VRIJE UNIVERSITEIT AMSTERDAM

MASTER THESIS

Automating Authorship Attribution in Heterogeneous and Sparse Publication Data through Supervised Machine Learning

Author:
Nizar HIRZALLA

Supervisor and examiner:
Dr. Victor DE BOER
Daily supervisor:
Sara Veldhoen MSc

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in

Artificial Intelligence
Faculty of Science

November 13, 2020

Declaration of Authorship

I, Nizar HIRZALLA, declare that this thesis titled, "Automating Authorship Attribution in Heterogeneous and Sparse Publication Data through Supervised Machine Learning" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a master's degree at the VU.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: 

Date: 22-10-2020

VRIJE UNIVERSITEIT AMSTERDAM

Abstract

Automating Authorship Attribution in Heterogeneous and Sparse Publication Data through Supervised Machine Learning

by Nizar HIRZALLA

Authorship attribution is the process of correctly attributing a publication to its corresponding author, which is is often done manually in real-life settings. This task becomes inefficient when there are many options to choose from due to authors having the same name. Authors can be defined by characteristics found in their associated publications, which could mean that machine learning can potentially automate this process. However, authorship attribution tasks introduce a typical class imbalance problem due to a vast number of possible labels in a supervised machine learning setting. In addition, we use more problematic data as input data as this mimics the type of available data for many institutions; data that is heterogeneous and sparse of nature. Due to this problematic data input, we consider investigating two main methodologies: similarity learning and author classification. Additionally, we conduct various experiments in order to determine how different types of contextual metadata can be used for increasing performance. Aside from metadata, text of the publication is available, which in this case study is limited to solely a title for a majority of the publications. Thus, an experiment is conducted to seek how sparsely available text of a publication can be represented the most effectively. Conclusively, we see that the implemented models can accurately predict the right author in a significant number of cases and reduce the number of possibilities effectively in other cases. We ultimately conclude that our machine learning implementation, following the pipeline as described in this thesis, can significantly reduce the costs and time consumption of manual authorship attribution in heterogeneous and sparse publication data. We also conclude that the addition of contextual publication metadata, using the BERT text representation for sparse text inputs and harmonizing machine learning implementation with end users' (experts) current work approach will lead to the best results.

Acknowledgements

First and foremost I want to thank my supervisors Dr. Victor de Boer and Sara Veldhoen for their continuous support and invaluable feedback.

I would like to thank Sara for helping me understand the various aspects of bibliographical metadata structures and the associated authorship attribution task. Sara also gave consistent in depth support throughout the project as the daily supervisor, whether it was technical or conceptual. I would also like to thank Sara for always being reachable and facilitating opportunities to explore my own curiosities.

I like to express my gratitude towards Victor for providing enthusiastic engagement through the process of this project as well as his role in steering me through the steps of the project as official supervisor. I would also like to thank Victor for sharing many useful ideas and comments, as his experience in the dynamic field of machine learning and cultural heritage proved to be vital for the research conducted in this thesis.

Furthermore, I would also like to thank the KB in general for providing the means to carry out the research. In particular, I would like to thank Martijn Kleppe who helped facilitating and setting up the project as well as providing additional support. My thanks also goes out to the people who helped out with gaining insights in current problems with manually attributing authors to publications and usage of heterogeneous data for this purpose: Dorien Haagsma, Renske Koster, Gert Jan Boskamp and Heleen van Lopik-den Heijer. I would also like to thank the people who helped out with using external sources for integrating data: Olaf Janssen, Michel Gruijter and Djoke Dam.

Last but not least, I want to thank my parents, family and friends for providing support in many other invaluable ways that were paramount for carrying out this masterproject.

Contents

List of Figures

List of Tables

List of Abbreviations

SVM	**S**upport **V**ector **M**achine
LDA	**L**inear **D**iscriminant **A**nalysis
DT	**D**ecision **T**rees
NB	**N**aive **B**ayes
NN	**N**eural **N**etwork
k-NN	**K**-**N**earest **N**eighbors
NCC	**N**earest **C**enroid **C**lassifier
GBM/GB	**G**radient **B**oosting **M**achine
RF	**R**andom **F**orest
PC	**P**ercepton
LR	**L**inear **R**egression
LogR	**L**ogistic **R**egression
BR	**B**ayesian **R**idge
EN	**E**lastic **N**et
ADA	**A**daboost

Chapter 1

Introduction

1.1 Authorship attribution

Authorship detection revolves around the notion of attributing an author to a certain piece of text or metadata that characterizes that text. The concept of authorship detection was already established over a century ago, when early findings showed that authors have distinct stylistic as well as content-based characteristics ingrained in and revolving around their writings (Stamatatos, 2009). With this key notion of textual traits being defined by their authors in mind, research in this field rose to prominence shortly after. This was due to the belief that application of author detection would be greatly beneficial in day-to-day tasks (Kale and Prasad, 2017). Application of author detection would mean that manual attribution of authors to texts would be a thing of the past, and thus cost-inefficient, laborious tasks would efficiently be replaced (Stamatatos, 2009). Especially, considering the current age where the world wide web facilitates ever growing (textual) data and publications that can be linked to authors. This idea came to fruition, as in the present-day the implementation of said authorship detection has been shown to be useful in a number of areas, such as bibliometrics, information retrieval and plagiarism detection (Rexha et al., 2018). For doing so, different approaches as well as techniques have been developed as support for research and also applied in practical settings.

However, in real life settings not every institution that works with the concepts of authors and associated texts have the opportunity to work with complete data, balanced data or full texts (Provost, 2008). Among the reasons for such incomplete data could for example be, lack of actual data (not available due to random or non-random reasons), not having the rights to work with full texts or human input error (Hughes et al., 2019). This is combined with the notion that author classification problems causes an unique multi class classification problem with a substantial number of classes to predict from (Qian et al., 2015; Castro et al., 2015). This incompleteness and heterogeneous character of underlying data traverses author detection in multiple ways, especially if ambitions for automated author detection are in place as reliable predictions of authors can no longer be consistently made (Rexha et al., 2018).

Based on a case study this thesis will conduct a systematic approach to find how complicated, heterogeneous as well as sparse data can be used for the goal of machine learning based author detection, with the goal to ultimately automate author attribution to publication. In addition, this thesis introduces an extensible pipeline that can effectively work with this type of data and lead to accurate and robust predictions. While doing so, this thesis will also provide specific contributions to current under-researched or complex areas within artificial intelligence research, these can be defined with the following research questions:

- *Does the addition of (1) contextual author information or (2) contextual publication metadata to heterogeneous data increase prediction accuracy and robustness, and how do these types of metadata compare to each other when used as input for a machine learning model?* (section 2.1.2)
- *To what degree can the complexities of the prototypical authorship attribution class imbalance problem be alleviated by converting publication and author data to the similarity space?* (section 2.3.2)
- *How do different types of text representations for heterogeneous and sparse textual data affect model predictions?* (section 2.2)
- *Do bibliographical metadata experts differ in work approach and their perceived importance of information for authorship attribution in comparison with machine learning models?* (section 4.1)
- *How can we obtain the most optimal performance when considering sparse and heterogeneous data input for (1) author classification and (2) similarity learning machine learning tasks?* (section 2.1.3 and section 2.1.4)

The addition of contextual information for the purpose of text classification is a field with limited research (Ostendorff et al., 2019; Ráez, López, and Steinberger, 2005). However, for the integration of contextual information for authorship attribution no research has yet been conducted. A 2018 study showed potential for including contextual information in the medical field, which could indicate that this can be useful for other domains (Blanco et al., 2020). In this thesis contextual author information will consist of autobiographical information about an author as well as a summarizing embedding of previous works done by the author, while contextual publication information will consist of information that describes the text of the publication as well as the publishing details of the publication.

In addition, different representations of text can lead to significant differences for results, however research for heterogeneous textual data is, also, still limited. The field of representing text can be divided into three main approaches: usage of term weighting schemes, word embeddings and contextualized embeddings (Grzegorczyk, 2019). All of these representations have their own benefits, but comparing performances with heterogeneous input data can showcase whether state-of-the-art contextualized semantic embeddings outperform more traditional static semantic embeddings or matrices representing topical importance.

Furthermore, similar data-structure problems can be found in other datasets and domains as well (Nazábal et al., 2020; Imtiaz and Shah, 2008; Kang, 2013; Wang, 2017). As such, the systematic approach as described for author detection can then be perhaps seen as an universal pipeline for combining machine learning and heterogeneous, sparse textual (and non-textual) data.

1.2 Background

This research is based on data made available by the Koninklijke Bibliotheek (KB, national library of the Netherlands) situated in The Hague, The Netherlands. The KB tries to house all Dutch publications, as well as publications in other languages but published in the Netherlands. This makes the KB the biggest library in the Netherlands in terms of storing Dutch publications. For scoping purposes, this thesis will

focus on all available publications classified as children's literature. This data consists of 245.140 book records as written and illustrated by 488.048 authors, spanning several centuries of publication. The publications are predominantly Dutch publications (approximately 90%), but publications from other languages are also included. The publications are described by different types of data which will be discussed in Chapter 3.

This data is used by the KB to link authors related to publications to a central database called the Nederlandse Thesaurus van Auteursnamen (NTA) [1], for structural purposes and to be able to efficiently collect publications by the same author. The task consists out of two aspects, on one hand a publication record and on the other hand a set of potential authors. The list of potential authors are authors that all have exactly the same name, with ultimately the goal being that the correct author needs to be linked to the associated publication record. This task is problematic in nature and propagates the main incentive for this thesis. Since this is a task that is currently done manually, this is seen as a cost and time inefficient task in many scenarios. This is especially the case if the author to be linked has a name that frequently occurs in the database, this task then becomes increasingly laborious as the dozens to potentially thousands of authors with the same name have to be inspected before linking the author-publication record to the correct author in the NTA. An example for such an ambiguous author name includes the name 'J. Jansen', which is the name of 4854 different authors. This number can be indicative of the problematic nature of manually linking every author-publication record to authors in the NTA with ambiguous names. Therefore, to alleviate this laborious task, ideas of automating author attribution to publication have been proposed. However, the potential of artificial intelligence and more specifically machine learning has yet to be discovered for this purpose.

In this thesis we investigate a solution to automate or make it easier and less time consuming to attribute authorship to publication. Artificial intelligence could play a role with much potential for this, as the task qualifies to be done by a supervised machine learning model due to all existing publications being labelled with their correct authors. Integrating machine learning leads to an automated task where the most plausible author for being the correct author out of a list of potential authors is chosen based on probability. The probability is determined by learning patterns in the existing collection of publications and seeking which author fits the current to-be-attributed publication record based on corresponding characteristics found in their previous works. As the set of existing publications is large enough to be able learn from this shows great potential theoretically. However, as previously mentioned in the Introduction the data itself has problems of its own that ultimately make it a challenging task that needs research. We discuss these challenges with regards to the data in Chapter 3, but first present an overview of related work and the theoretical foundations of this thesis in Chapter 2.

[1] https://www.bartoc.org/nl/node/18680

Chapter 2

Theoretical background and related work

2.1 Author Detection: Usage of Machine Learning

Author detection has been an area of research for nearly 150 years (Mendenhall, 1887). Early attempts at author detection revolved around using univariate measures to characterize a particular author in terms of the style adapted for writing. Some examples for this purpose include analysis of the word frequencies of specific words in a text (Mendenhall, 1887), but also analysis of the mean length of a sentence or a word in a text (Yule, 2014). The earlier attempts showed some potential, however, ultimately showed to be inadequate for the bigger purpose of systematic author detection (Grieve, 2007). With the rise of machine learning in recent decades, the integration of machine learning for the purpose of author detection proved to have increased levels of effectiveness and accuracy of author attribution to a certain piece of text (Sebastiani, 2002). When using machine learning, a key distinction has to be made between using supervised learning versus unsupervised learning (Alloghani et al., 2020). The main difference between these two types of learning is that in supervised learning environments there is a 'ground truth', i.e. the publications in the data that is trained upon are labelled accordingly with their authors. In unsupervised learning a ground-truth is not available and thus this means that classification is based on unlabelled data. For the research conducted in this thesis ground-truths have been provided as all publications are labelled by their respective authors, meaning that literature considering supervised learning will be mainly taken in consideration.

Another key distinction for author detection is the notion of having input data in the form of full texts or smaller texts (Chandra Sekharan, 2017). Full texts indicate that all text of the publication is available and ready for use. This gives more input for classification and therefore could provide more grounds for characterizing a document (Grieve, 2007). The other option is the usage of smaller texts, which indicates that documents consist of a sentence or a paragraph at most.

In this thesis text types of shorter nature are part of the available data of the KB. Specifically this means that text input will be in the form of text snippets existing of title, summary of the publication and other annotations about the publication. However, only the title can give an indication of the writer's stylistic characteristics as it is usually fabricated by the writer. The other type of textual features (abstract, annotations) tell us more about the content of the book as they are usually written down in conjunction with third parties. Thus, a combination of syntactic as well as semantic analysis will play a role in substantiating a machine learning model's ability to learn from textual input.

Other differences with the typical studies on the subject include the availability of contextual metadata. These are mostly categorical or numerical features that give some insight about the publication (see Chapter 3). Thus, in its totality the input data can be classified as 'smaller text' input with contextual metadata in addition. Accordingly, the literature research for this aspect will mainly focus on machine learning performance based on data inputs of this nature. An overview of the possible techniques and approaches for this purpose will be researched, however, a start will be made by considering if techniques for full text analysis can be also be fruitful for smaller text analysis.

In similar nature, converting the data to the similarity space for an alternative approach for the problem at hand will be discussed in Section 2.3.2.

2.1.1 Drawing insights from author detection in full text inputs

Author detection in full texts, e.g. all text of a published book or article, could add precedence for performing a more in-depth stylometric analysis (Rexha et al., 2018). This poses to be a more difficult concept in smaller texts as there might not be adequate text to detect stylometric characteristics (e.g. in a title of a book) (Rexha et al., 2018). In addition, only one of the textual features available is usually directly written by the author, which rather makes this a limited option.

Furthermore, analysis of full text includes text segmentation techniques such as style breach detection and author clustering, followed by learning (Tschuggnall et al., 2017). Style breach detection characterizes itself by splitting texts based on shifts of topics, which makes this not a viable solution for shorter text snippets that are centralized on summarizing a publication (Khan, 2017). Other usage of full text for the purpose of learning, however, show possibilities for the same usage with shorter text snippets. For example, a study that includes syntactic graph feature extraction methods which allows for integration of multi-layered language descriptions into a single structure (Gómez-Adorno et al., 2016). An alternative approach is using Deep Learning for learning from full texts (Mohsen, El-Makky, and Ghanem, 2016), this was done by using variable size character n-grams while using the Stacked Denoising Auto-Encoder (SDAE) for extracting document features. Afterwards a SVM is used for classification. Common ground can be found in the latter two studies as they provide some insights for author detection in full-text settings, while not depending on text segmentation or stylometric analysis too much, which can essentially also be used in shorter text snippet inputs.

