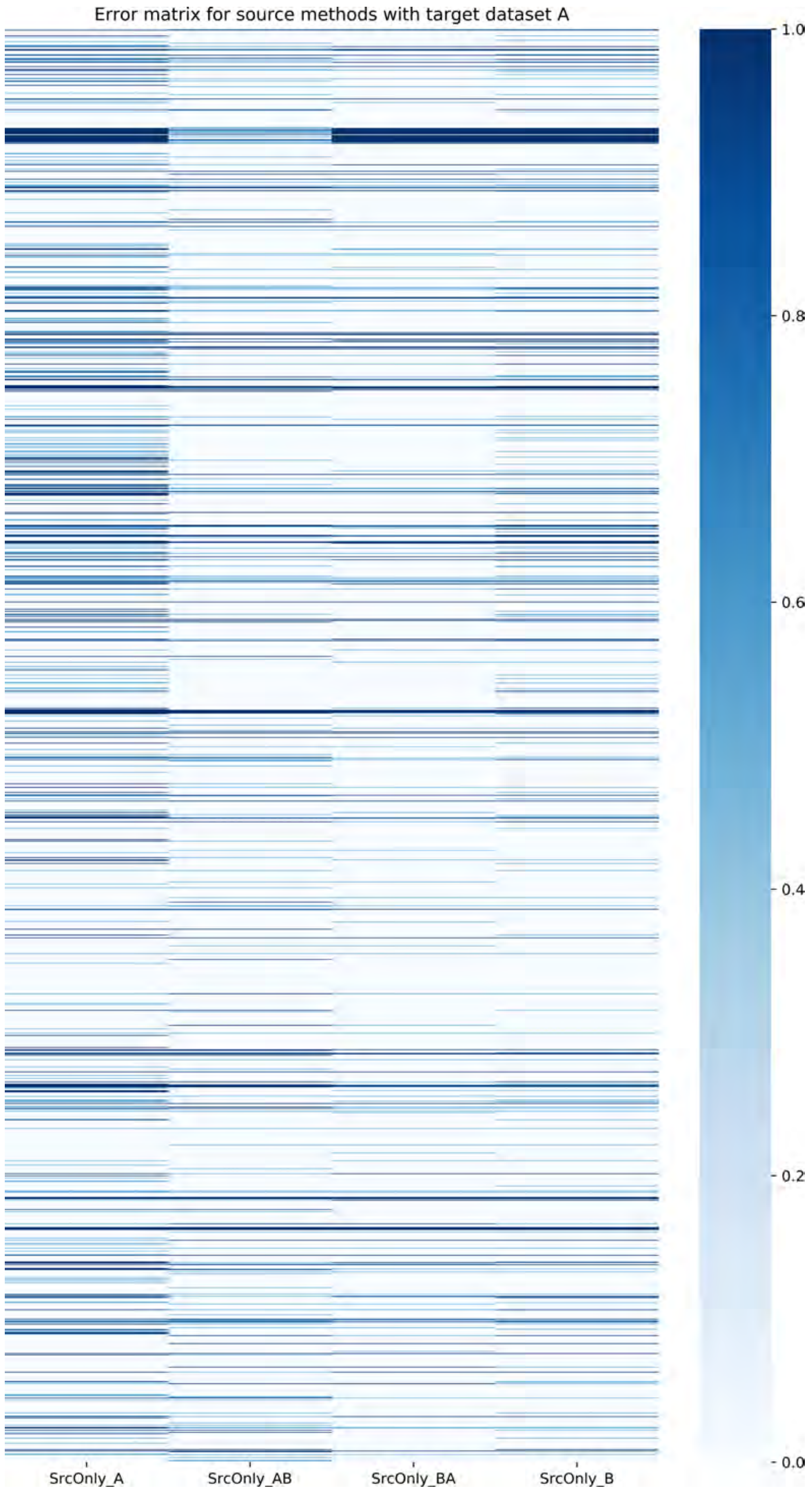


Figure 16: Error matrix showing the classification (0 is correct, 1 is misclassified) per email over the different model architectures evaluated on dataset A.

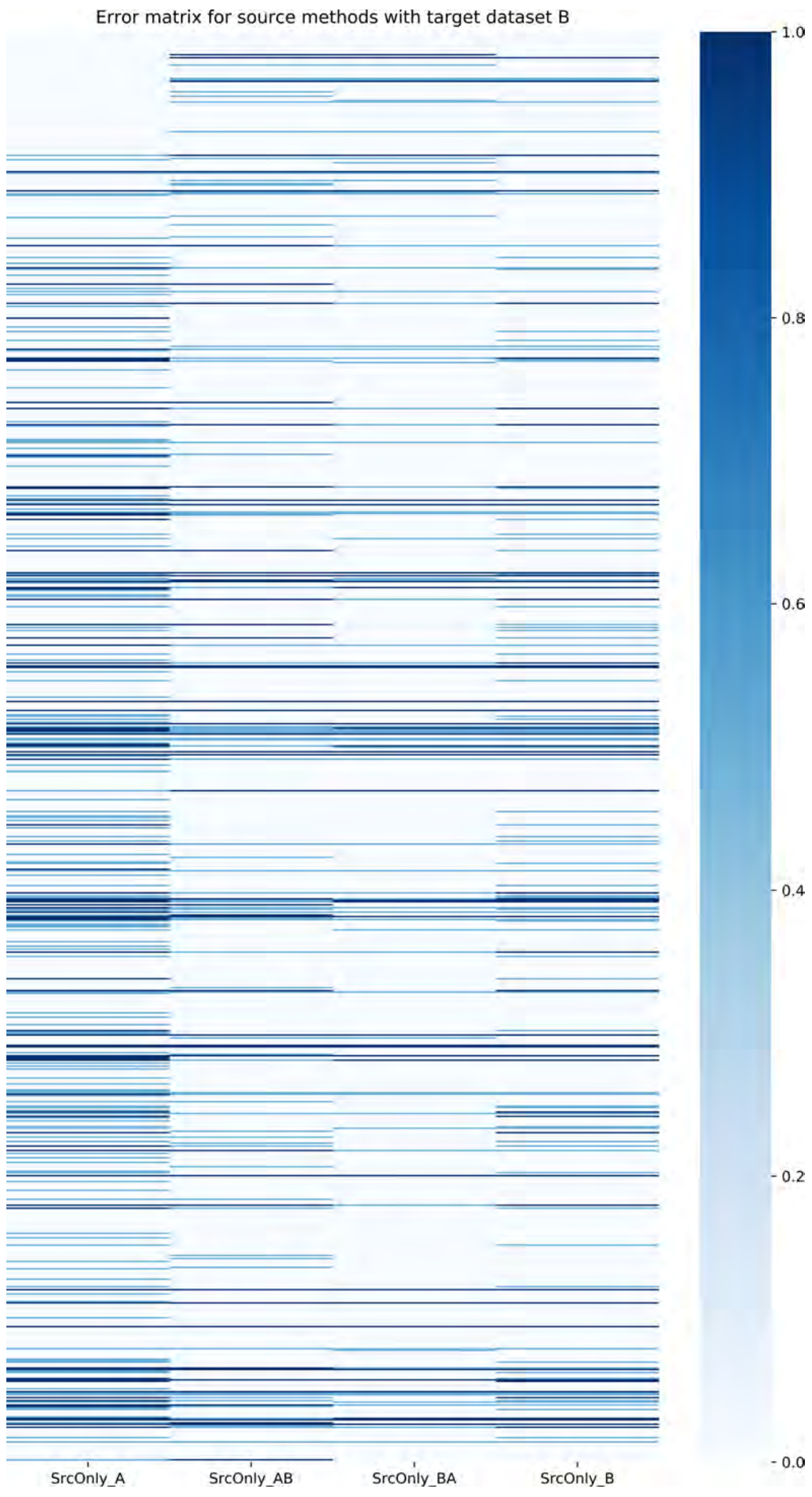


Figure 17: Error matrix showing the classification (0 is correct, 1 is misclassified) per email over the different model architectures evaluated on dataset B.