2.1.2 Author detection in smaller texts and associated metadata

Most research focuses on author detection revolving around full or complete text inputs (Denecke, Risse, and Baehr, 2009), while this thesis will focus on smaller text inputs (in the form of bibliographical descriptions). Such techniques applied to specifically smaller texts and associated metadata for the purpose of classification include the use of linear predictive modelling such as SVM (Diederich et al., 2003; Koppel, Schler, and Argamon, 2009; Fissette, 2010; Sudheep Elayidom et al., 2013; Schwartz et al., 2013), DT (Abbasi and Chen, 2005), NB (Abuhaiba and Dawoud, 2017) and LDA (Seroussi, Zukerman, and Bohnert, 2011; Seroussi, Zukerman, and Bohnert, 2014). As well as ensemble learning (Gressel et al., 2014; Kilinç, 2016; Liu et al., 2017) and on the other hand the use of neural networks (Ge, Sun, and Smith, 2016; Shrestha et al., 2017; Ruder, Ghaffari, and Breslin, 2016). The different types

of supervised learning models, as well as their potential performance on heterogeneous data, will be discussed in section 2.1.3 and 2.1.4.

Research articles that focused on supervised learning for author detection, with sparse text inputs combined with the use of associated metadata or contextual information have not been found. In fact, only limited works exist on using bibliographical information, such as textual snippets and metadata, for any classification purpose (Ráez, López, and Steinberger, 2005). However, a study is found that includes metadata and textual data as input for the purpose of author document classification, and was recently published (Ostendorff et al., 2019). The publication metadata consisted for example out of various information such as publication date, number of authors and the title of the publication. This metadata was represented as features and authors were represented through automatically generated graph embeddings as additional metadata. This caused the creation of dense vector representations for each author in a way that distances between vectors (i.e., the topical similarity between the authors) can predict the occurrences of edges in the graph. Using Wikidata, subsequently the graph model was trained and a translation operator was used to represent relations. A neural network that generates BERT word embeddings (Adhikari et al., 2019) was used to derive contextualized representations from textual features, and accordingly text and non-text features were concatenated and fed through a multilayer perceptron. Through this way of combining the different types of the data the model could then make predictions. The model performed relatively well on generalized labels, but had difficulties in detecting the actual labels on a more granular level and focused on classifying genres instead of authors. Even though the ultimate goal is different with the goal in this thesis this provides some useful insights in how to combine different types of textual and non-textual data.

A study with similar heterogeneous and sparse metadata also showed great potential, this study was also done in the domain of bibliographical research (Denecke, Risse, and Baehr, 2009). In this study limited bibliographic metadata was used, such as author name and information about the title. Consequentially, features are created and the text is pre-processed using n-grams frequencies. Two main features are created in the form of a class specific score, which tells us something about the similarity of a document and the class. In addition, a feature that tries to summarize a title with a keyword is created. In comparison to this thesis, a similar feature that maps keywords of the titles is available in the database provided, and a score can be calculated given the available data. After this, machine learning was used after applying a rule-based classification methodology in order to classify a text to 1 of the 6 possible topics. Different types of machine learning algorithms were used where ensemble learning (boosting with LogitBoost) proved to grant the best results. An interesting finding, is that author and publisher information performs worse when confined for input, while usage of class-specific-scores and publication metadata provides the highest accuracy. Even though this will differ from dataset to dataset, this could potentially tell us something about the usefulness of different types of metadata. Ultimately the model produces good results with up to 87% accuracy of prediction accuracy. Even though, the number of possible classes in this study is very limited, 6, meaning it does not compare with the complexity of the author classification problem, it showcases that even with limited bibliographical metadata great results still can be (efficiently) achieved.

When extending the scope to other domains, more research can be found with similar types of inputs for text classification. Including a study that focused on classifying app reviews with different review types as labels (bug reports, feature requests and so forth) (Maalej et al., 2016). Heterogeneous text inputs are used,

however additional (numerical) metadata such as star ratings is also used. Different experiments are conducted using a naive bayes machine learning model. Ultimately, the combination of metadata and preprocessed text inputs gave the best results, over only the usage of metadata or only texts. In the end accuracy scores of around 90% are achieved. This study indicates the usefulness of adding metadata descriptive of the textual information, as well as using techniques such as NLP for shorter text inputs.

On another note, a study focused on solely investigating the impact of adding metadata for patent classification (Richter and MacFarlane, 2005). Even though this was not for the purpose of author detection, some types of metadata that are used in this study are similar to those used for author detection. Some of this similar metadata includes author names, publication date and author affiliation. Using the k-nearest neighbour algorithm classification was ultimately done, and a comparison of a model using the aforementioned metadata versus a model that does not use metadata was done. Usage of metadata ultimately increased classification accuracy by 5% and recall with approximately 3%, which were deemed statistically significant. However, a search for the right parameters was also essential as the usage of wrong parameter values could have the opposite effect (i.e. deterioration of accuracy after adding metadata). The study concludes that adding metadata or contextual information can be 'extremely useful' for classifying patents, but warrants that usefulness of metadata should be reviewed on a case-to-case basis. In the context of this thesis, the usefulness of available metadata (both publication specific and author specific data) is explored by interviewing metadata-specialists (Section 4.1) before its addition to the input data. Furthermore, the addition of different types of data is also reviewed incrementally in Section 5.2 and 5.3.

In other related work more examples can be found, including text classification for inferring gender of movie reviewers (Otterbacher, 2010), web documents classification (Fathi, Adly, and Nagi, 2020) and text classification in the legal domain for the purpose of predicting the ruling of a supreme court (Sulea et al., 2017). The conclusions of these studies are in line with earlier mentioned studies. A trend can be observed when considering the different studies, namely that augmenting sparse or heterogeneous text snippets with metadata always has a net benefit while not increasing computational complexity drastically. Even though, the considered studies vary significantly in terms of the domains of application, they are all similar in the sense that they emphasizes on the usefulness of adding metadata that describes text while having varying machine learning pipelines. This is remarkable, as the addition of (created) metadata describing sparse textual inputs overall is still a scarce practice (Otterbacher, 2010; Ráez, López, and Steinberger, 2005; Denecke, Risse, and Baehr, 2009).

Overall, conclusions can be made that preprocessing of textual data and feature engineering in combination with the addition of metadata achieves empirically good performance when used with heterogeneous and sparse data inputs. Albeit, none of the studies had the goal of author detection, the findings and conclusions are persuasive as the underlying data structures (and even some of the features) are similar. Therefore, these studies and their respective conclusions should be considered for the problems described in this thesis. When translating these conclusions to this thesis we can for example use contextual bibliographical metadata, as well as possible author information to enhance the information that defines the author labels. Similarly we can preprocess the publication text and preprecocess non-textual data in various ways, apply feature engineering and integrate linked data (which we will discuss in Section 2.3.1).

2.1.3 Linear modelling and ensemble learning

As section 2.1.2 showcases, there are multiple viable options when it comes to using different machine learning algorithms for author classification with heterogeneous data inputs. As different machine learning algorithms can vary on a case-to-case basis in regards to performance, most of the time there is not a streamlined approach for implementation. Implementing the different, viable options and comparing them with each other will prove to be useful for garnering insights and finding the best performing model. For this purpose, we will consider three types of commonly used machine learning algorithm categories, which consist of (1) linear modelling, (2) ensemble learning and (3) neural networks or deep learning. In this section we will discuss linear modeling and ensemble learning, as they tend to be used in similar real world settings with regards to heterogeneous data inputs. The usage of neural networks and deep learning will be discussed in Section 2.1.4.

Linear modelling

Linear modelling exists of classical machine learning techniques such as SVM, LDA, Naive Bayes (classification) as well as logistic regression, ridge and lasso (regression). Linear classification and regression models use linear properties related to the characteristics of the input data. These linear properties are estimated by the model from relationships found within predictors (the features) and output variables (class or regression outcome) (Yuan, Ho, and Lin, 2012). These types of algorithms can either be generative or discriminative in their determination of the parameters of a linear model. Generative models (e.g. NB and LDA) model conditional density functions while discriminative models (e.g. SVM and logistic regression) perform regularization on the final model while maximizing output of the training set. Both can perform reasonably well on tasks with many features as well as heterogeneous data inputs, as referenced in Section 2.1.2. A collection of linear classification and regression models will be taken and used for this thesis and their performance will be compared. To engage with the research problems of this thesis a focus will placed on using models that have properties implemented that could (theoretically) work well on heterogeneous data inputs. In addition to some of the named models, an example of a model specifically catered for such tasks is the Stochastic Gradient Descent (SGD) algorithm with a modified huber loss function, which makes sure it is not heavily influenced by outliers or low-information density data (replacements of missing values) in the data learned upon (Zhang, 2004). At the same time SGD does not completely ignore outliers either, which means it considers both aspects.

Ensemble learning

Ensemble learning introduces an area of machine learning where multiple models are combined through ensemble techniques. Theoretically this leads to better performance as they use different 'perspectives' in their substantiation for making predictions (Zhou, 2012). These perspectives provided by different models tend to lead to decreases in noise, bias and variance, which are usually the main factors for cause of error in predictive modelling. In accordance, empirically ensemble learning has lead to better performing models if there is significant difference between the predictive nature and output of the models (Rokach, 2010). Heterogeneous data input tends to naturally produce diversity among models, which makes ensemble learning a theoretically well-reasoned fit for this thesis (Gashler, Giraud-Carrier, and Martinez, 2008).

Within the field of ensemble learning, there are various techniques for creating the ensemble model. There are simple methods such as taking the mode or (weighted) mean of the output results, but there are also more advanced methods. Most popular advanced methods are the ensemble methods of (1) bootstrap aggregating ('bagging') and (2) boosting (Zhou, 2012). A visualization of both techniques can be found in Figure 2.1. Bagging perpetuates different models to train in parallel, upon different subsets (samples) of the data. Afterwards the output of all models are combined through majority voting. A robust ensemble learning algorithm for this type of technique is *random forests* which will be explored in this thesis. Random forests takes an extra step in comparison to base versions by ingraining randomly chosen features for the different samples, which has empirically performed well (Breiman, 2001).

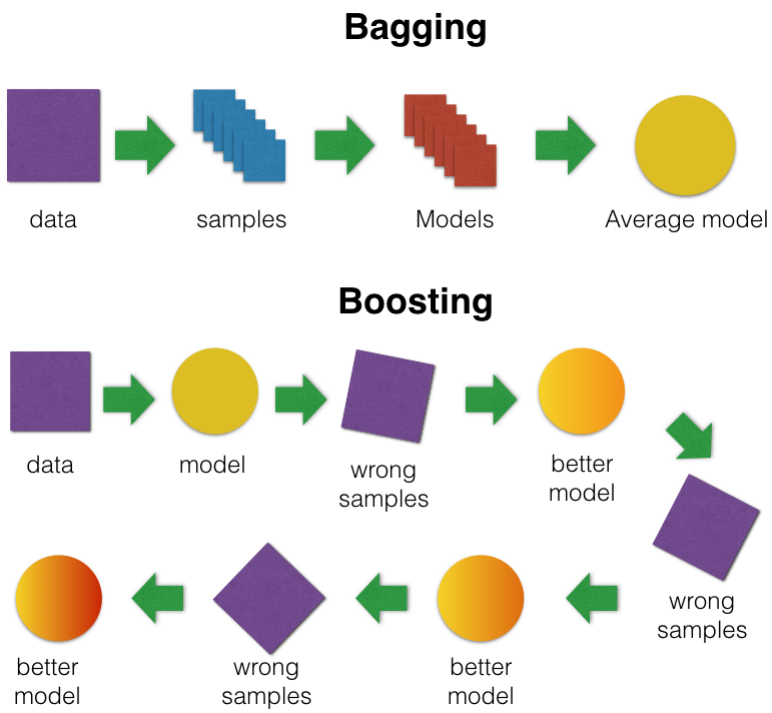

FIGURE 2.1: Main methods of ensemble learning visualized: bagging (parallel learning) versus boosting (sequential learning)

On the other hand, boosting is a technique that uses iterations with adjustments of weights based on the performance of prior classifications. If a previous classification was incorrect, then the weight of that sample will be increased. Unlike with bagging, boosting trains without the binding of models to certain subsets and instead the models train on all data, however, in a sequential fashion. Thus, models learn upon the mistakes of previous models and are fitted to the residuals of the prior models. Essentially, this creates a set of weak learners that will gradually convert to a strong learner. Examples for ensemble algorithms implemented with boosting is Gradient Boosting and AdaBoost, both of which have empirically had great results (Mason et al., 1999; Freund and Schapire, 1996). When comparing between the two main techniques, boosting has empirically performed better (with hyperparameter tuning) than the bagging approach in multiple studies (Bühlmann, 2012). However, both will still be taking in consideration due to the intrinsic trait of specifically the bagging algorithm of random forests that prevents overfitting, as well as both types'

characteristic performance on heterogeneous data inputs (Gashler, Giraud-Carrier, and Martinez, 2008).

2.1.4 Deep learning and neural networks

Contrary to linear modelling, neural networks can estimate non-linear relationships between variables and are known for approximating complicated functions of rich vectors of input variables (Wang, 2003). Thus, in this type of learning non-linear transformations are instead learned. A neural network's prototypical lay-out exists out of a multitude of networks composed of several layers. These layers in turn exist out of nodes where inputs are combined with computationally determined adjustable weights, which determine how the input should be perceived by the model. This can cause, for instance, amplification of the input's importance in relation to the learning task (such as attributing authors, based on the different feature values, to publications). Ultimately the product of the input and weight functions lead to a net input function (which combines all input-weight products) that is fed to an activation function. This is an important step, as the activation function (of a node) determines to what extent the node's output should affect the rest of the neural network as well as to what extent the signal proceeds through the network (thus influencing the prediction made by the model). The layers are chained in such a way that an output layer is usually also an input layer, which gives the 'network' lay-out, and this task is done for all nodes (of the different layers) in a parallel fashion. The intermediate layers between the first input layer and the final output layer are called *hidden layers*, where all the mentioned calculations (using input, weights and activation functions) happen. Variations exists in the form of deep learning networks, which are known to be intrinsically more enriched as they contain more hidden layers through which data inputs will travel and activated for learning patterns (such as BERT). A visualization of both types of neural networks can be found in Figure 2.2, which also showcases the described path of input data travelling through the network to the output layer.

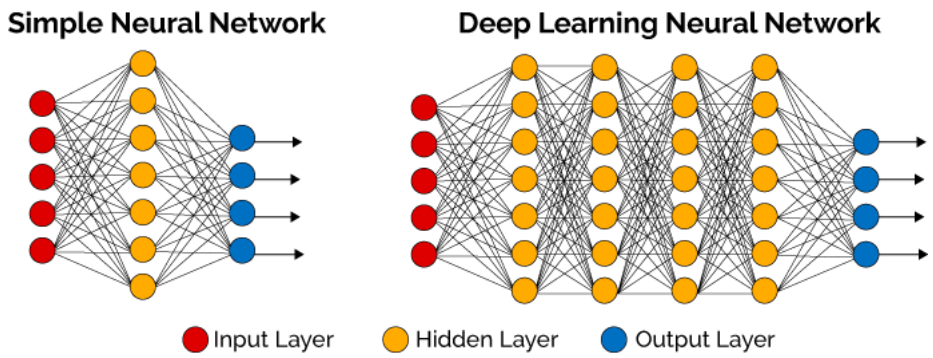

FIGURE 2.2: Simple neural networks and deep neural networks

Since most of earlier mentioned studies use linear machine learning models for similar tasks, these will mostly be used for modelling. This might be theoretically suitable due to the notion that descriptive contextual metadata is usually categorical of nature (and thus introduces mostly simplistic relationships between predictors and outcome variable). Using neural networks for linear data inputs might not be suitable and have adverse effects (Jiao et al., 2020; Park, 1994; Lee et al., 2017). The benefit of this aspect is that linear models tend to be much faster for training and research in experimental setups as well as use in production settings with similar

levels of accuracy (Yuan, Ho, and Lin, 2012; Park, 1994; Lee et al., 2017) . Ensemble learning proves to be a middle ground between linear models and neural networks in terms of training time but should theoretically perform better than singular linear models due to the integration of the 'multi-perspectives' theorem.