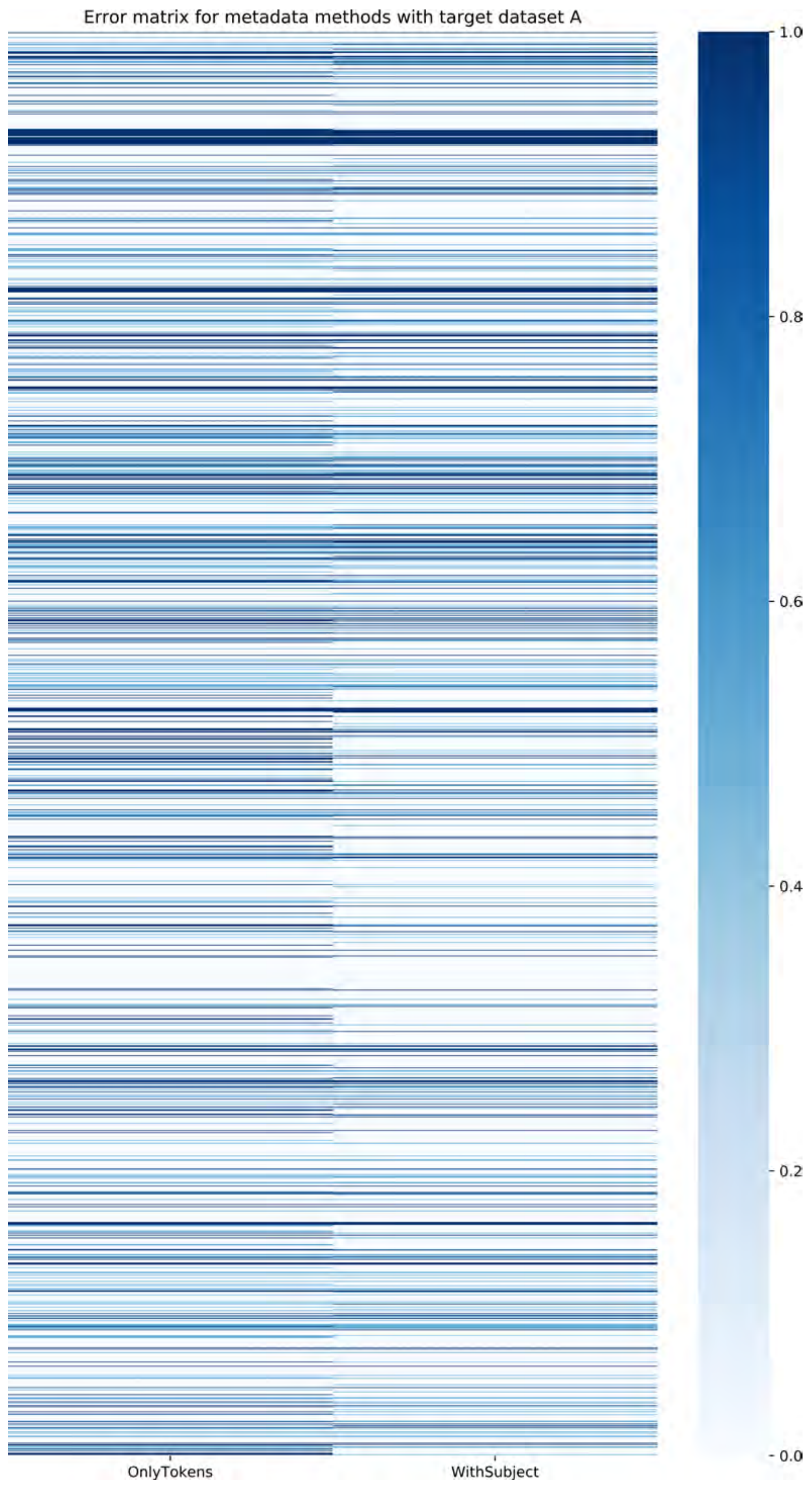


Figure 18: Error matrix showing the classification (0 is correct, 1 is misclassified) per email with and without the subject tokens added on dataset A.

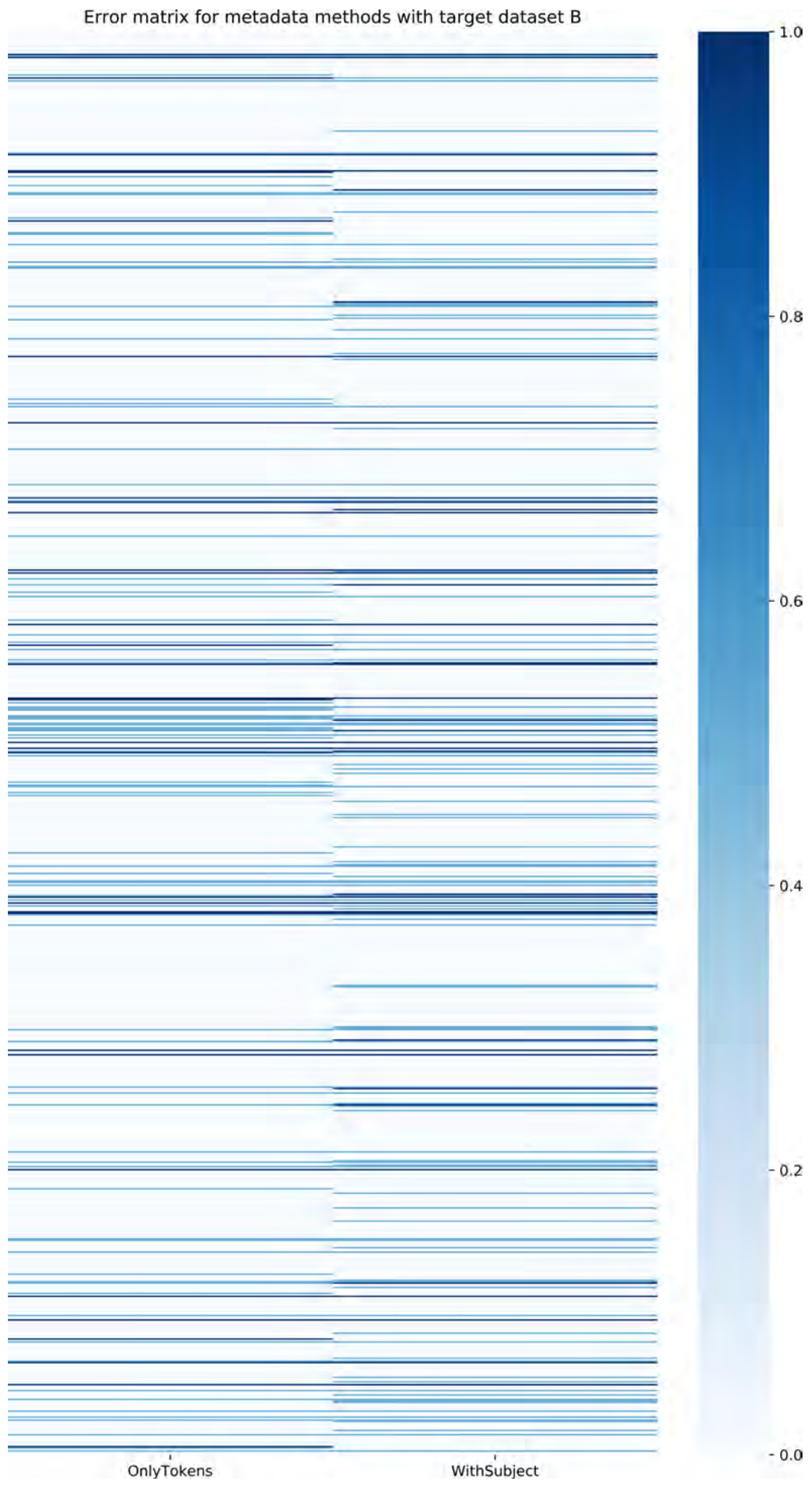


Figure 19: Error matrix showing the classification (0 is correct, 1 is misclassified) per email with and without the subject tokens added on dataset B.