Neural networks take the most time to train (and use), but can derive more complex patterns and partake in deeper learning from nonlinear data, which in the case of this thesis will primarily come from text representation data (Wang, 2003). Thus, the usage of neural network becomes interesting for determining the effect of different textual representations. As text representations of publication text, such as titles and abstracts, can lever wide arrays of input variables this gives a better fit for input in a neural network compared to linear models (Grzegorczyk, 2019). The heterogeneity problem found in the dataset, is also found on a feature-level within the textual features. This type of data can showcase how newer language models such as BERT perform on these types of texts versus word embeddings and term weighting schemes, which is also one of the main research points as defined in the introduction.

For Word2Vec and BERT, which are both NLP techniques implemented with neural networks, pre-trained models exist that can reduce the computational intensity of the training task of creating word-embeddings to specifically learning from text inputs of our dataset. The theoretical properties and differences of these text representation variations, with their underlying neural network structure, will be discussed in Section 2.2.

2.2 Author Detection: Text Representation

For authorship attribution, text can be represented in various ways. Text representation is a crucial part of the author detection pipeline, as text can not be interpreted in its 'raw' form by machine learning models. Therefore, text has to be represented in a computational useful way that represents meaning of words and makes topical and stylistic differences between texts apparent. Various ways have been designed for this purpose, and the most useful ways for the purpose of author detection in smaller texts will be presented in this section.

2.2.1 TFIDF

Term frequency-inverse document frequency (TF-IDF) is a numerical, statistical representation of words that indicate the importance of a word in relation to a document in a corpus (a collection of documents) (Karen, 1972). It is a type of vector space modelling (Salton, Wong, and Yang, 1975), where the numerical values are stored in a model to represent the set of words that entail a document as a vector. It is often used in Natural Language Processing, a sub domain of AI and has empirically been shown to be effective and efficient (Qaiser and Ali, 2018; Shi, Xu, and Yang, 2009; Ramos, 2003). TFIDF is also one of the most popular text representations following a term-weighting scheme in present day, and approximately 83% of recommender systems use TF-IDF (Beel et al., 2016).

From a theoretical background TF-IDF can be separated into two components: TF and IDF (Shi, Xu, and Yang, 2009). TF stands for term frequency and can be seen in Equation 2.1. Here $f_d(t)$ stands for the number of times term t appears in document d. The denominator that encapsulates $\max_{w \in d} f_d(w)$ represents the total number of words w in document d.

$$\mathbf{tf}(t,d) = \frac{f_d(t)}{\max\limits_{w \in d} f_d(w)} \tag{2.1}$$

The second part of the TFIDF formula can be seen in Equation 2.2 which represents the idf term. IDF stands for inverse document frequency and explains the number of documents a word appears in. In Equation 2.2 |D| represents the total number of documents, while the denominator represents the total number of documents D with term t (of a document d) in it. Finally, the logarithmic value of the resulting product is taken.

$$\mathbf{idf}(t,D) = \log\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right) \tag{2.2}$$

After calculating both components of TFIDF, the multiplication of the two components is calculated for the TFIDF value for a term in a document given all documents, as is shown in Equation 2.3.

$$\mathbf{tfidf}(t,d,D) = \mathbf{tf}(t,d) \cdot \mathbf{idf}(t,D) \tag{2.3}$$

TFIDF normalizes words (terms) that might appear frequently within a document, but also appear frequently in all documents. Vice versa, words that appear frequently within a document but not in a general sense will emphasizes the exclusivity of the topical leanings of that document. As a result, terms of different documents can be accordingly weighted to their importance based on their frequency in relation to the whole corpus. This makes it easier to differentiate between different documents and thus can be used for comparing documents in terms of topical similarity.

2.2.2 Word embeddings: Word2vec and fastText

While TFIDF provides insightful numerical representations of text and is useful for topic modelling as well as measuring topical similarity, it has its limitations in deriving semantic meaning in text. Usage of word embeddings are an alternative approach. Word embeddings are more catered to deriving semantic and syntactic similarities between texts and thus can be useful as well for the purpose of attributing the right author to a piece of text (Naili, Chaibi, and Ben Ghezala, 2017). This is based on the notion that different texts can have different words but share similar meaning.

In essence, a word embedding is a mathematically driven embedding from a higher-dimensional space to a continuous lower-dimensional vector space (Mikolov et al., 2013; Naili, Chaibi, and Ben Ghezala, 2017). This means that words will be represented as real-valued vectors and mapped to a predefined vector space. These words are represented as densely distributed representations, which are in turn derived by considering the used vocabulary and how different words are used within different contexts. When words seem to share a similar type of usage or contextual placement within text they will get similar representations, hence indicate the same semantic meaning. Different types of distant metrics can then be applied to find which vectors are similar in representation.

Within the field of word embeddings different algorithms for implementation exit. For scoping purposes this thesis will in this category focus on two variations. One of these variations is the most frequently used word embedding model, and has

also been dubbed as the unofficial standard for developing pre-trained word embeddings. This is the so called Word2Vec method, which was developed in 2013 by Tomas Mikolov (Mikolov et al., 2013). Word2Vec is a group of shallow models that are two-layer neural networks, that can derive word embeddings efficiently. Furthermore, Word2Vec is especially useful due some of its inherent beneficial features, these include (1) low space and time complexity, (2) possibilities of learning larger, higher-dimensional embeddings and (3) working effectively on extensive corpora (existing of up to billions of words) (Li et al., 2019; Mirończuk and Protasiewicz, 2018; Naili, Chaibi, and Ben Ghezala, 2017). The other variation we use is known as fastText, which operates similarly as Word2Vec, but represents words as n grams of characters instead of full words and learns upon these representations instead. This key difference in representation of words for training can lead to an improvement in performance as coverage of words increases, which especially becomes noticeable when trying to deal with out-of-vocabulary words.

Word2Vec and fastText operate following either a underlying Continuous Bag-of-Words (CBOW) model or a Skip-gram model. The architecture of both models can be seen in Figure 2.3. In the CBOW approach the model examines the words surrounding the to-be-predicted-word and tries to infer context and based on that predict word $W(t)$. Each word is consecutively represented as a vector resembling a feature and ultimately become word vectors, post-training. On the other hand, the Skip-gram model is essentially the CBOW model in reverse. Instead of surrounding words, a singular word is used for input. Based on this singular input, surrounding words are predicted and ultimately context is inferred that way. While CBOW is usually faster and operates more efficiently than Skip-gram, Skip-gram tends to achieve higher accuracy with words that do not appear frequently (Suleiman, Awajan, and Al-Madi, 2017). Depending on context this gives both options viable precedence for usage.

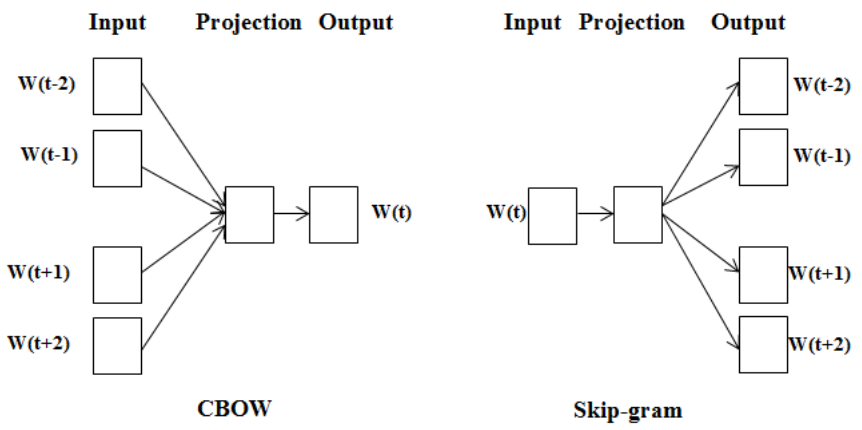

FIGURE 2.3: Underlying training models in the Word2Vec architecture. SOURCE: (Suleiman, Awajan, and Al-Madi, 2017)

When looking at the option of word embeddings in the context of the available data, this makes it a good fit. The database that ultimately lists all dutch publications consists of millions of books. This considerable number of publications will in turn lead to a corpus with many (distinctive) words. Thus, the aforementioned efficient and scalable nature of Word2Vec and fastText make this computationally and functionally a suitable alternative approach for representing the text of publications.

2.2.3 Bidirectional Encoder Representations from Transformers

While Word2Vec, fastText and other word embedding models provide the adequate means to carry out semantic similarity analysis given a set of documents, these approaches have their limitations as well. Examples include the inability to handle Out-Of-Vocabulary (OOV) words (words that are not linguistically familiar) and the ineptness to represent different contextual uses of the same word (homonymy) in separate vectors (Naili, Chaibi, and Ben Ghezala, 2017). For instance, we consider sentence A and B, with sentence A being "John bought a beautiful ring for Mary" and sentence B being "The children sat on the floor in a ring". The word "ring" has a different meaning in both sentences, but will get a singular vector representation trying to map both contexts when using word embedding methods.

In 2018 Google developed a system encapsulating a pre-trained neural network called BERT (Bidirectional Encoder Representations from Transformers) that has been the best performing language model, as of to this date. State-of-the-art performance was, consequentially, achieved on 11 different NLP tasks (Devlin et al., 2018). BERT is a technique that uses contexualized embeddings for text representation, thus, meaning that in the aforementioned example the word embedding will be different for every use of the same word (according to the sentence). This approach is fundamentally different from Word2Vec (and TFIDF), which in turn can be an explaining factor in its state-of-the-art performance. However, as BERT is a relatively new technique, the theoretical explanation for its state-of-the-art performance has not yet been fully uncovered (Kovaleva et al., 2019). Several studies are currently analyzing BERT from different perspectives for explainability in order to understand its performance (Clark et al., 2019; Kovaleva et al., 2019).

Fundamentally, BERT is driven on the notion of pre-training deep bidirectional representations from unlabeled text (Devlin et al., 2018). It combines bidirectional training of a typical transformer machine learning model to language modelling, which makes it a innovative technique (Devlin et al., 2018). Additionally, BERT has also been applied to classification tasks with labels that are in line with the author attribution (classification) task with success (Sun et al., 2019; Adhikari et al., 2019). Furthermore, while constructing bidirectional representations, BERT jointly conditions in all layers on the entire sentence unlike with traditional directional encoders (which operate from left-to-right or right-to-left). Finally, BERT gets contextualized embeddings through a combination of two underlying techniques before its fed word sequences, this is done using (1) Masked Modeling (MLM) and (2) Next Sentence Prediction (NSP).

The Masked Learning Model can be visually seen in Figure 2.4 and entails a set of steps. First of all, the model uses a token called [MASK] which masks a certain word. This is done for approximately 15% of the words, which is then used as a label for prediction and input for the transformer encoder mechanism. At the output of the transformer encoder layer, a classification layer is added for this purpose. Here, the output vectors are transformed into the vocabulary dimension ('Embedding to vocab step' in Figure 2.4). The model tries to predict these words based on the surroundings of the masked words, which are non-masked and thus resemble their true values. Finally the masked word is predicted by calculating the softmax probability for all words in the vocabulary and consequentially choosing the most probable word. Even though non-masked words are also fed to the classification layer, these are ignored as they are already 'seen'.

Parallel to this process, NSP is also trained. This process can be found in Figure

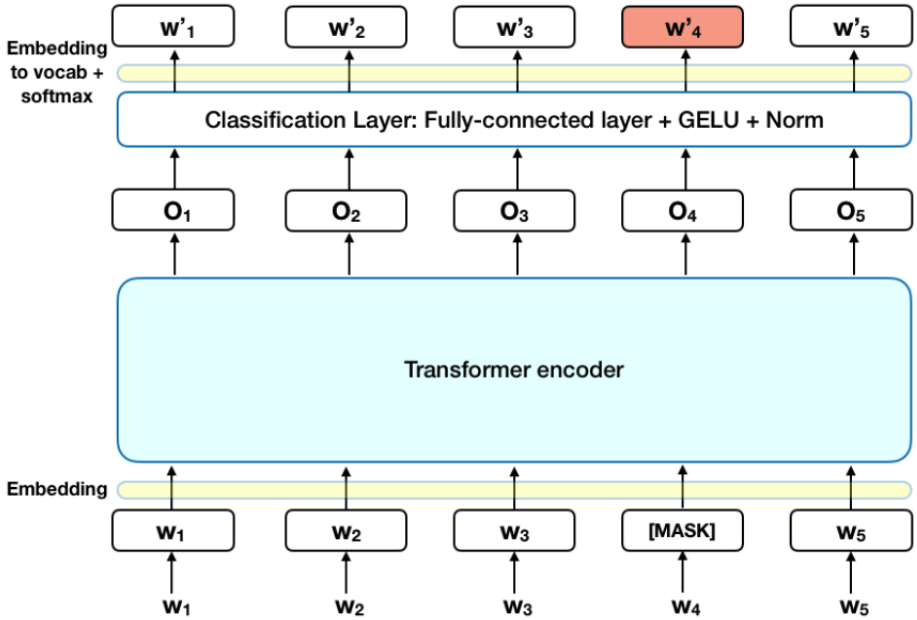

FIGURE 2.4: Masked Learning Model (Gannon, 2019)

2.5. NSP encapsulates a different type of contextualization, whereas the focus revolves around seeking if two sentences are linked, i.e. sentence 2 is the subsequent sentence of sentence 1. For this purpose BERT randomly designates 50% of the sentences as a subsequent sentence in a pair in the input text, and 50% as a random sentence from somewhere else in the corpus (and thus has no relationship with sentence 1). Next to the process of NSP, Figure 2.5 also showcases the expected input for BERT. Three different embeddings, which in summation form the input, can be found:

1. Transformer positional embedding; indicating the position of the token in the input text.
2. Sentence embedding; indicating to which sentence a token belongs
3. Token embeddings; indicating vocabulary ids for the respective tokens

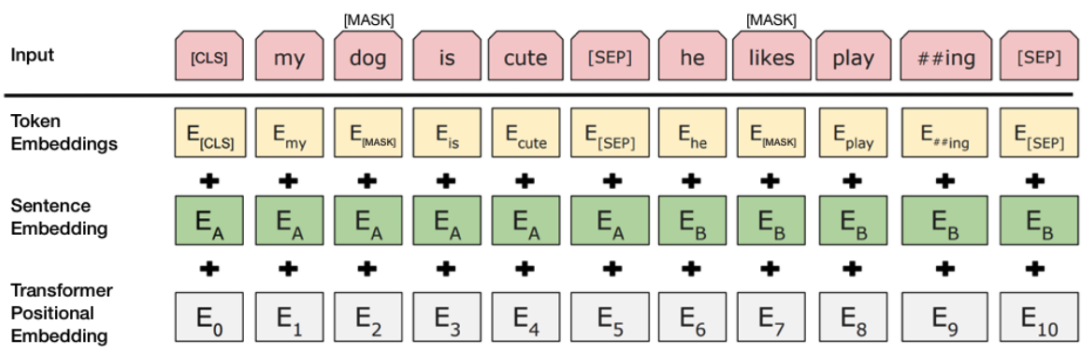

FIGURE 2.5: Input sequences for BERT as well as NSP visualized (Devlin et al., 2018)

In the subsequent input we can also observe the masked words as seen for the purpose of MLM, as well as two key tags: [CLS] and [SEP]. [CLS] denotes the first

token of every sequence, it is used as a token for classification tasks. [SEP] is a sequence delimiter token, which is used for the purpose of NSP to indicate when two different sentences are separated. Ultimately, predictions are made by feeding the combination of the different embeddings (the input) as seen in Figure 2.5 for the subsequent transformer model. Afterwards a label indicating whether sequence A is subsequent to sentence B is predicted (true or false).

Thus contextualization is done on sentence-to-sentence basis, as well as on word-within-a-sentence basis. This gives the model a clear understanding of how words are exactly used, and what their contextualized meaning entails. Therefore MLM and NSP are trained together, and their respective loss functions are combined for minimizing loss. This combination of loss functions makes it so the model trains at an optimal level, however, at the expense of a slower convergence time.

For the goals of this thesis BERT has to be fine-tuned, as the task of author attribution using associated text and metadata as input are specific tasks. However, fine-tuning has been a main consideration as BERT's usage was intended to be for a set of varying NLP tasks. Therefore, fine tuning can be done relatively easily with BERT without changing the architecture as it can already been achieved with only one additional output layer. This notion makes it beneficial to use BERT (which we will use in Section 5.6), as it is easily adaptable to a set of different goals (Devlin et al., 2018).

2.3 Heterogeneous data for predictive modelling

Extending the sole scope of author detection, we also take a look at dealing with heterogeneous data in a more general sense. Heterogeneous data brings forward all kinds of problems that should be addressed in the pursuit of a model that can make robust and accurate predictions. The data that is being used has mainly two problems as previously mentioned: class imbalance and disparity of information richness. Thus, this chapter will be centered around discussing the theoretical possibilities for implementing solutions to these two main problems, as well as how the respective solutions performed in other research.

2.3.1 Disparity of information richness and availability

The stored data regarding publications differ significantly between the different publications, e.g. some publications have dozens of fields of metadata describing the publication while other publications might only have a few fields to describe this. This can be problematic as it can lead to a model that predicts its labels or likelihood, in this case authors, with much less confidence as there is not enough metadata that could be used for classification to associate certain values of the features with an author. In turn, this could lead to bad predictions and thus a model that can not be deployed in a real life setting. Next to this, not having enough data can also let authors fade away from the overall ranking, as lack of data for a certain author means that it will not match well with input (or test) data. Therefore the likelihood of this author being 'correct' for prediction will decrease, even if the input/test author is the exact same author and thus in fact should get predicted/be number 1 in the ranking. This is a paradoxical situation that can diminish the power and accuracy significantly of the model.

A few approaches can be considered to be a solution for this disparity in information per publication, which will be discussed in this section.

Preprocessing

Preprocessing is usually the first essential step in the machine learning pipeline (Famili et al., 1997). It has the potential to deal with a multitude of problems with data, as it can be a solution to missing, incomplete, erroneous, corrupt or out-of-range data (Famili et al., 1997; Bhaya, 2017). Furthermore preprocessing can be useful for transforming data in more meaningful ways, this could be for example done by applying normalization, discretization or hierarchy derivation. Finally, reducing dimensionality and numerosity as well as aggregating data are also procedures typical to the data preprocessing step. As these data problems usually contribute to disparity in information richness on a feature-to-feature basis, this makes it a particularly fundamental step in processing heterogeneous data structures.

The data as provided by the KB has three main categorizations: numerical, categorical and textual data, which will be discussed more in depth and showcased in Chapter 3. The numerical data is not sensoric but rather of describing nature. For such numerical features, missing values can be imputed by using the median or mean (depending on the distribution of the data). In some numerical features cases this would not be a logical imputation, such as the age of the author at the time of a certain publication. In such a case the missing values can be imputed based on the year of publication feature, which gives insights in how often an author publishes as well as the timeline of all its publications. In addition, when also considering the distribution of the rate of publications for different age groups, missing values can then be imputed for such a feature. For categorical variables domain knowledge is usually required, as these are classifying labels that are specific to the publications, thus metadata-specialists have to be interviewed for this purpose. For textual data a set of different approaches can be taken. These include stopword removal, tokenization, stemming/lemmatization, making all words lowercase and removing special or redundant characters. Missing textual data is not a problem, as all publications have at the very least a title that can be used to describe the publication.

Feature engineering

Feature engineering is a fundamental process in the machine learning pipeline and has been deemed integral to whether a machine learning model succeeds or fails (Zheng and Casari, 2018; Domingos, 2012). It is used to derive new features from already existing raw data, using domain or empirical knowledge. These new features then either 1. provide additional information or insights about the raw data that can aid the predictive modelling process or 2. make raw data compatible with the requirements of the used machine learning algorithm (Zheng and Casari, 2018; Domingos, 2012). As heterogeneous data is by nature already harder to use for predictive modelling, it is considerably more important to have a correct feature engineering step in the machine learning pipeline (Han and Lam, 2003). A correct feature engineering step can then alleviate some of the disparity in the information richness and availability and thus in turn increase the accuracy of the predictions made.

Feature selection

Another important step in the machine learning pipeline is feature selection (Dash and Liu, 1997). Feature selection is the process of selecting relevant features from a set of features and returning these as the subset of data that can be used for further purposes (Dash and Liu, 1997; James et al., 2014). In essence this means that feature selection deals with omitting redundant or irrelevant features for the input of

a machine learning model, and thus removing data that can negatively impact the machine learning model while limiting the loss of valuable information (Bermingham et al., 2015). In a general sense this has multiple benefits: avoid erroneous predictions based on noisy data, shorter training times for learning, reinforce proper generalization, avert the curse of dimensionality and simplify the model for better interpretability (James et al., 2014; Bermingham et al., 2015). Considering these benefits this can already compensate for some of the overall problems with heterogeneous data. Additionally, this is also beneficial for reducing heterogeneity as overall heterogeneity can be reduced by removing features that individually consist of heterogeneous data.

Usage of external knowledge

In line with the earlier discussed study that used automatically generated graph embeddings (Ostendorff et al., 2019), studies have showcased the potential of using external resources. Including external sources for gathering information can enrich text representations and the available metadata as these sources can potentially have useful information for use that was not available previously. Studies have empirically shown that integrating external knowledge can improve the task of text and associated metadata classification (Wang et al., 2009; Zhang et al., 2019). In the cited cases this was done by considering the relations between entities as well as incorporating knowledge graphs. Examples include using external knowledge bases such as Wikipedia and Wikidata that have considerable amount of structured data available that provide additional information. This notion is substantiated by the fact that Wikipedia and Wikidata have so-called linked data (all entities are mapped with relationships) which has shown to be useful in other domains for the purpose of research, as well (Farda Sarbas and Müller-Birn, 2019). In addition to the mentioned general databases, there are several external databases that are specifically catered to bibliographical information and information about certain authors. In this thesis we will explore both types of external sources and the potential information gain for the goal of predictive modelling.

2.3.2 Class imbalance: using Similarity Learning

The result of having a multi-class classification problem with an underlying class imbalance is iconic to the author attribution task (Qian et al., 2015; Castro et al., 2015). This is due to the fact that in prototypical author classification tasks there can be up to millions of classes. Every class represents an author, that has a varying number of publications substantiating the class label. This can lead to a significant problem, as it can lead to different types of predictive machine learning models perform worse on classes that are underrepresented in the data (Qian et al., 2015). This imbalance is amplified by the notion that the number of classes is much larger than in other class imbalance (binary or multi-class) machine learning tasks.

Therefore, an approach that reverses the training architecture can be used to alleviate this problem, which is also known as similarity learning. Instead of classifying the publications with author labels, the process is reversed and for every publication similarity with potential authors is calculated. For the disambiguation task in this thesis the computational complexity can also be reduced as the name of the author is given with publication data. Similarity learning is driven on the notion that authors are distinctive in their biographical, topical and stylistic characteristics, as well as

other metadata associated with their publications. In essence this is in line with the traditional author classification task, however this differs in two aspects:

1. Similarity learning considers all possible authors for a publication equally based on their characteristics, independent of the number of books they have published. (Qian et al., 2015; Castro et al., 2015)

2. Using explicit author information is made possible.

Thus, in this variation the earlier mentioned contextual author information in the form of author embeddings can be used. This can not be done when using publication metadata for a direct author classification task, as feeding the model direct information about an author within data that represents the publication is a form of deceiving the model. This is based on the notion that in a real-life setting, author information can not directly be linked to a publication as the author is still unknown.

A possible approach for implementing similarity learning is transforming the publication data into a similarity space (Qian et al., 2015). This means that embeddings are made from a set of publications resembling previous works of an author and compared with the information of the to-be-attributed-publication. The previous works are received from the training set, while the comparison is made with an unseen publication of the test set. The distance is calculated between the embeddings representing the author and the publication. These set of distances represent similarity scores that can be calculated using a metric such as cosine similarity or the euclidean distance. After obtaining a set of similarity scores, the similarity between the author and the publication should be a relationship of the notion that more features having a higher similarity should lead to an overall higher likelihood that the author wrote the publication. This type of learning has showed good results in different studies, ranging from social media texts inputs, emails, unknown documents and publications (Chen et al., 2011; Boenninghoff et al., 2019; Arcia et al., 2017; Castro Castro et al., 2015; Chen et al., 2009).

In fact, contextual information of the author is available for a portion of the authors (which will be discussed in Chapter 3). As mentioned in the introduction, addition of such information is still a topic that lacks research. Thus, in addition to the publication data embedded into a similarity space this type of information is additionally used. For the available biographical information about an author the semantic distance can be calculated with the content of the book, while age and role can intrinsically be telling features by considering their specific values for different publications.

Chapter 3

Data

In this section we will explore and discuss the data, which is available at the KB. The data can be accessed by all researchers of the KB, but external researchers could potentially also get access to the data on request [1]. The data can primarily be divided into two types of data: contextual metadata and descriptive textual data related to the content of the publication. The metadata includes various types of information, such as year of publication, language of publication, original language, name of publisher(s), CBK genre of book and CBK theme of book. CBK [2] is a national catalogue used by various libraries aimed at describing children's literature, for example by assigning categorical genre and theme labels to a publication. Furthermore, there is also the NUGI and NUR genre and rubric descriptive features, which fulfil a similar purpose but tend to have more general descriptions as they are not catered to children's literature. In addition, we also have the available descriptive textual data, which consists of the title of the book, the summary of the book and additional notes about the publication. Examples for these types of data entry points can be found in Figure 3.1, 3.2 and 3.3. The data examples give insight in the heterogeneous nature of the data; some features are sparse in information availability while others are rich depending on the publication.

titelvermelding	samenvatting_inhoudsopgave
@Zwemmen met Caligula	NULL
Een @belangrijke keus	NULL
@Nog een mop?	Dit moppenboekje is voor kinderen van 6 jaa...
@Levende schaduwen	Jacob Reckless heeft het leven van zijn broer...
@Wie je morgen bent	Bridge, Emily en Tabitha zijn al jarenlang be...
@Binnenstebuiten	NULL

FIGURE 3.1: Data examples for content related features (title and abstract).

In addition to the publication data, there is also author information available as mentioned in the introduction. This information includes biographical notes about the author, role of the author (for a certain publication) and birth year. Data entry points for these types of data can be found in Figure 3.4. Figure 3.4 showcases that autobiographical notes can contain some potentially useful information about what the author does in their life (i.e. profession or hobby's). However, we can also observe that some authors have low quality input for this feature, such as intractable codes that do not contain useful information. In addition, we can see some of the

[1] https://www.kb.nl/bronnen-zoekwijzers/dataservices-en-apis

[2] https://www.kb.nl/bronnen-zoekwijzers/kb-collecties/moderne-gedrukte-werken-vanaf-1801/kinderboeken/over-het-centraal-bestand-kinderboeken

j_v_u	l_v_u	t_o	t_p	uitgever
2016	nl	ned	ned	Amsterdam : Leopold
2015	nl	ned	ned	[Den Bosch] : [Blink Uitgevers]
1915	nl	eng	ned	Rotterdam : D. Bolle
1908	nl	ned	ned	Bussum : uitgave van C.A.J. van Dishoeck
2014	nl	dui	ned	Den Haag : Koninklijke Bibliotheek

FIGURE 3.2: Data examples for language and publishing related features.

genres	themas	NUR_rubriek	NUGI_genre
['eerste leesboeken', 'schoolboeken']	NULL	NULL	NULL
['gezinsverhalen', 'avonturenverhalen']	['geestelijke ...	NULL	320
['voetbalverhalen']	NULL	NULL	NULL
NULL	['uitvindingen']	NULL	NULL
['informatieve boeken']	NULL	NULL	NULL

FIGURE 3.3: Data examples for genre and themes classifying features.

heterogeneous data patterns in accordance with the publication data here as well when observing the missing values.

author_ppn	autobiography	role	birthyear
068423578	theoloog (studeerde theologie Trinity College te ...	NULL	1667
070332118	NLMD/C 2193	aut	1844
06962173X	componist, pianist ('prins der Romantiek', had enig...	NULL	1810
07106933X	NULL	NULL	1917
069331278	mb	NULL	1819

FIGURE 3.4: Data examples for author information.

The heterogeneity can be mostly explained by the rates of distinct values, percentage of value that appears the most for a feature as well as missing values. On average a publication has a rate of 44% missing data, while for individual features the missing rate values can be much higher. The missing rates for each different feature will be discussed in Section 4.2.1, where we also discuss our attempts at alleviating the high rates of missing values. The rates of distinct values as well as rates of values that appear the most for a feature can be seen in Table 3.1 for a set of varying features. Some features have many distinct values (for example, publisher and CBK themes) which can lead to low quality information and bad generalizability for a machine learning model. Other features have dominating values with a high frequency (such as the language-oriented features or role of the author) which can make it difficult to extract useful distinctions between publications. Combining these problematic elements with the high missing value rates leads to a typical heterogeneous data input that can negatively impact predictive performance of a machine learning model.

Furthermore, authors linked to the publications can be either primary authors or secondary authors. Primary authors are the artists or writers that made the book and wrote the story, while secondary authors can for example be editors or translators that have a more peripheral role. For scoping purposes, this thesis will focus on primary authors. This is also due to the notion that the primary authors are the main

TABLE 3.1: Data descriptions of various metadata features.

Feature	**Distinct values**	**Most appearing value (Missing values not included)**	
		Value	***Frequency***
Language of publication	243	nl	92%
Original language	250	nl	63%
Country of publication	196	nl	78%
Year of publication	355	2016	2%
Publisher	27066	Zwijsen (Tilburg)	2%
Number of authors	8 (Range: 1-8)	2	47%
CBK Genres	696	Picture book	14%
CBK Themes	5856	Animals	3%
NUGI genre	274	Children's literature	12%
NUR rubric	345	Picture book <6 years	12%
Role author	326	illustrator	42%

case for automating authorship attribution as they tend to encapsulate the bigger part of cases regarding linking authors with ambiguous names to publications.