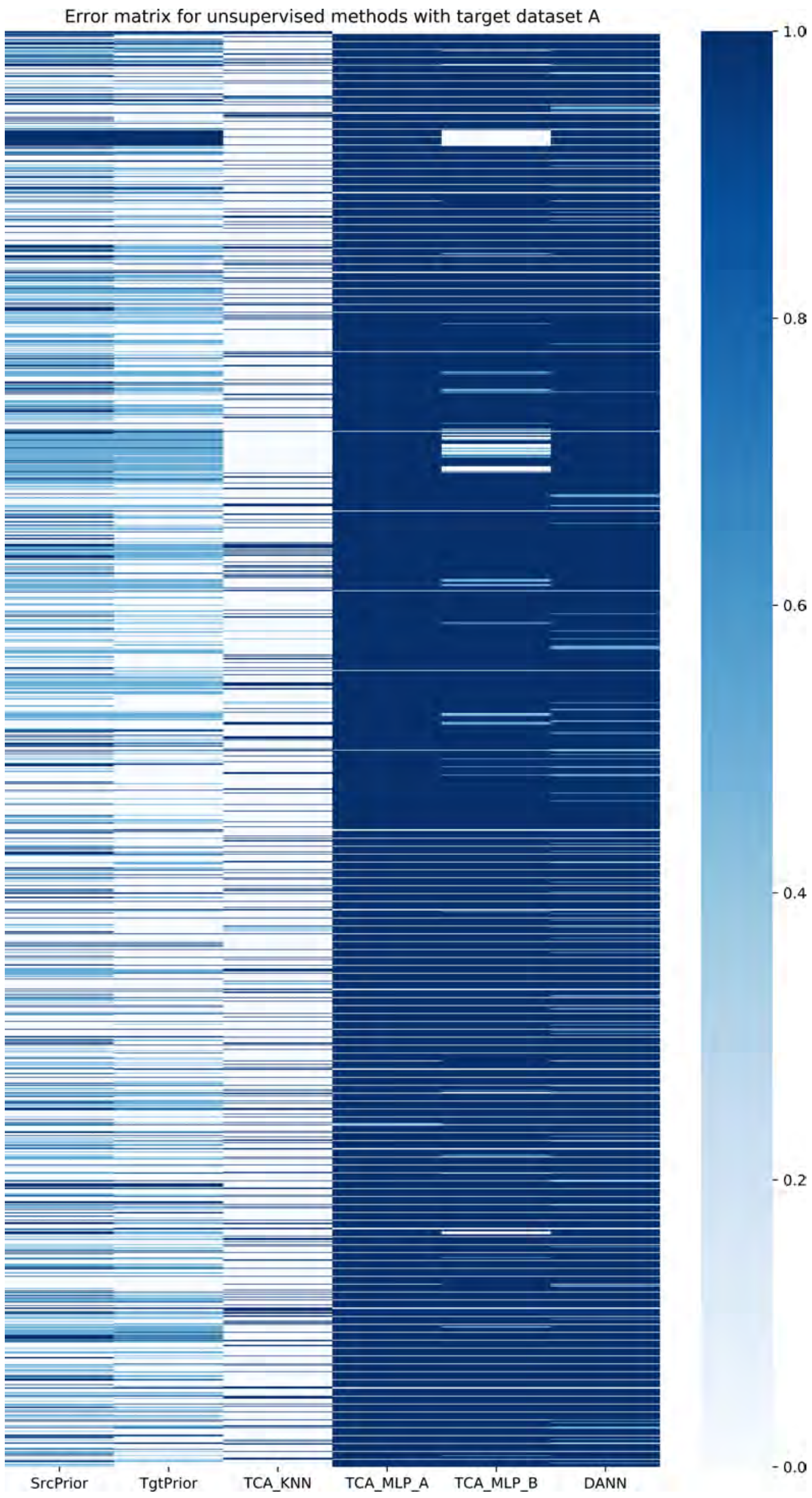


Figure 20: Error matrix showing the classification (0 is correct, 1 is misclassified) per email for unsupervised transfer learning methods with dataset A as target dataset.

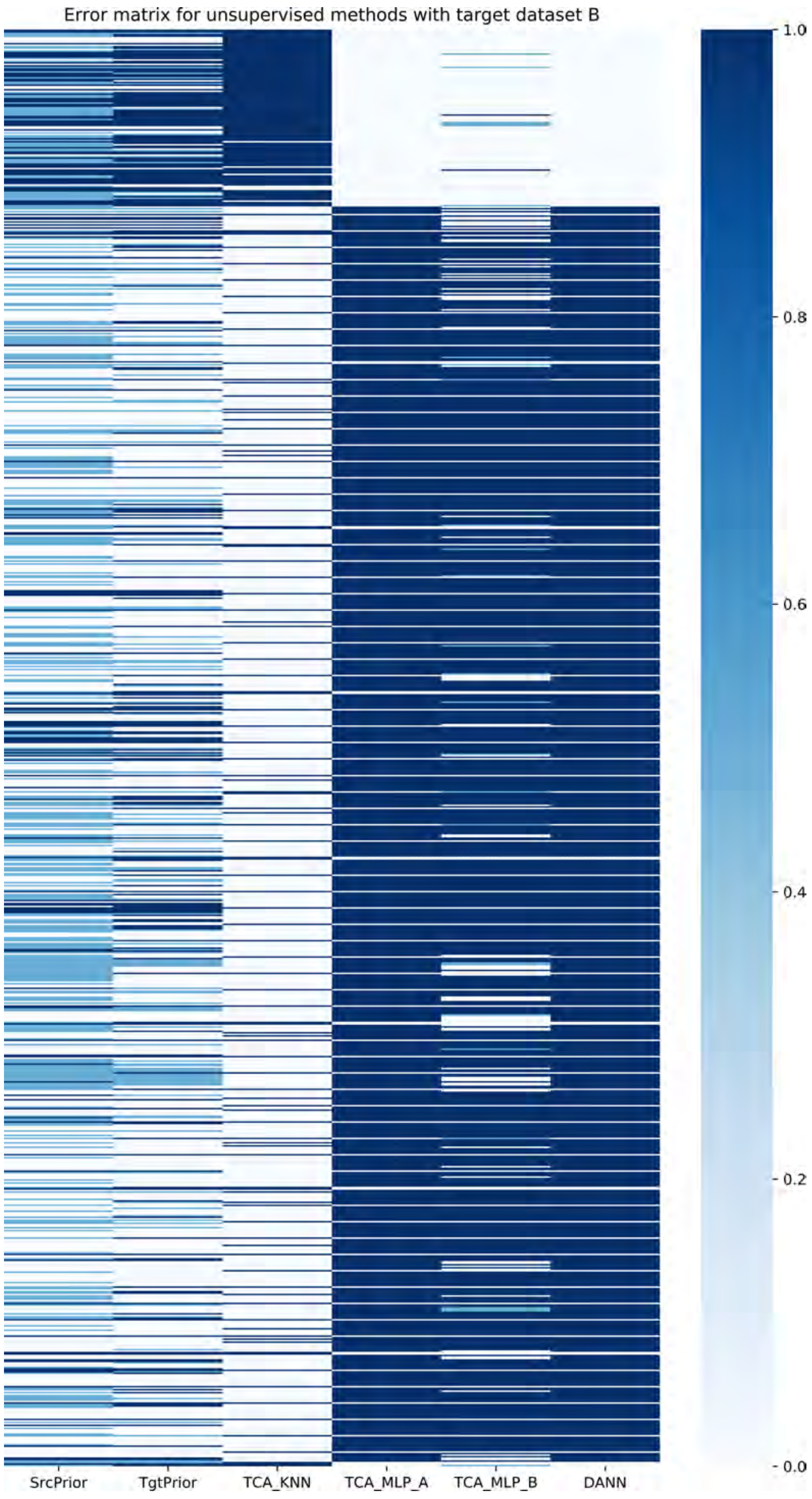


Figure 21: Error matrix showing the classification (0 is correct, 1 is misclassified) per email for unsupervised transfer learning methods with dataset B as target dataset.

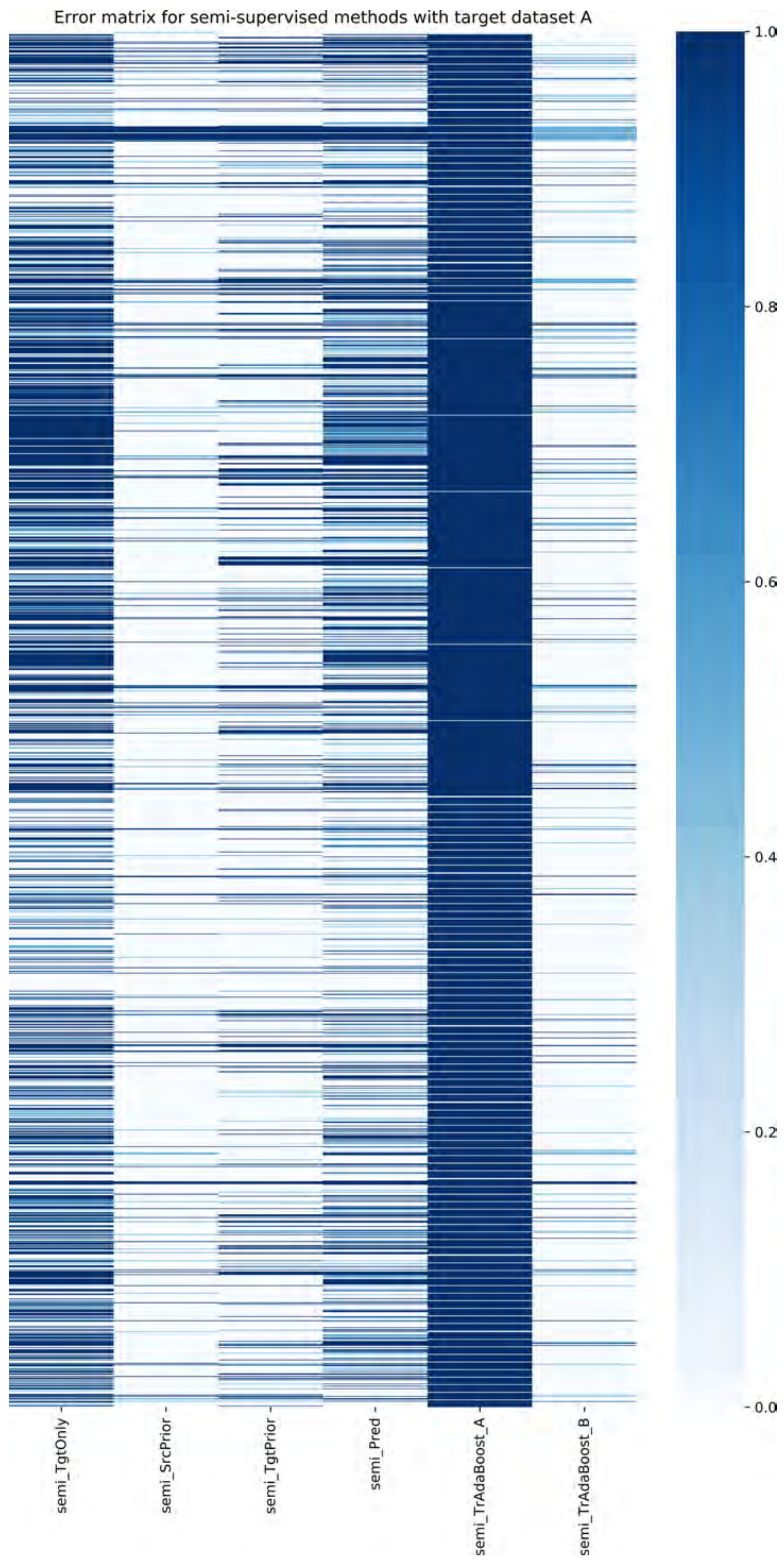


Figure 22: Error matrix showing the classification (0 is correct, 1 is misclassified) per email for semi-supervised transfer learning methods with dataset A as target dataset.

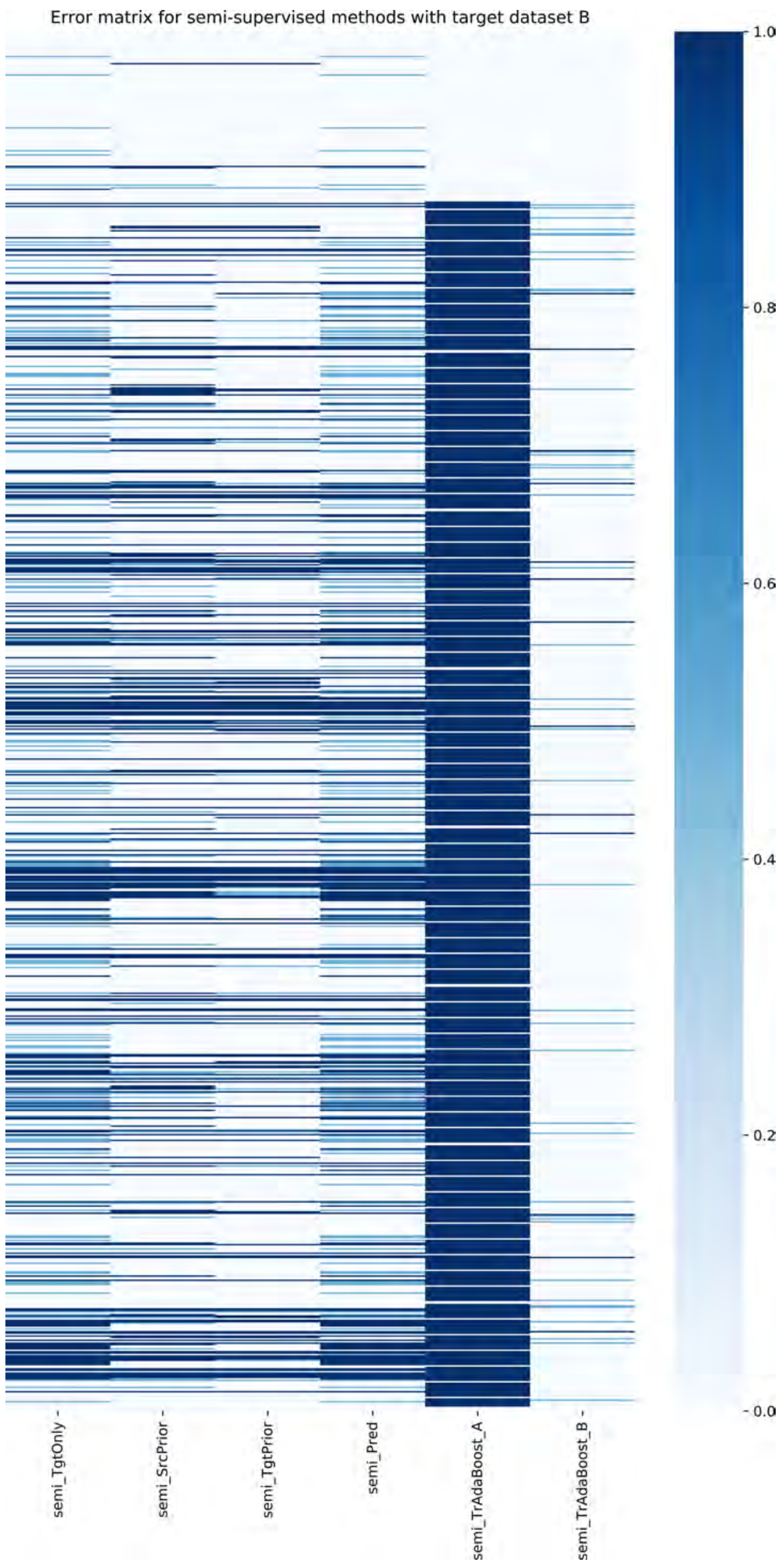


Figure 23: Error matrix showing the classification (0 is correct, 1 is misclassified) per email for semi-supervised transfer learning methods with dataset B as target dataset.