There are 38259 primary authors linked to all children's books, of which 19559 have at least 2 publications. A number that gives a strong indication of the previously mentioned prototypical class imbalance in author classification tasks. Within this wide selection of different authors, there are significant differences in the number of publications per author. In the context of machine learning this means that some classes will only have a few instances, while other classes will have more instances to learn upon. In Figure 3.5 we showcase this class imbalance, as well as the total instance sizes for different publication counts by combining all primary authors (labels) with the same number of publications.

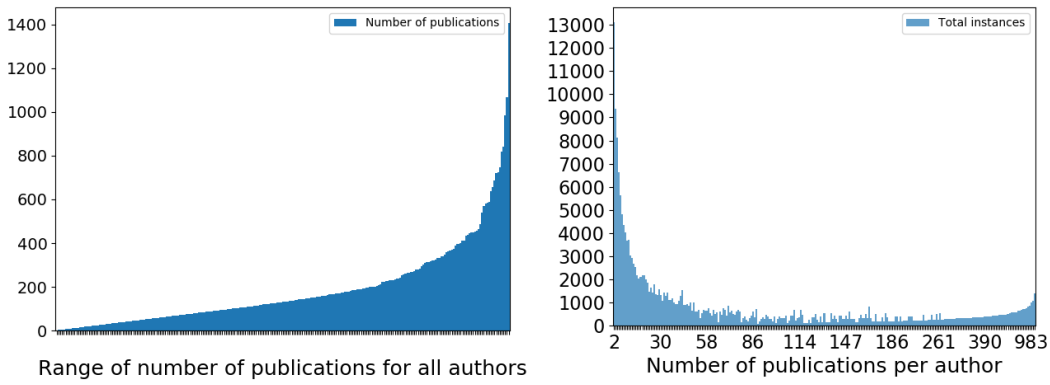

FIGURE 3.5: LEFT: range of number of publications for all authors. RIGHT: number of total records for classes (authors) with the same number of instances (publications).

We can observe that the class imbalance problem leads to an uniquely skewed dataset. Authors with many publications naturally have the most instances to learn from, while authors with only a few publications will in turn only have a few instances. Figure 3.5 (left graph) makes this visually apparent by showcasing the range of number of books published by different authors, and thus the degree of difference found in authors and their number of publications. At the same time, the classes

(authors) with the fewest instances (publications) encapsulate the biggest number of records in the dataset when combined, while classes with the most instances have a lower number of records when combined (right graph).

Chapter 4

Methods

This section describes the steps that define the machine learning pipeline for the different experiments as part of answering the research questions, as well as a description of the conducted case study. In accordance with Chapter 2, the main aspects that define this pipeline can be summarized into 5 steps: (1) pre-processing heterogeneous data and feature selection, (2) feature engineering, (3) using different data and text representations, (4) converting the publication-author space into the similarity space and (5) deciding upon fitting machine learning models for the different experiments conducted. Before getting into the technical and data-driven considerations, the preparation for mimicking the human thought process (by integrating domain knowledge from metadata specialists) and understanding the available data will be elaborated upon.

4.1 Interviewing cataloguers and metadata specialists

Understanding the expert thought process in linking hard-to-identify publications accurately to an author can be useful for several purposes, such as substantiating extra features to be made as well as using the available data in correct ways as input for machine learning. For the purpose of getting insights in the expert thought process in linking authors with ambiguous names to publications, as well as getting insights in the available metadata for feature engineering purposes, interviews were conducted with three different metadata specialists and cataloguers at the KB. In addition, a few difficult example cases were requested to be showcased for observation. During these observations the experts were asked to adhere to the thinking aloud protocol, so insights in what they are thinking were also obtained. These interviews and observations were taken independently on separate moments to showcase (potential) differences between work approaches as well as not getting biased answers. We supplement the case study by holding a survey to seek how a bigger number of metadata specialists, cataloguers and bibliographical researchers perceive the importance of the different types of information available. The results of the survey are then compared to the results of the machine learning models to seek if there are significant differences in attribution of authorship.

4.1.1 Interviews

The specialists were asked a set of questions regarding the available data features and how they perceive the importance of these in relation to linking authors to publications. Based on the answers we can categorize the importance of the different features on three levels: 1. important, 2. somewhat important and 3 not important. The features that are categorized on the first level are features that all specialists deemed important and were reliable features that the specialists could fall back

TABLE 4.1: Perceived importance of different features for expert authorship attribution

Importance	**Feature**
Important	Content (title, abstract), genre, themes, publisher, background information about the author
Somewhat important	Publication language (in combination with) original language, country of publication, year of publication, age of author, role author
Not important	No. of pages, thickness of book, unique identifiers pertaining to the author

on when facing ambiguous authors for selection. The second level introduces data features that can be important depending on context, but were either (1) not unanimously deemed important by all three experts or (2) deemed as lesser important than the features in the first level. Furthermore, the third level consists of features that are deemed not important, these are usually features that are not considered when selecting an author.

The experts clarified that in most cases authors are easily distinguishable by looking at the combination of first name and last name, however the purposes of machine learning implementation for detecting authors is meant for the cases where authorship attribution is not always clear. In those cases, the experts adhere to several stages for trying to identify which author should be attributed to the publication. A pattern can be found that divides this process in 4 stages, which are in chronological order as following:

1. Try to find the author with data available internally
2. Use external sources or software to get more information about the author
3. Elevate the publication to the data acquisition team, which will contact the publisher
4. Make a new record for the ambiguous author, possibly being a duplicate author record

At the first stage, the experts try to find the correct author for the publication using the aforementioned features as mentioned in Table 4.1. When this is unsuccessful, the process elevates to the second stage which introduces using external sources or software for finding the correct author.

These sources are primarily Wikipedia and LinkedIn, which potentially house additional useful information about the author, including a list of their publications. External software is also used, an example for this is Delpher. Delpher is an application that houses over hundreds of millions of Dutch newspapers, books, magazines and so forth. In some of these texts additional information can be found about a certain publication or authors, for example, publications from earlier centuries were systematically announced in newspapers by the publishers. An expert could then read such an announcement which could also include biographical information about the related author. This kind of newfound information can potentially give clear directions to the correct author for the publication.

When both stage 1 and stage 2 fail and the expert can not derive the correct author-publication match, the publication is sent to the data acquisition team. This

team will then contact the publisher of the book and request further information. In some cases the publisher will co-operate, but in the majority of the cases the publisher does not cooperate. The main reasons for this being that the publisher simply does not respond to requests made by experts, or, does not allocate resources or time for searching for additional information about the author.

After this, the only option left is to make a new record for the author, even though it could be a duplicate record. This could lead to data redundancy, but according to the experts it is better than the alternative of incorrectly attributing a publication to an author. Due to the fact that the latter option will lead to erroneous and polluted data. The experts made it clear that the data redundancy could also be resolved at at later time when a cataloguer discovers this through information obtained at a later time or by chance.

The expert perceived importance to the available data as well as the work protocol to approaching the problem of ambiguous names for authors gives insights that can be used for predictive modelling. However, observations were also made and these showcase favouritism between experts for using different data as well as differences in work approaches.

4.1.2 Observing difficult cases

In conjunction to the conducted interviews, we observe the specialists selecting authors for publications in multiple difficult cases for more in depth insights. These cases involved authors with the same first name as well as the same surname, with no additional clues to entice to which unique author identifier they belong. A rundown of the observations made will be done in this section, while also noting the key differences between the experts.

Expert 1 mainly firstly observed the author background information that appears after the name of the author, and accordingly filtered out authors that seem highly unlikely. Furthermore, additional inspection is done by expert 1 by observing whether the author in question had a potential baptismal name that differed between two data entries. If looking at the specifics of different authors is not conclusive for the task, then the expert starts spectating and comparing titles and genres of the books of different authors in relation to the information found in the to-be-attributed publication. This is done to find a topical or thematic similarity between the two. Somewhere in this process the expert found a suitable author to the publication usually, if not then the expert would revolt to using external sources (in this case LinkedIn) to try to get more information about the author and try to link a publication to an author through such means.

Expert 2 had a different approach, systematically prioritizing titles of books when comparing publications. In all cases this gave a good indication whether the to-be-attributed publication fitted the authors that were being compared. This could indicate that prioritization of the textual data consisting out of a sentence or paragraphs can be useful for implementation of the machine learning model. As this type of data is textual and conveys a semantic meaning, this means that the inclusion of Text Mining or Natural Language Processing can improve the performance for predictive modelling. With this approach, expert 2 clicked through more author records and spent less time spectating additional author information in comparison with expert 1. Expert 2 also placed a higher priority at considering the publisher of the corresponding book. In comparison with expert 1 age was equally considered, and similarly external sources were used if no author was found.

Expert 3 considered all data features based on context and shifted in prioritization of data features. In one case a higher priority was allocated to title and content, while in an other case priority was given to publisher but also language and possible translations. This can be indicative of shifting feature weights in a machine learning model, as obviously metadata can differ from publication to publication in terms of how useful the conveyed information is. Furthermore, expert 3 had more experience than expert 1 and 2 and this showed when observing the work approach as expert 3 also had a feeling for 'what felt right' in attributing books to authors. Furthermore, expert 3 mainly focuses on children book's and older publications for which they used Delpher significantly if stage 1 failed to attribute an author to a publication.

Expert 1, 2 and 3 all showcased authors that could be problematic for computational prediction. These authors can be categorized into three categories. (1) Authors that completely shifted the genres and topics of their publications during their writing career, for example, going from writing about music to writing horror stories. (2) Authors that have a completely different background in correspondence to the type of books they write, for example, an author that is an accountant but writes books for children. This could be indicative that the biographical notes that describe an author can be misdirecting and perhaps could reduce the robustness of an machine learning model if used as an input feature. (3) Authors that have a duplicate records in the database, this can be confusing as they are the same person but implied to be different persons due to either human error or as a result of resorting to stage 4 of the aforementioned work protocol.

In conclusion, even though the experts all follow the same protocol as described in section 3.1.1 and share a big portion of the way they approach the problem, there are differences. These differences can thus be mainly found in prioritization of data, approaches to attribution of publication to book, experience and efficiency of attribution. These difference indicate that features matter depending on context and that there are multiple ways for attributing an author to a publication when facing an ambiguous name. Therefore these aspects are taking in consideration, and substantiate a more experimental nature when implementing a machine learning model.

4.1.3 Survey

To supplement the case study of conducted interviews and observations, we also hold a survey for a bigger number of experts [1]. For this purpose we asked 18 bibliographical researchers, cataloguers and metadata specialists. Due to the bigger number of respondents for the survey we can have more elaborate understanding regarding if different types of people that work with bibliographical data on a day-to-day basis have the same perception of the available information. The survey consists out of two multiple choice questions for the respondents. The first question focuses on features they deem to be the most important for attributing an author to a publication. The second question focuses on features they would deem to be the least important (i.e. would not use for basing predictions). To reduce the possibility of noise in the results, we limit the total number of possible selections; the respondents can only select a maximum of three features for both questions. Two experts were also asked to review the survey beforehand for validity and reliability, as well as to review whether the questions and answers are clear and well understandable.

Ultimately, the survey provides insights in expert-attributed importance to a wide set of available (metadata) features for the purpose of predicting a publication's author. When inspecting the answers we find that we can find clear patterns,

[1] https://forms.gle/1hvYMGvivnyLXpss6

which will be discussed in Chapter 5 and 6. We also find that the answers to question 1 and question 2 are in correlation to each other, as features that have the most votes in question 1 are not voted for in question 2 and vice versa. Thus, we can represent the results into a singular pie chart in Chapter 5.

4.2 Supplementing and pre-processing heterogeneous data

The available data as discussed in Chapter 3 comes in various ways with many features having significant rates of missing values, which can transcend up to missing value rates of 90%. Aside from missing values, the database also contains data with human error, spelling mistakes and ambiguous input. To alleviate some of these problems, this section will elaborate upon how we pre-process different types of data with their associated problems for a more useful data input. In supplementation, rationale behind the consequential feature selection is also discussed. Finally, we elaborate upon the methods for integrating linked data in an attempt to make the data more elaborate and refined.

4.2.1 Considering missing values

Many features contain a higher number of missing values than usual due to the heterogeneous nature of the data. The percentages of missing values for different features can be found in Figure 4.1. Only a few features have no or a negligible percentage (< 0.1%) of missing values, these are the language, year and country of publishing of the book as well as the title of the publication and the publisher. In Figure 4.1 these have been labeled as 'Others*'. The other features vary in missing value percentages, from reasonable ranges up to high percentages possibly indicating unusable features.

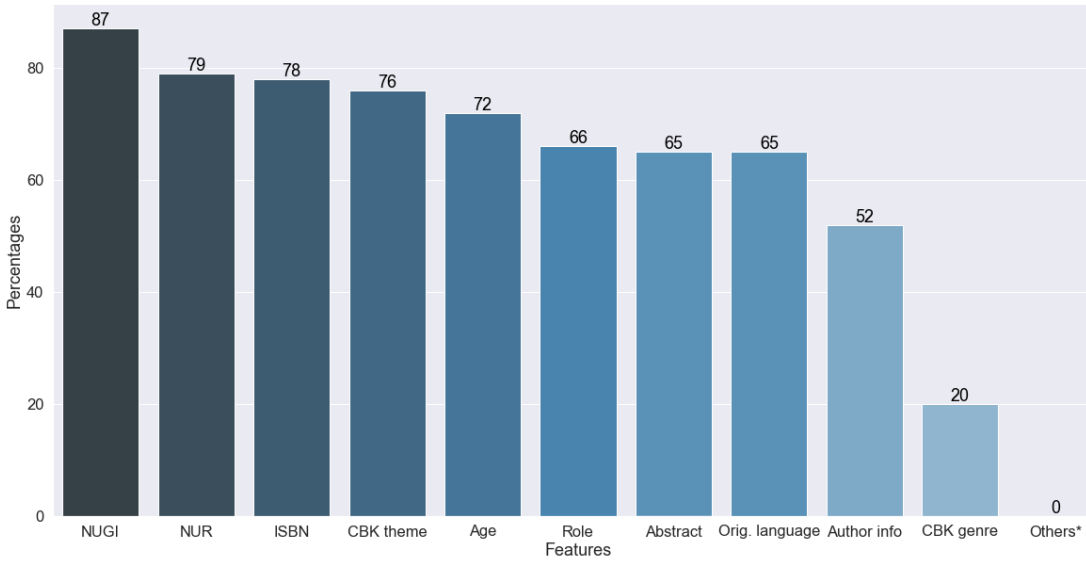

FIGURE 4.1: Percentages of missing values for each feature

Publication metadata

The NUGI genre and NUR rubric features contain respectively approximately 87% and 79% missing data, where missing data values could theoretically be any genre

or rubric depending on the publication's content. In addition it is also impossible to define a heuristic that can impute these kinds of values. However, we will consider these features for experimental use, to seek whether the low percentage of non-missing values can still have (any) positive impact on the model. However, if this is not the case then the features can be safely omitted. This is then done as the percentage of missing values is deemed too high and ultimately these features try to convey the same information as the more dense CBK genre and CBK theme features. We inspect whether filled values in either of the NUGI or NUR features have a correlation with missing values found in the CBK features, however, a potential correlation can not be found. The ISBN [2] feature also has a considerable missing value rate, but this feature can a priori be deemed as not usable as it is a unique identifier for publications.