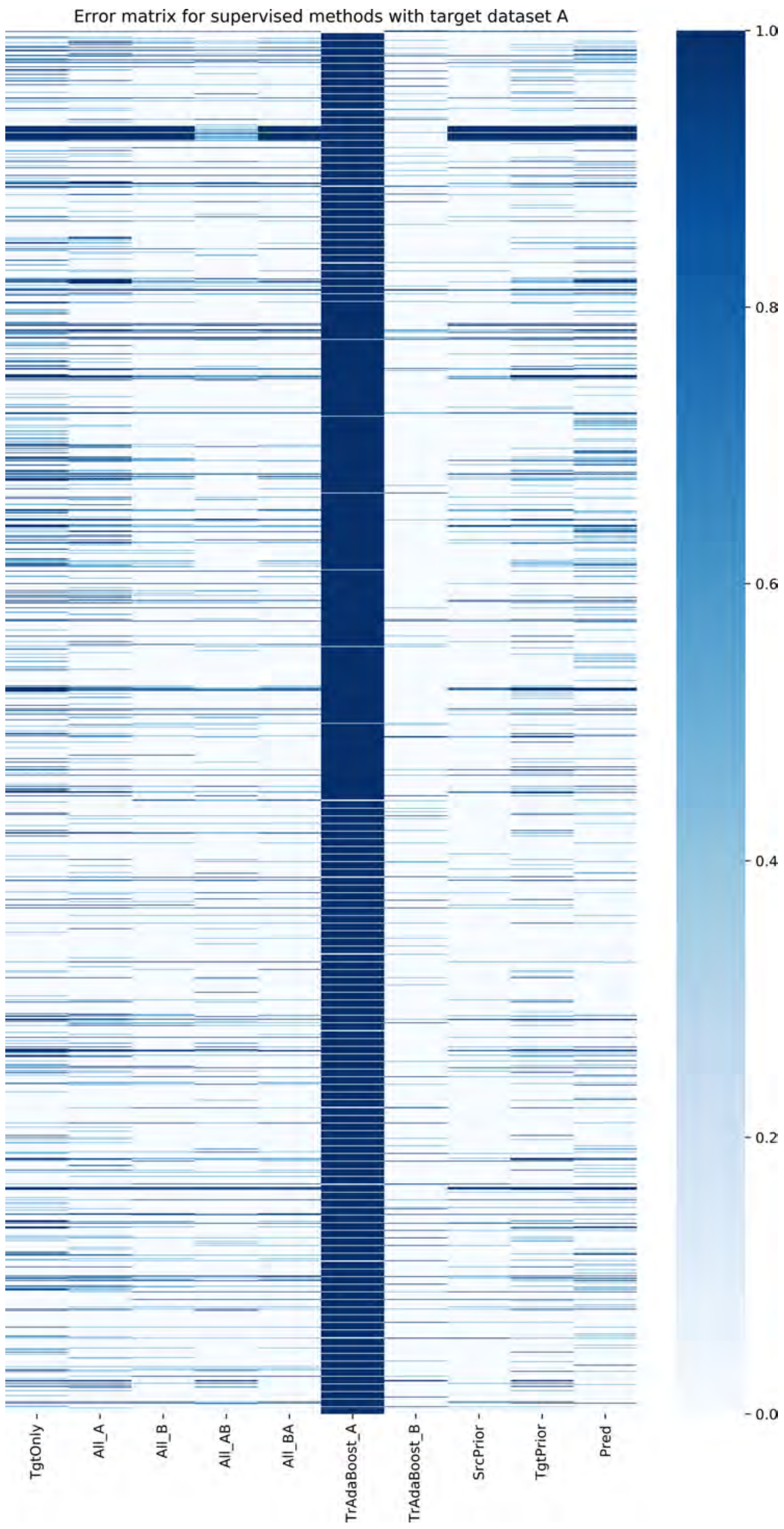


Figure 24: Error matrix showing the classification (0 is correct, 1 is misclassified) per email for supervised transfer learning methods with dataset A as target dataset.

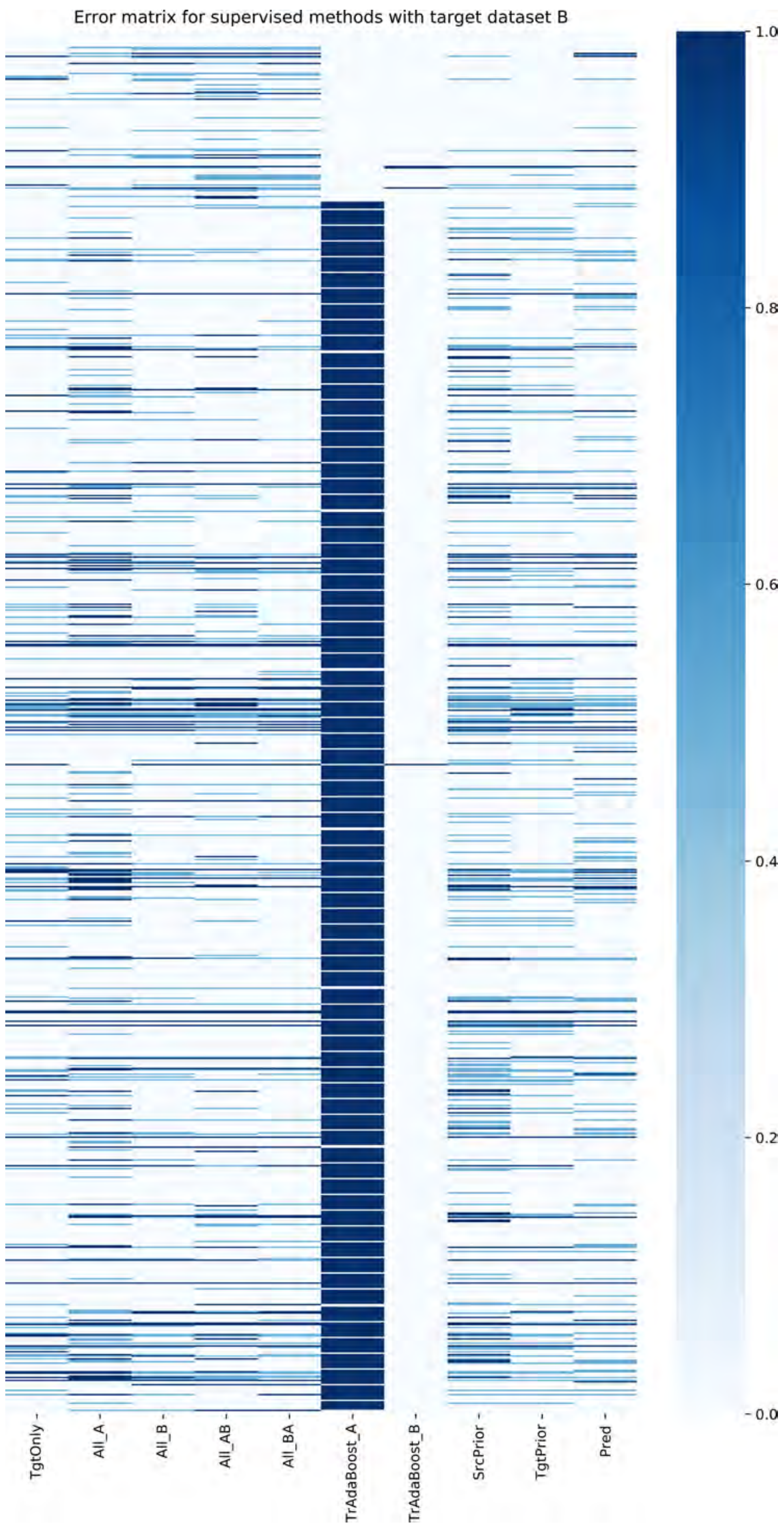


Figure 25: Error matrix showing the classification (0 is correct, 1 is misclassified) per email for supervised transfer learning methods with dataset B as target dataset.

## References

[1] Martìn Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* 2015. URL: https://www.tensorflow.org/.

[2] Amina Adadi and Mohammed Berrada. "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)". In: *IEEE Access* 6 (2018), pp. 52138–52160. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2870052. URL: https://ieeexplore.ieee.org/document/8466590/.

[3] Hana Ajakan et al. "Domain-Adversarial Neural Networks". In: (2014). arXiv: 1412.4446. URL: http://arxiv.org/abs/1412.4446.

[4] Alan Akbik et al. "FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP". In: *Proc. 2019 Conf. North* (2019), pp. 54–59. DOI: 10.18653/v1/N19-4010. URL: http://aclweb.org/anthology/N19-4010.

[5] Leila Arras et al. ""What is relevant in a text document?": An interpretable machine learning approach". In: *PLoS One* 12.8 (Aug. 2017). Ed. by Grigori Sidorov. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0181142. URL: https://dx.plos.org/10.1371/journal.pone.0181142.

[6] Hosein Azarbonyad, Robert Sim, and Ryen W. White. "Domain Adaptation for Commitment Detection in Email". In: (2019), pp. 672–680. DOI: 10.1145/3289600.3290984.

[7] Hosein Azarbonyad et al. "Words are Malleable: Computing Semantic Shifts in Political and Media Discourse". In: 3 (Nov. 2017). arXiv: 1711.05603. URL: http://arxiv.org/abs/1711.05603.

[8] Thomas I. Barnett. "AWAY WITH WORDS: The Myths and Misnomers of Conventional Search Strategies and the Search for Meaning in eDiscovery". In: *DESI 2015.* 2015. URL: http://users.umiacs.umd.edu/%7B~%7Doard/desi6/papers/Barnett.pdf.

[9] Thomas I. Barnett and Svetlana Godjevac. "Faster, better, cheaper legal document review, pipe dream or reality?" In: *DESI 2011.* 2011, pp. 1–16. URL: http://legacydirs.umiacs.umd.edu/%7B~%7Doard/desi4/papers/barnett3.pdf.

[10] Deborah Baron, Angela Bunting, and Brian J Krupczak. "Turning Back Time : The Application of Predictive Technology to Big Data". In: *DESI 2013* (2013), pp. 1–7. URL: http://www.umiacs.umd.edu/%7B~%7Doard/desi5/%7B%5C#%7DPapers.

[11] Jason R. Baron and Jesse B. Freeman. "Cooperation, Transparency, and the Rise of Support Vector Machines in E-Discovery: Issues Raised by the Need to Classify Documents as Either Responsive or Nonresponsive". In: *DESI 2013.* 2013, pp. 1–19. URL: http://users.umiacs.umd.edu/%7B~%7Doard/desi5/additional/Baron-Jason-final.pdf.

[12] Yoshua Bengio. "Practical recommendations for gradient-based training of deep architectures". In: *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)* 7700 LECTU (2012), pp. 437–478. ISSN: 03029743. DOI: 10.1007/978-3-642-35289-8-26. arXiv: 1206.5533.

[13] Yoshua Bengio, Ian J. Goodfellow, and Aaron Courville. "Sequence Modeling : Recurrent and Recursive Nets". In: *Deep Learn.* 2015. Chap. 10, pp. 324–365.

[14] James Bergstra and Yoshua Bengio. "Random search for hyper-parameter optimization". In: *J. Mach. Learn. Res.* 13 (2012), pp. 281–305. ISSN: 15324435.

[15] James Bergstra et al. "Algorithms for hyper-parameter optimization". In: *Adv. Neural Inf. Process. Syst. 24 25th Annu. Conf. Neural Inf. Process. Syst. 2011, NIPS 2011.* 2011. ISBN: 9781618395993.

[16] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Ed. by Michael I. Jordan, Jon Kleinberg, and Bernhard Schölkopf. 1st ed. Springer-Verlag New York, 2006. ISBN: 9780387310732.

[17] John Blitzer. "Domain Adaptation - a PhD thesis". PhD thesis. University of Pennsylvania, 2007.

[18] John Blitzer, Ryan McDonald, and Fernando Pereira. "Domain adaptation with structural correspondence learning". In: *Proc. 2006 Conf. Empir. Methods Nat. Lang. Process. - EMNLP '06.* July. Morristown, NJ, USA: Association for Computational Linguistics, 2006, p. 120. ISBN: 1932432736. DOI: 10.3115/1610075.1610094. URL: http://portal.acm.org/citation.cfm?doid=1610075.1610094.

[19] John Blitzer et al. "Learning bounds for domain adaptation". In: *Adv. Neural Inf. Process. Syst. 20 - Proc. 2007 Conf.* (2009), pp. 1–12.

[20] Kendrick Boyd, Kevin H. Eng, and C. David Page. "Area under the Precision-Recall Curve: Point Estimates and Confidence Intervals". In: *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics).* Vol. 8190 LNAI. PART 3. 2013, pp. 451–466. ISBN: 9783642409936. DOI: 10.1007/978-3-642-40994-3_29. URL: http://link.springer.com/10.1007/978-3-642-40994-3%7B%5C_%7D29.

[21] Ulrik Brandes. "A faster algorithm for betweenness centrality". In: *J. Math. Sociol.* 25.2 (June 2001), pp. 163–177. ISSN: 0022-250X. DOI: 10.1080/0022250X.2001.9990249. URL: http://www.tandfonline.com/doi/abs/10.1080/0022250X.2001.9990249.

[22] Jason Brownlee. "How to Calculate McNemar's Test to Compare Two Machine Learning Classifiers". In: *Mach. Learn. Mastery* (2018), pp. 1–19. URL: https://machinelearningmastery.com/mcnemars-test-for-machine-learning/.

[23] Jason Brownlee. *How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras.* 2018. URL: https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/ (visited on 12/17/2019).

[24] William P Butterfield, Conor R Crowley, and Jeannine Kenney. "Reality Bites: Why TAR's Promises Have Yet to be Fulfilled". In: *DESI 2013.* 2013.

[25] José Camacho-Collados and Roberto Navigli. "Find the word that does not belong: A Framework for an Intrinsic Evaluation of Word Vector Representations". In: *Proc. 1st Work. Eval. Vector-sp. Represent. NLP.* Stroudsburg, PA, USA: Association for Computational Linguistics, 2016, pp. 43–50. DOI: 10.18653/v1/W16-2508. URL: http://aclweb.org/anthology/W16-2508.

[26] Lawrence Chapin, Simon Attfield, and Efeosasere Moibi Okoro. "Predictive Coding, Storytelling and God: Narrative Understanding in e-Discovery". In: *DESI 2013.* 2013, pp. 1–15.

[27] Jianbo Chen et al. "Learning to Explain: An Information-Theoretic Perspective on Model Interpretation". In: *Proc. 35th Int. Conf. Mach. Learn.* Vol. 80. Feb. 2018, pp. 883–892. arXiv: 1802.07814. URL: http://arxiv.org/abs/1802.07814.

[28] Ravikiran Chimatapu, Hani Hagras, and Andrew Starkey. "Theory and Practice of Natural Computing". In: *Theory Pract. Nat. Comput.* Springer International Publishing, 2017, pp. 3–20. ISBN: 978-3-319-71068-6. DOI: 10.1007/978-3-319-71069-3. URL: http://link.springer.com/10.1007/978-3-319-71069-3.

[29] F Chollet. *Deep Learning with Python.* Manning, 2018. ISBN: 9783958458406. URL: https://books.google.de/books?id=ouVcDwAAQBAJ.

[30] Jack G Conrad. "E-Discovery Revisited: A Broader Perspective for IR Researchers". In: *DESI 2007.* 2007, pp. 321–345.

[31] Jack G Conrad. "E-Discovery revisited: the need for artificial intelligence beyond information retrieval". In: *Artif. Intell. Law* 18.4 (Dec. 2010), pp. 321–345. ISSN: 0924-8463. DOI: 10.1007/s10506-010-9096-6. URL: http://link.springer.com/10.1007/s10506-010-9096-6.

[32] Sam Corbett-Davies and Sharad Goel. "The Measure and Mismeasure of Fairness: A Critical Review of Fair Machine Learning". In: *CoRR* (July 2018). arXiv: 1808.00023. URL: http://arxiv.org/abs/1808.00023.

[33] Gordon V. Cormack and Maura R. Grossman. "Evaluation of machine-learning protocols for technology-assisted review in electronic discovery". In: *Proc. 37th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR '14*. New York, New York, USA: ACM Press, 2014, pp. 153–162. ISBN: 9781450322577. DOI: 10.1145/2600428.2609601. URL: http://dl.acm.org/citation.cfm?doid=2600428.2609601.

[34] Gabor Csardi and Tamas Nepusz. "The igraph software package for complex network research". In: *InterJounal* Complex Sy (2006). URL: http://igraph.org.

[35] Wenyuan Dai et al. "Boosting for transfer learning". In: *Proc. 24th Int. Conf. Mach. Learn. - ICML '07*. New York, New York, USA: ACM Press, 2007, pp. 193–200. ISBN: 9781595937933. DOI: 10.1145/1273496.1273521. URL: http://portal.acm.org/citation.cfm?doid=1273496.1273521.

[36] Jeffrey Dastin. *Amazon scraps secret AI recruiting tool that showed bias against women.* San Francisco, 2018.

[37] Hal Daumé III. "Frustratingly Easy Domain Adaptation". In: (July 2009). arXiv: 0907.1815. URL: http://arxiv.org/abs/0907.1815.

[38] Jesse Davis and Mark Goadrich. "The relationship between Precision-Recall and ROC curves". In: *Proc. 23rd Int. Conf. Mach. Learn. - ICML '06*. Vol. 73. 10. New York, New York, USA: ACM Press, 2006, pp. 233–240. ISBN: 1595933832. DOI: 10.1145/1143844.1143874. arXiv: 1609.07195. URL: http://portal.acm.org/citation.cfm?doid=1143844.1143874.

[39] Oscar Day and Taghi M. Khoshgoftaar. "A survey on heterogeneous transfer learning". In: *J. Big Data* 4.1 (2017). ISSN: 21961115. DOI: 10.1186/s40537-017-0089-0.

[40] Janez Demšar. "Statistical comparisons of classifiers over multiple data sets". In: *J. Mach. Learn. Res.* 7 (2006), pp. 1–30. ISSN: 15337928.

[41] Thomas G. Dietterich. "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms". In: *Neural Comput.* 10.7 (1997), pp. 1895–1923.

[42] Patrick Doetsch, Pavel Golik, and Hermann Ney. "A comprehensive study of batch construction strategies for recurrent neural networks in MXNet". In: (May 2017). arXiv: 1705.02414. URL: http://arxiv.org/abs/1705.02414.