After consultation with domain experts, we find that the CBK theme feature appears to be a further sub-specification of the CBK genre feature. The CBK theme feature is thus usually added in cases of ambiguous or shallow CBK genre data entry points. This is done to describe the book's subject more in depth. This also explains the high rates of missing values, as this feature can be seen as optional. Thus for the CBK theme feature it would not make sense to impute missing values, as the CBK genre can be considered the primary feature here for describing a book. Instead, the supportive CBK theme feature will be modeled as such for the machine learning model. When considering the primary feature, the CBK genre, we find a relatively small percentage of missing values and thus we impute these values with a 'unknown' label.

When considering the language-related metadata pertaining to the publication, a distinguishment can be made between the original language of the publication and the language after publishing. When these two fields differ in value this indicates that the publication is a translation of an original work. Usually, the language of the publication after publishing is Dutch while the original language can be any language with a majority being Dutch (as could be seen in Chapter 3). While all languages at publication are filled, the field that indicates the original language of a publication can be empty, which can also be seen in Figure 4.1. When this is the case, this (usually) indicates that the original language is the same as the publication language (thus not a translation). Therefore, original language values that are *NULL* where set to be equal to the publication language.

Lastly, regarding the publication data inputs we have the abstract feature which gives a summary about the publication. Figure 4.1 showcases a relatively high missing value rate of 65%. This feature is found in different datasets and is either in the form of a direct summary or an annotation about the book. Since this information is thematically alike with the title, and both represent the content, a new feature was created that concatenates title + abstract and additional annotations describing the content. Due to the fact that every book has a title at the very least, the missing value problem is mitigated by concatenating these different features. In turn this results into the notion that books that have more information available (content-wise) will have more input data, while publications with only a title can be compared with more confidence with those types of publications as the newly made content feature makes it possible to have more intricate comparisons.

[2] https://www.isbn.nl/

Author metadata

Outside of the publication specific data, there is also the information that pertains to the author that can be used for the second machine learning model that is based on similarity learning. The same trends are found here as well as with the publication data, as we can observe in Figure 4.1. It becomes apparent that similar high rates of missing rate values can be found for age, role and author information (i.e. notes or an autobiography that is available.).

The *age at publication* feature is derived using the birth year and year of publication features, however this results into a feature with non-computables ages for 72% of the authors. Upon further inspection in the previously mentioned NTA (the official author database) birth years or year of publications that were only partially known, such as '197X' or '18XX', were also used. These were incorporated into the computation of age by replacing the X with a 5. This is considered a sufficient replacement for X as it is in the middle of the possible 0-9 range of the placeholder and therefore is the least deviance inducing overall. Afterwards, the distribution of the values was considered as can be found in Figure 4.2. While the values have a normal distribution, there is also a significant number of outliers (mostly posthumous releases or incorrect inputs). Considering these trends in the data, the remaining missing values are imputed with the median of the training set.

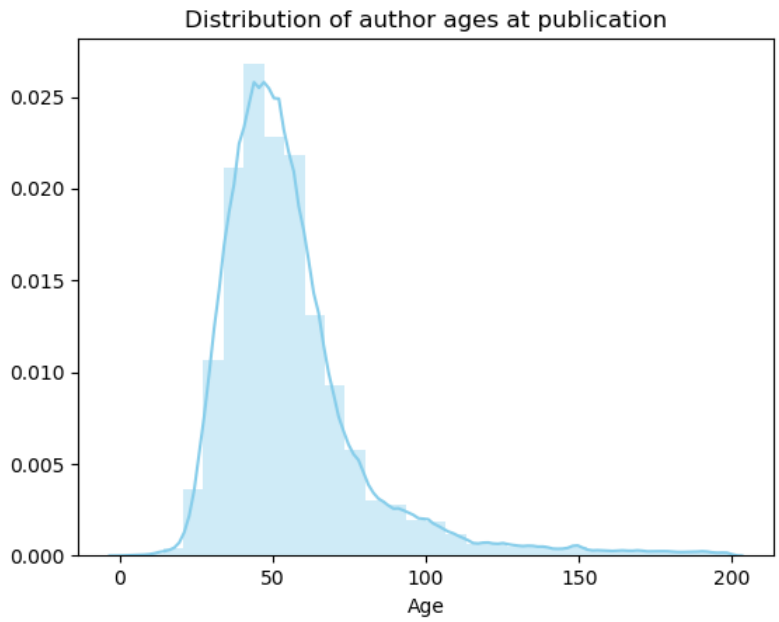

FIGURE 4.2: Distribution of ages at publication

Regarding the role feature, we can observe in Figure 4.1 that this is another feature with a high rate of missing values (65%). As this feature is solely based on domain knowledge, we consulted experts and consequentially a heuristic was made in place for value imputation. The rules for this heuristic can be found in Table 4.2.

TABLE 4.2: Heuristic for imputing role-based values

Kind	**Type of book**	**Rule**
Primary author	Picture book	Impute with the 'creator' label
	Text book	Impute with the 'writer' label
Secondary author	All	Role can take on any value.

Table 4.2 showcases that authors can be classified as either primary or secondary authors. If the author is an primary author (almost always the person who wrote/created the story of the book) a distinguishment is made whether the book is a picture book or a text book (which can be derived from the CBK genre feature). If it is the former, missing values are imputed with a 'creator' label, which is essentially a superset of the 'writer' and 'artist' labels as they are both possibilities. When it is the latter, the primary author of the book will always be a writer, and thus this label gets imputed in those cases. For secondary authors, the number of possibilities are too large as practically any label can fit here. Therefore these are imputed with an 'unknown' label.

The final feature is the author information feature, which encapsulates information related to their background. This is for some authors available but 52% of the authors do not have such information. Imputation is not possible here as biographical note(s) about authors are unique and thus must be compared through semantic analysis instead of representing it as a category.

4.2.2 Clustering publishers, CBK genres and themes

The publisher feature has practically no missing values, however the input values can be of low quality or badly interpretable for a machine learning model. This is mainly due to (1) spelling mistakes, (2) typo's, (3) variations in names of the same publisher and (4) publishers that moved to a different location in the past centuries. A few examples of these can be found in table 4.3. Table 4.3 indicates the severity of this problem by illustrating some of the found errors/variations for one of the most popular publishers, Leopold. A few examples are given, from the in total 70+ different variations of the same input value. Using this data as raw input without processing by means of clustering, will decrease the performance of the model significantly. This is due to the fact that values with different spellings but in fact represent the same publisher can not be interpreted as such. Therefore, processing was done to cluster (most of) the variations of a publisher to be represented by an umbrella and correct value. In this case, this value is the value that appears the most frequently for a publisher in the database. Thus in the case of Leopold, this will be in fact, only the word 'Leopold' without anything else. This was done through a pipeline of processing consisting of a few steps.

TABLE 4.3: Input errors or ambiguity in publisher feature

Type	**Examples**
Publisher, correct value input (by frequency)	Leopold
Spelling mistakes, typo's	Leopold Uitgevereij Lepold, Loepold
Different locations	[Amsterdam] : Leopold [Den Haag] : Leopold
Variations in name of the same publisher	H.P. Leopold H.P. Leopolds Uitgeversmij H.P. Leopold Uitgeversmaatschappij H.P. Leopold's Uitg. Mij Leopold 9+ Leopold N.V. And... 60 more variations

First off, the publisher input was processed using regex expressions and some textual processing. The regex expressions shortened the string to remove the location part in the input value and return the string that appears after the location string. This is done due to the finding that publishers with the same name, but in different locations, are approximately always publishers that moved their headquarters. The alternative could be that publishers have the same name but are situated in different cities, however, practically this is only the case in a very small subset. Due to these findings location was removed for better comparisons. After this, the leftover string was processed by making the entire string entirely lowercase and by removing all special characters. The resulting string was then compared to a list of 30+ throwaway words that appear frequently, but have very low meaning. Some examples of those types of throwaway words include 'Uitgeverij' (publisher), 'N.V.' (nameless partnership) and 'Drukkerij' (printer). These types of words were removed, yielding a pre-processed publisher string with more similar representations in comparison to other values of the same publisher.

These publishers are then compared to each other using an improved version [3] of the Gestalt Pattern Matching algorithm (Ratcliff and Metzener, 1988). This version integrates the official algorithm which searches for the longest contiguous matching subsequences, splits on these and recursively searches for longest contiguous subsequences in the leftovers until resulting strings are smaller than a specified value. The larger the found cumulative matching subsequences are (offset by the total number of characters), the bigger the similarity score will be. However, this version extends the original version by including the notion of 'junk' elements, which usually tend to be (sequences of) characters that are not interesting when comparing strings for matching sequences. These junk elements are then omitted from having a role in determining similarity between strings. This way of determining similarity between strings outperforms the alternative approach of using edit-based similarity algorithms such as the Levenshtein and Hamming distance algorithms. The latter algorithms will yield higher scores on variations of publisher names that have additional (unnecessary) words, indicating that these publishers are not similar while in reality they are the same. Ultimately after trial-and-error, a threshold of a similarity score of 0.88 was ultimately chosen for clustering publishers together. This is a relatively high number as strict clustering is preferred in this case (i.e. we want to prevent erroneous clustering). Upon visual inspection this performed reasonable, with only a few cases of underclustering being detected which were then manually clustered.

In accordance with the described method for clustering publishers, the CBK genres and themes values were also clustered. This was done as the same types of erroneous data input could be found in the latter two mentioned features. However, as these features have more bland inputs, no regex expressions or throwaway words were defined here as it would not add improvement in performance.

Ultimately clustering increased the quality of the data and reduced the number of distinct values significantly for all features as can be seen in Table 4.4. Publishers especially had a notable reduction of 58.2%, with CBK genres having a substantial reduction of 36.2% and CBK themes having a relative smaller reduction, but that could be explained by the high percentage of missing values (see Figure 4.1) that can be found in this feature.

[3] Difflib SequenceMatcher: https://docs.python.org/3/library/difflib.htmlmodule-difflib

TABLE 4.4: Distinct number of values before and after clustering

Feature	Before clustering	After clustering	Reduction
Publisher	27066	11314	58.2%
CBK Genre	696	444	36.2%
CBK Theme	5856	5084	13.2%

4.2.3 Preprocessing of text

For the content-based features of title, abstract and annotations that describe the book (which have been concatenated as described in Section 3.2.1), text has to be pre-processed differently. This is especially important in cases where corpora for pre-training are not an option (such as when using TFIDF textual representations) as syntactic meaning has to be derived, based on a corpus of the own data. Some of these techniques used for TFIDF will still apply to Word2Vec, while BERT requires a completely separate type of textual preprocessing. These various methods will be discussed in this section.

TDIDF and Word2Vec

To get useful representations of text for TFIDF representations, text is gone through a preprocessing pipeline. The diagram of this pipeline can be found in Figure 4.4. Figure 4.4 showcases that the text in its raw form goes through six different steps that essentially then give a cleaned text output. The first step is the so-called unidecode [4] step, which is a Python implementation that represents unicode textual data into ASCII for better universal representation of words in the to-be-made corpus. This is useful for words that include characters that are only used limitedly or words with umlauts, accents on characters and so forth. Afterwards, the entire sequence of text is converted to lowercase so words are not seen as different from each other based on the usage of capital letters. The third step tokenizes the input into a list of tokens and removes white spaces. This step is necessary for creating a Bag of Words corpus later on (from which a TFIDF representation can be created). These tokens are then inspected one by one for stopwords as well as non-alphanumerical characters (which are usually irrelevant special characters). The list of stopwords [5] used is compared with other lists and in comparison is the most complete and correct list. Removal of non-telling words or characters such as stopwords and special characters is essential as otherwise they will clutter up a corpus with tokens that do not provide meaningful comparison between different text inputs.

Finally, when the text has been cleaned and converted the final step of stemming is applied. This step actively changes the structure of the words with the means to group words that are the same but are in a different form or tense. For example, when you have the dutch verb 'werken' (work), you can have different forms such as 'werk', 'werkt', 'gewerkt', 'werkten' and so forth. Proper stemming algorithms will recognize that all these words convey the same meaning and reduce them to their base form of 'werk'. Several stemming algorithms are considered for this purpose but ultimately the Kraaij-Pohlmann (KP) stemmer [6] is chosen as it produces the most consistent results (Kraaij and Pohlmann, 1996). Other popular stemmers such as the

[4] https://pypi.org/project/Unidecode/
[5] https://countwordsfree.com/stopwords/dutch/txt
[6] http://snowball.tartarus.org/algorithms/kraaij_pohlmann/stemmer.html

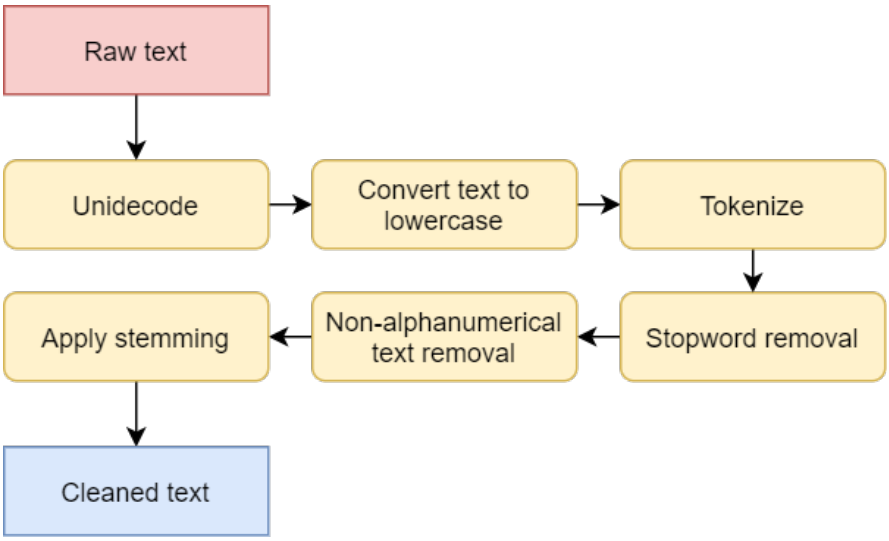

FIGURE 4.3: The pre-processing pipeline for textual data

NLTK-implemented Dutch Snowball stemmer [7] contain a multitude of unresolved issues that become apparent when used, leading to lower quality stems (Boer, 2019). Older stemmers (such as the original Porter stemmer) also tend to fail on a larger percentage of words (Porter, 2001). The KP stemmer uses a more complex algorithm than the alternative approaches (Gaustad, 2004), which becomes clear when taking an in-depth look into the source code. This leads to better results as the KP stemmer (1) aggrandizes a number of common derivational morphemes, (2) has more robust rules for plural forms and other easily detectable suffixes in the Dutch language, (3) anticipates diminutive words and (4) has spelling rules that fix some of the other stemmers' failures in regards to dealing with double consonants at the end of base forms.