[43] Finale Doshi-Velez and Been Kim. "Towards A Rigorous Science of Interpretable Machine Learning". In: (2017), pp. 1–13. arXiv: 1702.08608. URL: http://arxiv.org/abs/1702.08608.

[44] Coen Drion. "Stare decisis". In: *Ned. Juristenbl.* (2014).

[45] Duke Law. *EDRM*. URL: https://www.edrm.net/ (visited on 05/15/2019).

[46] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern Recognit. Lett.* 27.8 (2006), pp. 861–874. ISSN: 01678655. DOI: 10.1016/j.patrec.2005.10.010.

[47] Tom Fawcett and Foster Provost. "Adaptive fraud detection". In: *Data Min. Knowl. Discov.* 1.3 (1997), pp. 291–316. ISSN: 13845810. DOI: 10.1023/A:1009700419189. URL: https://link.springer.com/content/pdf/10.1023%7B%5C%%7D2FA%7B%5C%%7D3A1009700419189.pdf.

[48] Basura Fernando et al. “Unsupervised Visual Domain Adaptation Using Subspace Alignment”. In: *ICCV 2013*. 2013, pp. 2960–2967.

[49] Linton C. Freeman. “Centrality in Social Networks I: Conceptual Clarification”. In: *Soc. Networks* 1.3 (Jan. 1979), pp. 215–239. ISSN: 03788733. DOI: 10.1016/0378-8733(78)90021-7. arXiv: 0112110 [cond-mat]. URL: https://linkinghub.elsevier.com/retrieve/pii/0378873378900217.

[50] Preet Gandhi. *KDnuggets Explainable Artificial Intelligence.* 2019. URL: https://www.kdnuggets.com/2019/01/explainable-ai.html (visited on 10/02/2019).

[51] Pascal Germain and Hana Ajakan. *Domain Adaptation Representation Learning Algorithm.* 2017. URL: https://github.com/GRAAL-Research/domain%7B%5C_%7Dadversarial%7B%5C_%7Dneural%7B%5C_%7Dnetwork (visited on 06/01/2020).

[52] Leilani H Gilpin et al. “Explaining Explanations: An Overview of Interpretability of Machine Learning”. In: *5th IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA 2018).* 2018. arXiv: 1806.00069. URL: http://arxiv.org/abs/1806.00069.

[53] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *J. Mach. Learn. Res.* 9 (2010), pp. 249–256. ISSN: 15324435.

[54] Ian J. Goodfellow. “Practical Methodology”. In: *Deep Learn.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. Chap. 11. DOI: 10.1007/978-3-642-66712-1_3. URL: http://link.springer.com/10.1007/978-3-642-66712-1%7B%5C_%7D3.

[55] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. “Optimization for Training Deep Models”. In: *Deep Learn.* 2016, pp. 334–376. ISBN: 9780262035613. DOI: 10.1007/978-3-319-94463-0_8. arXiv: 1411.4555v1.

[56] Isabelle Guyon and André Elisseeff. “An Introduction to Variable and Feature Selection”. In: *J. Mach. Learn. Res.* 3 (2003), pp. 1157–1182.

[57] Karen Hao and Jonathan Stray. “Can you make AI fairer than a judge ? Play our courtroom algorithm game”. In: *MIT Technol. Rev.* (2019).

[58] Tim Head et al. *Scikit-optimize.* 2018. DOI: 10.5281/zenodo.1207017.

[59] Jeff Heaton. *The Number of Hidden Layers.* 2017. URL: https://www.heatonresearch.com/2017/06/01/hidden-layers.html (visited on 11/29/2019).

[60] Thomas J. Helling, Johannes C. Scholtes, and Frank W. Takes. “A Community-Aware Approach for Identifying Node Anomalies in Complex Networks”. In: *Stud. Comput. Intell.* Vol. 812. 2019, pp. 244–255. ISBN: 9783030054106. DOI: 10.1007/978-3-030-05411-3_20. URL: http://link.springer.com/10.1007/978-3-030-05411-3%7B%5C_%7D20.

[61] Hans Henseler. “Network-based filtering for large email collections in E-Discovery”. In: *Artif. Intell. Law* 18.4 (Dec. 2010), pp. 413–430. ISSN: 0924-8463. DOI: 10.1007/s10506-010-9099-3. URL: http://link.springer.com/10.1007/s10506-010-9099-3.

[62] History.com Editors. *Enron files for bankruptcy.* 2009. URL: https://www.history.com/this-day-in-history/enron-files-for-bankruptcy (visited on 10/08/2019).

[63] Laszlo A. Jeni, Jeffrey F. Cohn, and Fernando De La Torre. “Facing Imbalanced Data–Recommendations for the Use of Performance Metrics”. In: *2013 Hum. Assoc. Conf. Affect. Comput. Intell. Interact.* IEEE, Sept. 2013, pp. 245–251. ISBN: 978-0-7695-5048-0. DOI: 10.1109/ACII.2013.47. URL: http://ieeexplore.ieee.org/document/6681438/.

[64] Amanda Jones et al. “The Role of Metadata in Machine Learning for Technology Assisted Review”. In: *DESI 2015.* 2013. 2014, pp. 1–12. URL: http://users.umiacs.umd.edu/%7B~%7Doard/desi6/papers/JonesFinal.pdf.

[65] Michael I. Jordan and Tom M. Mitchell. “Machine learning: Trends, perspectives, and prospects”. In: *Science (80-. ).* 349.6245 (2015), pp. 255–260. ISSN: 10959203. DOI: 10.1126/science.aaa8415.

[66] Jongbin Jung et al. “Simple rules for complex decisions”. Feb. 2017. URL: http://arxiv.org/abs/1702.04690.

[67] Daniel Jurafsky and James H. Martin. *Speech and language processing.* 2nd ed. Prentice-Hall Inc., 2008. ISBN: 0130950696.

[68] Bryan Klimt and Yiming Yang. “The Enron Corpus: A New Dataset for Email Classification Research”. In: *Eur. Conf. Mach. Learn. 2004.* Springer, Berlin, Heidelberg, 2004, pp. 217–226. DOI: 10.1007/978-3-540-30115-8_22. URL: http://link.springer.com/10.1007/978-3-540-30115-8%7B%5C_%7D22.

[69] Wouter M. Kouw. *LibTLDA.* 2018. URL: https://github.com/wmkouw/libTLDA.

[70] Siwei Lai et al. “Recurrent convolutional neural networks for text classification”. In: *Proc. 29th AAAI Conf. Artif. Intell.* 3 (2015), pp. 2267–2273.

[71] Jessica Leber. “The Immortal Life of the Enron E-mails”. In: *MIT Technol. Rev.* (2013). URL: https://www.technologyreview.com/s/515801/the-immortal-life-of-the-enron-e-mails/.

[72] Zachary C. Lipton. “The mythos of model interpretability”. In: *Commun. ACM* 61.10 (Sept. 2017), pp. 36–43. ISSN: 0001-0782. DOI: 10.1145/3233231. arXiv: 1606.03490. URL: https://dl.acm.org/doi/10.1145/3233231.

[73] Huan Liu and Lei Yu. “Toward integrating feature selection algorithms for classification and clustering”. In: *IEEE Trans. Knowl. Data Eng.* 17.4 (Apr. 2005), pp. 491–502. ISSN: 1041-4347. DOI: 10.1109/TKDE.2005.66. URL: http://ieeexplore.ieee.org/document/1401889/.

[74] Yong Liu et al. “An evaluation of transfer learning for classifying sales engagement emails at large scale”. In: *Proc. - 19th IEEE/ACM Int. Symp. Clust. Cloud Grid Comput. CCGrid 2019* (2019), pp. 542–548. DOI: 10.1109/CCGRID.2019.00069.

[75] Jie Lu et al. “Transfer learning using computational intelligence: A survey”. In: *Knowledge-Based Syst.* 80 (May 2015), pp. 14–23. ISSN: 09507051. DOI: 10.1016/j.knosys.2015.01.010. URL: http://dx.doi.org/10.1016/j.knosys.2015.01.010.

[76] Scott M Lundberg and Su-in Lee. “A Unified Approach to Interpreting Model Predictions”. In: *31st Conf. Neural Inf. Process. Syst. (NIPS 2017).* Long Beach, CA, USA, 2017.

[77] Tomas Mikolov. *Word2Vec pre-processing.* 2015. (Visited on 10/23/2019).

[78] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: (Jan. 2013), pp. 1–12. arXiv: 1301.3781. URL: http://arxiv.org/abs/1301.3781.

[79] Tim Miller. “Explanation in Artificial Intelligence: Insights from the Social Sciences”. In: (June 2017). arXiv: 1706.07269. URL: http://arxiv.org/abs/1706.07269.

[80] Christoph Molnar. *Interpretable Machine Learning.* 1st ed. 2019. URL: https://christophm.github.io/interpretable-ml-book/.

[81] Douglas W. Oard et al. “Evaluation of information retrieval for E-discovery”. In: *Artif. Intell. Law* 18.4 (Dec. 2010), pp. 347–386. ISSN: 09248463. DOI: 10.1007/s10506-010-9093-9. URL: http://www.forrester.com/Research/Document/Excerpt/0,7211,40619,00.html%20http://link.springer.com/10.1007/s10506-010-9093-9.

[82] Randal S. Olson et al. “PMLB: A large benchmark suite for machine learning evaluation and comparison”. In: *BioData Min.* 10.1 (2017), pp. 1–13. ISSN: 17560381. DOI: 10.1186/s13040-017-0154-4. arXiv: 1703.00512.

[83] Nicholas M. Pace and Laura Zakaras. *Where the Money Goes Understanding Litigant Expenditures for Producing Electronic Discovery*. Ed. by RAND Corporation. 2012. ISBN: 9780833050632. DOI: 10.1214/07-EJS057. URL: http://www.rand.org/content/dam/rand/pubs/monographs/2011/RAND%7B%5C_%7DMG996.pdf.

[84] Sinno Jialin Pan and Qiang Yang. "A Survey on Transfer Learning". In: *IEEE Trans. Knowl. Data Eng.* 22.10 (Oct. 2010), pp. 1345–1359. ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.191. URL: http://ieeexplore.ieee.org/document/5288526/.

[85] Sinno Jialin Pan et al. "Domain Adaptation via Transfer Component Analysis". In: *IEEE Trans. Neural Networks* 22.2 (Feb. 2011), pp. 199–210. ISSN: 1045-9227. DOI: 10.1109/TNN.2010.2091281. URL: https://www.cse.ust.hk/%7B~%7Dqyang/Docs/2009/TCA.pdf%20http://ieeexplore.ieee.org/document/5640675/.

[86] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *J. Mach. Learn. Res.* 12 (2011), pp. 2825–2830.

[87] Kristina Penner and Louisa Well. *POLITICAL DEBATES ON ASPECTS OF AUTOMATION – GOVERNMENT AND PARLIAMENT*. Tech. rep. AlgorithmWatch, 2019.

[88] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global Vectors for Word Representation". In: *Proc. 2014 Conf. Empir. Methods Nat. Lang. Process.* 96. Stroudsburg, PA, USA: Association for Computational Linguistics, 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: http://www.aclweb.org/anthology/D14-1162%20http://aclweb.org/anthology/D14-1162.

[89] Sebastian Raschka. "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning". In: *CoRR* (Nov. 2018). arXiv: 1811.12808. URL: http://arxiv.org/abs/1811.12808.

[90] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?"". In: *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. - KDD '16*. New York, New York, USA: ACM Press, Feb. 2016, pp. 1135–1144. ISBN: 9781450342322. DOI: 10.1145/2939672.2939778. arXiv: 1602.04938. URL: http://arxiv.org/abs/1602.04938.

[91] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Anchors : High-Precision Model-Agnostic Explanations". In: *32nd AAAI Conf. Artif. Intell.* 2018.

[92] C.J. van Rijsbergen. *Information Retrieval*. 1st ed. 1975.

[93] Ryan Rowe et al. "Automated social hierarchy detection through email network analysis". In: *Proc. 9th WebKDD 1st SNA-KDD 2007 Work. Web Min. Soc. Netw. Anal. - WebKDD/SNA-KDD '07*. New York, New York, USA: ACM Press, 2007, pp. 109–117. ISBN: 9781595938480. DOI: 10.1145/1348549.1348562. URL: http://portal.acm.org/citation.cfm?doid=1348549.1348562.

[94] Stefan Rüping. "Learning Interpretable Models". PhD Dissertation. Universität Dortmund, 2006.

[95] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. "Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models". In: *CoRR* (Aug. 2017). arXiv: 1708.08296. URL: http://arxiv.org/abs/1708.08296.

[96] Johannes C. Scholtes, Tim Van Cann, and Mary Mack. "The Impact of Incorrect Training Sets and Rolling Collections on Technology-Assisted Review". In: *DESI 2013* (2013). URL: http://www.umiacs.umd.edu/%7B~%7Doard/desi5/additional/Scholtes.pdf.

[97] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning Important Features Through Propagating Activation Differences". In: (Apr. 2017). arXiv: 1704.02685. URL: http://arxiv.org/abs/1704.02685.

[98] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”. In: (Dec. 2013), pp. 1–8. arXiv: 1312.6034. URL: http://arxiv.org/abs/1312.6034.

[99] Sabine Sluijters. *Indica / 'Je kunt niets met je documenten als je er niet naar kunt zoeken'.* Nov. 2015.

[100] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *J. Mach. Learn. Res.* 15 (2014), pp. 1929–1958.

[101] D. Stathakis. “How many hidden layers and nodes?” In: *Int. J. Remote Sens.* 30.8 (Apr. 2009), pp. 2133–2147. ISSN: 0143-1161. DOI: 10.1080/01431160802549278. URL: https://www.tandfonline.com/doi/full/10.1080/01431160802549278.

[102] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic Attribution for Deep Networks”. In: (Mar. 2017). arXiv: 1703.01365. URL: http://arxiv.org/abs/1703.01365.

[103] Chuanqi Tan et al. “A Survey on Deep Transfer Learning”. In: *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics).* Vol. 11141 LNCS. 2018, pp. 270–279. ISBN: 9783030014230. DOI: 10.1007/978-3-030-01424-7_27. arXiv: 1808.01974. URL: http://link.springer.com/10.1007/978-3-030-01424-7%7B%5C_%7D27.

[104] The Sedona Conference. *The Sedona Conference Cooperation Proclamation.* Tech. rep. The Sedona Conference, 2008.

[105] The United States Department of Justice Archives. *Enron Trial Exhibitis and Releases.* URL: https://www.justice.gov/archive/index-enron.html (visited on 06/12/2019).

[106] Eric Tzeng et al. “Adversarial Discriminative Domain Adaptation”. In: *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017* (Feb. 2017), pp. 2962–2971. DOI: 10.1109/CVPR.2017.316. arXiv: 1702.05464. URL: http://arxiv.org/abs/1702.05464.

[107] Jyothi K. Vinjumur. “Predictive Coding Techniques With Manual Review to Identify Privileged Documents in E-Discovery”. PhD thesis. University of Maryland, 2018. ISBN: 9780438145832.

[108] Bernhard Waltl and Roland Vogl. “Explainable Artificial Intelligence – the New Frontier in Legal Informatics”. In: *Jusletter IT* (Feb. 2018). URL: http://codex.stanford.edu/.

[109] Xuezhi Wang. “Active Transfer Learning”. PhD thesis. Carnegie Mellon University, 2016.

[110] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. “A survey of transfer learning”. In: *J. Big Data* 3.1 (Dec. 2016). ISSN: 2196-1115. DOI: 10.1186/s40537-016-0043-6. URL: http://journalofbigdata.springeropen.com/articles/10.1186/s40537-016-0043-6.

[111] Eugene Yang et al. “Effectiveness results for popular e-discovery algorithms”. In: *Proc. 16th Ed. Int. Conf. Articial Intell. Law - ICAIL '17.* New York, New York, USA: ACM Press, 2017, pp. 261–264. ISBN: 9781450348911. DOI: 10.1145/3086512.3086540. URL: http://dl.acm.org/citation.cfm?doid=3086512.3086540.

[112] Jason Yosinski et al. “How transferable are features in deep neural networks?” In: 27 (Nov. 2014). arXiv: 1411.1792. URL: http://arxiv.org/abs/1411.1792.

[113] Dong Zhang et al. “Modeling Interactions from Email Communication”. In: *2006 IEEE Int. Conf. Multimed. Expo.* IEEE, July 2006, pp. 2037–2040. DOI: 10.1109/ICME.2006.262614. URL: http://ieeexplore.ieee.org/document/4037030/.

[114] Fuzhen Zhuang et al. *Transfer Learning Toolkit: Primers and Benchmarks.* Tech. rep. 2019. arXiv: 1911.08967. URL: http://arxiv.org/abs/1911.08967.