Ultimately, after applying these six steps the text is processed and returns a clean set of tokens that can build a BoW corpus with richer information. For the Word2Vec approach four of the six steps are used for pre-processing, these include converting text to lowercase, tokenizing and removal of information-sparse words (stopwords) and characters. Unidecode and stemming are both not applied as they modify the structure of words, which is not necessary considering the Word2Vec implementation is based on word embeddings that are the result of a pre-trained model on millions of articles. Thus, this means that words are used in contexts which makes it possible to derive their semantic meaning, contrary to TFIDF. Applying techniques such as stemming can even decrease performance as it can create words that do not appear in the pre-trained corpus, making it impossible to compare between different text inputs. This is further discussed in Section 4.5, where a look is taken at how Word2Vec embeddings are derived and how accordingly different publications are semantically compared.

BERT

BERT has its own set of rules for text input, and thus the texts that represent the content of the book have to accordingly be processed. This pre-processing structure can be found in Figure 4.4. A sentence is usually tokenized in similar ways as with TFIDF, but with dynamic word embeddings that use language models such as BERT,

[7] https://www.nltk.org/_modules/nltk/stem/snowball.html

a more inclusive tokenization also known as *wordpiece tokenization* is applied. This can be seen in Figure 4.4 with the word 'rumination' that is tokenized into rum and ##ination. Effectively this increases the model's performance to deal with language-specific Out-Of-Vocabulary (OOV) words, which are essentially words that are rarer or do not appear in the corpus. Through this way a more frequent subword will appear during training, leading to the model's recognition of these rarer words and learn about its nature. Furthermore, in line with Figure 2.5 in Chapter 2 explaining BERT's NSP mechanic, a [CLS] and a [SEP] tag are added. Finally, the tokens are substituted with their IDs, which translates the text now as a valid input for the BERT model.

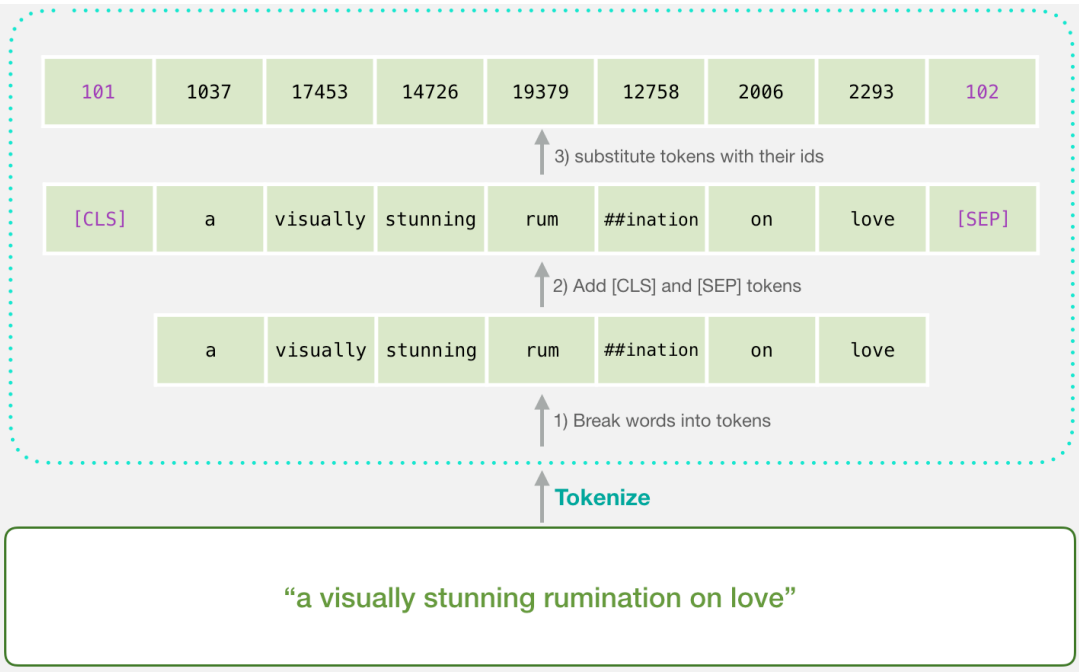

FIGURE 4.4: BERT's own pre-processing structure

4.2.4 Integration of linked data

To enrich the information density of the available data and add supplementing information, a final option to consider is the usage of external linked data, as discussed in Section 2.3.1 *Usage of external knowledge*. For this purpose several databases are considered: the previously mentioned Nederlandse Thesaurus van Auteursnamen (NTA), Wikidata [8], Digitale Bibliotheek voor de Nederlandse Letteren (DBNL) [9] and Worldcat [10]. The NTA houses (practically) all Dutch authors that have one or more publications as well as metadata describing the author. In a somewhat similar nature, DBNL and Worldcat are digital libraries that house information about Dutch (DBNL) and worldwide (Worldcat) publications, articles and their associated authors. Lastly, Wikidata is a general knowledge base that has structured data about any subject or concept found in the Wikimedia [11] cluster of knowledge bases and encyclopedias. The more general nature of Wikidata can be a benefit, as Section 2.3.1

[8] https://www.wikidata.org/wiki/Wikidata:Main*page*
[9] https://www.dbnl.org/
[10] https://www.worldcat.org/identities/
[11] https://commons.wikimedia.org/wiki/Main*page*

Usage of external knowledge section showed great potential for integrating Wikidata. The different databases were observed and can provide some useful information in addition to the already existing heterogeneous data. Table 4.5 showcases this type of information, while also showcasing the percentages of authors in our dataset that have a record available in the different libraries/databases.

TABLE 4.5: Various statistics regarding the different sources for integration of linked data

	Authors with a record	**Available information of interest**	**PPN**
DBNL	7%	Birthyear, gender, author notes	No
NTA	100%*	Birthyear, autobiography, role	Yes
Wikidata	18%	Birthyear, gender, occupation	Yes
Worldcat	87%	Birthyear, publication statistics	Yes

Accessibility to NTA data is provided by the KB, while Wikidata, DBNL and Worldcat are accessed through an API and data is consequentially scraped with self made scripts and saved locally. Links to Worldcat and Wikidata (via Wikipedia) were automatically generated through the universally used VIAF identifiers [12].

When using the data it becomes apparent that 3 out of 4 sources do not supplement information sufficiently. DBNL and Wikidata have insufficient amounts of information available for enrichment. Even within the considerably low percentage of authors that have a record in these knowledge bases (7% and 18% respectively), the vast majority of these authors do not have the typical heterogeneous data problems in the children's literature publication database or in the GGC [13]. This is mainly due to the fact that more known authors, and thus associated with more information-rich data, are the authors that tend to have an external record. Furthermore, DBNL does not have a PPN (unique author identifier) yet for linkage with authors in the database nor with the NTA, which makes it infeasible to use this type of data either way. On a similar note, while Worldcat tends to have a publication record for a significant percentage of authors, the quality of the information resembles the original database. Many of the heterogeneity problems can be found here as well. Worldcat provides some publication statistics as well as the birth years that can be potentially interesting. Publication statistics are mainly basic information types such as publication timeline and list of associated titles. However, features such as the 'year of publication' and 'title' feature in the original database are some of the few features that have practically no missing values, making this type of data from Worldcat redundant (as discussed in Section 3.2.1). Regarding birth year, the NTA has significantly more birth years available for usage, which makes this source a preference for integration over Worldcat. Ultimately, only the NTA provides a useful integration of external knowledge and is therefore used. The rate for the authors with a record in the NTA is also 100%, as this is the most complete external database in relationship to the children books dataset. However, some authors in the children's book database do not have a PPN (mostly due to corrupt data), which technically implies that not all records in the NTA could be linked. As the PPN is fundamental in functioning as a label for segregating authors with the same name, the authors missing a PPN were omitted from usage.

[12] https://www.oclc.org/nl/viaf.html
[13] http://support.oclc.org/ggc/richtlijnen/?id=12ln=nlsec=k-3000

4.3 Feature Engineering

Extracting features from the available (processed) data is a fundamental step in the machine learning pipeline. For elaborating upon the decisions that we will make here, we will use the distinction as made in Section 2.3.1. Thus, this section will be divided into two main parts of feature engineering: 1. engineering features that provide (new) additional information or insights about the data (in Section 4.3.1) and 2. engineering features that make data compatible with the requirements of machine learning models (in Section 4.3.2).

4.3.1 New features

The available data encourages to make new features as several author characteristics can be found that are not modelled as a feature of their own. With this notion in mind 8 features were engineered:

1. number of authors
2. number of words in title
3. number of characters in title
4. length of title in ranges (categorical representation)
5. mean word length of publication title
6. median word length of publication title
7. age at publishing for every possible author (with the same name) and publication combination
8. concatenated content feature

Age and concatenated content feature

The 'age at publishing' and 'concatenated content feature' have been discussed in Section 3.2.1 *Considering missing values*, which showcased their role in alleviating discrepancies caused by missing information. The age at publishing feature was constructed with implementing the simple subtraction of *year of publication - birth year*. This feature was added due to the notion that age can tell how likely it is a certain author wrote the book, or completely remove the possibility that a certain author wrote the book. An example for the latter is when the author's age turns out to be negative when comparing different ages of authors for a certain publication. As the author was not born, the publication could not have possibly been written by them. The feature can also tell when it is less likely, i.e. the author is a teenager or an elderly person. The distribution of age in Section 3.2.1 Figure 4.2 gives some insight into the probabilities of writing. A machine learning model can learn upon these probabilities and notions and integrate age for deciding how likely it is that a certain author wrote a certain publication.

The concatenated content feature is created by combining all available text, descriptive of the publication (including the ingrained stylistic components, as it is written by the author of the publication). The rationale for this has been addressed and is explained more in depth in Section 3.2.1 *Considering missing values*.

Statistical features

Different statistical measures have been shown to successfully make distinctions between authors based on their style of writing for classification purposes (Litvinova et al., 2016; Pervaz et al., 2015; Ostendorff et al., 2019). With this in mind, several statistics that are directly tied to the way authors write or operate were computed with the available data.

One of such features is the 'number of authors' feature, which can be telling as authors can have a preference in working alone, with a partner or multiple other people in regards to the publication. This type of information conveys characteristics of authors that the model can use for classifying purposes. The number of words and characters in title features can also be distinctive of authors; some might have typically short titles while others can have considerably long titles. In the same vein, the mean and median word length in the title are calculated and saved as new features. To illustrate the use of these types of features, we compare different (types of) authors. For example, authors that create picture books without text have an average title that consists of 3.8 words and 23 characters, while authors that write religious books have an average title of 6.3 words and 40 characters. On an individual basis these differences can be found as well, when comparing authors within a genre (such as picture books without text). For example, children's literature writer Ali Mitgutsch has an average title consisting out of 6.7 words and 36 characters while author Dieter Schubert has an average title consisting of 2.8 words and 16 characters. These statistics indicate significant difference between authors and their way of writing. With these significant differences in mind, this shows that these values can generate distinctive features for a machine learning model to make predictions upon.

Finally, the 'length of title in ranges' feature is a categorization of the different lengths of titles, as it is not likely that an author will always write a title with the same numbers of characters but rather in close proximity. Thus, this feature is instead of a numerical feature, a categorical variation that might increase performances as there is more leeway to represent an author's writing style. In addition, this representation is also more useful for the second machine learning model that operates in the similarity space.

4.3.2 Representing data in machine-interpretable forms

As earlier described, the available data comes into three forms: categorical, numerical and textual data. In this section we will discuss the appropriate representations for different types of variables. These types of representations are mainly meant for an author classification model that operates in the publication with author as label space.

One hot encoding

A number of the features are of categorical nature, where each publication can only have one unique value/category assigned to. These are the following features: language of publication, original language, country of publication, NUGI genre and NUR rubriek. For these types of features one-hot encoding [14] is used for representing the data. One hot encoding is an approach that turns categorical features into a one-hot numeric array. This is an array filled with two possible numerical values,

[14] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

either 0 or 1. A value of 0 indicates that this category is non-existent for this publication, while a value of 1 indicates that this publication is of this category type. This is an essential step for machine learning models that want to use categorical variables, as their 'label forms' can not be interpreted well. A machine learning model will interpret such labels, for example 'Dutch' and 'English' for language of publication, as numbers that are lesser or greater than each other. This leads to faulty predictions as these types of categorical values do not have an ordinal relationship. Using binarization through means of one hot encoding fixes this problem and gives the model a staunch idea of what to categorize the publication as. Through this way a set of new features are created in the form of 'is_x' where x is a distinct value. For example for the language feature, the newly created features are named 'is_dutch', 'is_english' and so forth. The number of new features created this way is proportional to the number of distinct values found within the different categorical features. For all the named categorical features combined this yields 1308 one hot-encoded features.

Count vectorization

There are also categorical features that can assign multiple categories to a single publication. In such cases one-hot encoding will not work as it only supports one level per publication per category. These types of features are the CBK genre and CBK theme features, where a publication can have up to four different categories assigned. For these two features a different solution is used in the form of a count vectorizer [15]. In this case the principle stays the same as with one hot encoding, however the numerical arrays consisting of 0 and 1 are now not confined anymore by having only a singular 1 in the array for a certain publication. The count vectorizer will 'count' all the categories that apply to the publication and give these a value of 1 and then accordingly make a vector where the irrelevant categories get a value of 0.

Reducing dimensionality with compressed sparse row matrices

For features that have many distinct values after pre-processing, such as the publisher value or the TFIDF matrices output, the data is represented with compressed sparse row matrices (CSR). The rationale behind this is that some of the mentioned pre-processing techniques can yield over ten thousand columns, mostly consisting of 0 values for the publications. Thus, representation of data as sparse matrices leads to significantly more computational efficiency as well as memory usage, as zero values are omitted from the representation while their meaning is still (implicitly) taken in consideration. Representing data as dense does not significantly change predictive performance for the experiments conducted in this thesis, however, at the same time does in fact significantly increase computational resources and memory usage. This means that CSR from both perspective should get precedence for usage.

Latent semantic analysis with truncated singular value decomposition

Latent semantic analysis (LSA) is used for the title and content features, in the context of TFIDF representations. LSA operates through truncated singular value decomposition (SVD), with the goal of simulating meaning. This is a form of dimensionality reduction, and in this case means that multiple, different words sharing a similar meaning should be represented as a lower dimensional entity encapsulating

[15] https://scikit-learn.org/stable/modules/generated/sklearn.feature$_{e}xtraction.text.CountVectorizer.html$

their meaning. Truncated SVD uses matrix decomposition to essentially reduce the overall matrix to its principal parts. This decomposition is done by splitting Matrix M into the product of 3 separate matrices: $M = U * S * V$, with U and V being orthonormal matrices and S a diagonal matrix of the singular values of M. Obtaining singular vectors and singular values as the result of this allows identification of information found as with eigendecompisition. The eigenvectors that correspond to the largest eigenvalues can then be used to convey a substantial part of the variance found in the original data. In a practical sense, this can lead to useful data reduction that has shown to have effective and efficient uses in machine learning (Wall, Rechtsteiner, and Rocha, 2003). In addition, truncated SVD works well with with the earlier discussed sparse matrices representation (unlike the non-truncated PCA alternative which would require dense data).

The idea of applying SVD to TFIDF matrices (LSA) is somewhat similar as what semantic word embeddings such as Word2Vec try to establish. However, these techniques differ fundamentally, as LSA assumes that the semantic similarity between words does not exist explicitly but exist through so called *latent* variables that can explain how similarly different words affect passages in which they occur.

With some testing, approximately 1000 components showcase to have the best performance for LSA. These components represent the 1000 largest singular values and the first 1000 columns of U and V. In addition, the original TFIDF representation and scores are also kept as features to seek if there is an performance increase between the two versions.

Standardization of numerical features

As previously discussed, the missing values problem in numerical features have been imputed through usage of the median after considering the distributions of the values. This yields numerical features that have no missing data and are represented by integers and floats, which theoretically can be interpreted by the model. However, it is a better idea to standardize the data. Standardization of the data is recommended as all the features are measured at different scales. Essentially these intrinsic differences mean that they will not contribute equally to the model's fit nor the model's learned function. This could lead to worse predictive power or bad behaving models (Raschka, 2014). Thus to prevent this, standardization is applied for every feature. Standardization will normalize every feature by scaling them in such a way that they all have a mean of 0 and a standard deviation of 1, which can be done with Equation 4.1.

$$z = \frac{x - \mu}{\sigma} \tag{4.1}$$

In the standardization equation, μ is the mean and σ is the standard deviation, while x is the value for the numerical feature of publication x. The standardized value z will then replace the original values for publication x for all the numerical features and provide the well-adjusted input for the model.

Classifier model versus similarity model

The described steps suffice for the first classifier model, however, additional conversion of *all* data is needed for the second model. As this model operates in a similarity space, this space has to be created from the author and publication metadata spaces. This will be discussed in Section 4.4, as this warrants a section of its own.

4.4 Modelling the similarity space

A main research question of this thesis is whether using similarity learning can enhance predictive modelling performance for the author attribution task (as discussed in the Introduction and Section 2.3.2). A conversion pipeline is needed as the available data is naturally in a publication and author metadata space. Essentially, this requires three steps:

1. convert data to a form that includes the possibility of all authors with the same name for a certain publication
2. create author embeddings based on x% of previous work (where x is the percentage of training data in the training-test split ratio)
3. calculate similarity between the publication metadata and author embeddings for all features

The first step can be done by retrieving all authors with the same name as the author of the book. All possible authors are then linked with the publication record. The correct author gets a target value of 1, while the incorrect authors get a target value of 0. An example of this training structure can be found in Figure 4.5. In Figure 4.5 an example of the book 'Het spookhuis' (translates to 'haunted house') and its author Tromp is showcased, as well as all potential authors named Tromp. In the author_ppn field their unique author number is displayed as well, which is key in differentiating between authors with the same names. With this training structure, all relevant publication metadata for that publication can be accordingly retrieved from the database and used for similarity comparisons. In addition, the earlier mentioned types of author information, can now also be retrieved for all authors and compared with the publication metadata.

titelvermelding	last_name	target	author_ppn
Filter	Filter	Filter	Filter
Het spookhuis	Tromp	1.0	072432950
	Tromp	0.0	072719451
	Tromp	0.0	068229496
	Tromp	0.0	073050687
	Tromp	0.0	147848253

FIGURE 4.5: Training structure of the data for conversion to the similarity space

Proceeding with step 2 of the pipeline, embeddings are created of all authors and stored. For this a training-test split is used (that will also be used later for training the machine learning model) where 75% of the data is labeled as training data. All previous work in this training split of the author are retrieved and all values associated with the authors for the different features are stored in their personal embedding. The personal embedding then summarizes these inputs by giving a count to each value found for a certain feature for the author.

With the creation of all embeddings and implementation of the correct training structure, the similarities can be calculated. This is done using the cosine similarity

algorithm, which is a measure of similarity considering the inner product space of the two non-zero vectors. The equation for this measure can be defined as following:

$$\cos(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{AB}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} \mathbf{A}_i \mathbf{B}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{A}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{B}_i)^2}} \tag{4.2}$$

In this case the two non-zero vectors of A and B in the equation, consist out of the author embedding with its value/term frequencies normalized for the feature (A) and the normalized values of the feature of the publication (B). The cosine of the angle between these two vectors is thus calculated for every possible publication-author combination. This yields similarity values ranging from 0 to 1. A value of 0 means no similarity at all with increasing similarity as the value increases. Thus, a value of 1 means identical values. This also means that all previous publications of a certain author have an equal impact when determining similarity. These similarity values are calculated for every individual feature. Combining all similarity scores for every author-publication combination for every feature creates the similarity space. This similarity space is thus a matrix of the number of possible publication-author combinations (based on the number of names that are identical with the actual author of the publication) multiplied by the number of features. Accordingly, the similarity space can now be used as input for the regressor similarity learning model.

4.4.1 Calculating similarity between content related features

Most of the features have values that can be counted or labeled that can culminate into an embedding that represents all the according values for an author by summing the values for each occurring label, as well as their interrelationship. This can afterwards be directly used for comparison with cosine similarity. This is the case for all features except for the textual (content) features. However, the content features have to be converted to the similarity space as well as they introduce theoretically useful features. For doing this, two out of the three text representations (TFIDF and Word2Vec) as discussed previously are used. BERT has not yet been found suitable for training in calculating similarity in sentence(s) (Devlin et al., 2018). Using both TFIDF and Word2Vec gives insight about different text representation usage (term weighting scheme vs semantic word embeddings) for similarity learning. For both implementations, every previous publication found in the embedding (title and the concatenated content feature) is compared with the title and content of the publication. For this purpose, the paragraphs are tokenized into sentences and afterwards tokenized into words.

For calculating similarity through TFIDF, the publication to be attributed as well as a previous publication of a relevant author (in their TFIDF representations) are compared for the entirety of the vectors. This yields a similarity value for every comparison, and afterwards different metrics can be used to determine the similarity (to combine all these individual values) of the author in comparison to the publication. Some metrics include the usage of the (1) average value, (2) the max value or (3) using the top k% titles with the most similarity with the publication. For this thesis we test with the average value and the top k% titles metrics as they give both perspectives: similarity scores based on an author's total bibliography as well as a similarity score based on only the most similar scores. Both metrics can be useful as the average can indicate how often the author has written similar books, while the top k% metric can offset for authors that write about many different subjects. Upon

experimenting, the average value metric produces a space that the machine learning models ultimately 'deem' more useful to learn upon in our experimental setup. This is tracked by calculating the feature importance and permutation importance for both features created with the different metrics. Thus, the average value metric is ultimately used for the final results.

A different approach is used for Word2Vec. A neural network is trained using a pre-trained Wikipedia corpus considering approximately 25 million Dutch articles. This pre-trained Wikipedia corpus has empirically performed well on some similarity-derivation tasks between sentences (Tulkens, Emmery, and Daelemans, 2016) and produced reasonable results for a set of test cases in personal use. In addition, the usage of a much larger (external) corpus produces more elaborate word embeddings that will provide better performance. Thus, these word embeddings are used to compute similarity between the sentences of the publication's content and the authors' previous publications. For this purpose the average vector for all words in every sentence is calculated, and cosine similarity is afterwards used to calculate similarity between these vectors for every comparison between texts. This yields the similarity values that indicate the semantic similarity between an author's past writings and the publication. In line with TFIDF, both the average metric and the top k% metric are tested upon for the purpose of summarizing the similarity values. In accordance with TFIDF, the average metric performs here better as well.

Noteworthy is that there is a feature that can only be semantically compared with the content of the book, which is the autobiography feature. This can showcase whether the subjects of a publication are in accordance with the author's expertise, interests and so forth. Thus this means that the Word2Vec similarity implementation is solely used for this feature.

4.5 Machine learning models

A set of different machine learning models are used based on the type of experiment. As Section 2.1 showcased there are a multitude of viable options present for integrating machine learning for authorship attribution tasks with skewed/heterogeneous data. The types of models can be mainly categorized into three categories: linear predictive modeling, ensemble learning and neural networks. Within these categories the type of predictive modelling can be divided as well into classifiers and regression modelling types which are deemed both relevant due to the differing spaces of operation.

Precedence for implementation of a classification model is given due to the fact that all publications (and associated metadata) are labeled by an author (class). This type of structured supervised data with labeled classes is prototypical for classifier models, which in this thesis will define experiment 1.

On the other hand, the regressors will operate in the similarity space (see Section 4.4). A continuous output variable (y) from the similarity inputs will be predicted through a mapping function and have a value between 0 and 1 (approximately). Unlike with the classification task, this output will not be a label but a real-value float. The predicted value will depend on how similar the publication is deemed to be in relationship to a potential author's previous works, with a higher value indicating a more plausible match. The implementation of these types of machine learning models with their associated research questions will be defined as experiment 2.

Finally, there is also experiment 3 that is going to consider different text representations for the textual content features. This is an experiment on its own as the

regular data has dozens of non-textual features that are relevant as well but do not tell us much about how we can use the actual textual content of the publication for author attribution. As defined in Section 2.2 we will look at three different text representations TFIDF, word embeddings (Word2Vec and fastText) and BERT for this purpose.

For these three different purposes varying machine learning algorithms are used to seek the best intrinsic performing model for our task. The list of used algorithms can be found in Table 4.6. These algorithms have mostly been mentioned or discussed in Section 2.1 and Section 2.2. In addition, we choose a wide set of different algorithms to cover most of the machine learning intricacies found within the algorithms. This also provides us a more substantiated comparison. We will use the standard parameters in this comparison of algorithms, as initialized by their respective Python libraries. However, we check for each algorithm whether a parameter needs to accordingly be changed if its standard parameter configuration does not fit with the heterogeneous input data or the CSR representation. Accordingly, we will tune the hyperparameters for the best performing model using a grid search optimization implementation (which will be discussed in Section 5.2 and Section 5.3).

TABLE 4.6: The used machine learning algorithms and text representations for different experiments

Experiment 1 (Author classification)	**Experiment 2 (Predicting similarity)**	**Experiment 3 (Text representations)**
SGD learning	Linear regression	TFIDF (NN)
K-nearest neighbors	Lasso	Word2Vec (NN)
SVM	ElasticNet	BERT (NN)
Nearest centroid	K-nearest neighbors	
Logistic regression (classifier)	Logistic regression	
Naive Bayes	Bayesian ridge	
Decision trees	SGD regressor	
Random forests (ensemble)	Decision trees	
Gradient boosting (ensemble)	Gradient boosting (ensemble)	
AdaBoost (ensemble)	AdaBoost (ensemble)	

4.5.1 Experimental setup

For evaluating the performance of the models on unseen data a training-test set split with a 75-25 ratio is used. This split is done by randomly sampling different publications to either set. The pipelines for the different experiments/models as thus far described are fitted on the training data with the labels/output variables, and accordingly transform the data. The resulting model which is learned upon is then used for making predictions. The test set is used for inference and to make predictions for never seen data. This simulates a real world setting and thus give insight in how the model will realistically perform and generalize, as well as reveal if the fit is adequate (i.e. no overfitting and selection bias).

Furthermore, as the database consists of nearly 250.000 records, the experiment conducted in the similarity space can become considerably computationally complex as all combinations for a certain publication with authors with ambiguous names have to be inferred. For this purpose the number of possible combinations is scoped to ambiguous names linked to between 5 and 20 authors. This means that publications that could have been written by 5 to 20 possible authors (due to

having the same name) will be learned and tested upon. The distribution of this set of authors can be seen in Figure 4.6. Figure 4.6 indicates a relatively skewed distribution towards publications with fewer potential authors (names that are less ambiguous than others). However, when we normalize these publication numbers for the number of total instances of each type of name in the used data, a more uniform distribution is approximated. When considering the instances that are actually learned upon, publications linked to 17 potential authors are the biggest cut of the data.

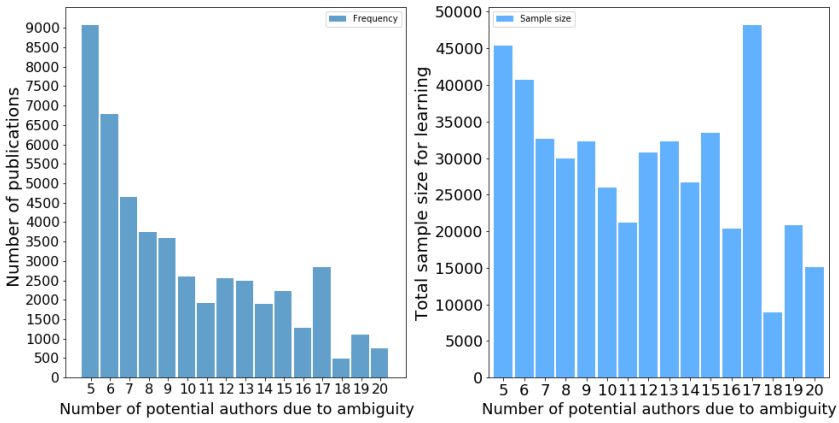

FIGURE 4.6: Distribution for authors with ambiguous names for (A) publications and (B) normalized for total number of instances associated with that number of potential authors

4.5.2 Evaluating metrics

A set of metrics is used to evaluate the models. Accuracy (the percentage of correct predictions) will be used as a general metric to give an indication of the quality of the predictions. Since accuracy can potentially be misleading with skewed data, this metric will be supplemented with the precision, recall and f1 score metrics (Goutte and Gaussier, 2005). Precision and recall give an indication of the predictive performance regarding true positives and sensitivity, respectively. Combining these two metrics introduces the useful metric of F1 Score, which gives a harmonic middle ground regarding how precise and robust the constructed model is in its totality. For a ranking context, which is provided by experiment 2, this can be done for the top k predictions. In this context a precision and recall set at k = 1 will prove to be useful for comparison purposes. This is due to the fact that the output of the classification model will give labels as prediction (one item), while the regressor will give the most likely match at the number 1 of the ranking.

Recall scores in the context of this thesis will give an indication regarding the number of authors correctly attributed to a publication, as a fraction of the sum of true positives and false negatives (publications where the correct author was not attributed). Precision will then additionally give insight in how many times the model attributed an author to publications they did not write, by considering the true positives (correctly attributed authors) as a fraction of the sum of true positives and false positives (erroneously attributed authors). In the task of disambiguation this becomes mainly interesting due to the fact that we can observe this within clusters of authors with the same name. For example, if we have 20 potential authors with the name 'Bruna', we can observe for each author with this name (A) whether

this author's publications were correctly attributed and (B) whether this author was attributed to publications of other authors with the same name. Considering the scores for every author (as determined by considering all publications associated - correctly or erroneously - with that author), we can then consequently calculate the recall and precision scores on average. These average scores will give an indication of the model's performance for the entire test set.

To get a more accurate representation the metrics will be calculated using weighted and macro performance. We can define these two types of performances as following:

1. **Weighted**: averages the support-weighted mean per label
2. **Macro**: averages the unweighted mean per label

Essentially, this means weighted performance will gravitate to the most populated classes, while the macro metric will gravitate to the least populated classes. The distinction is useful as it showcases the performance on authors with many publications as well as authors with only a few publications.

In addition, recall at larger k's (3,5 and 10) will also be calculated for the model implemented in the similarity space. Since similarity learning gives us the option to get a ranking out of the results, this will be explored as well. This is primarily done to determine the performance of a model in a ranking context and to determine whether modelling the predictions with a ranking is a more appropriate solution for the authorship attribution task.

4.5.3 Technical implementation

The code and technical implementation as used for the experiments in this thesis can be found on GitHub [16]. For the models of experiment 1 and experiment 2, the technical implementation follows the theoretical properties of Section 2.1.3 after accordingly implementing the machine learning pipeline as described in Section 4.1-4.4. Section 4.4 is exclusively used for the second experiment, which culminates into a similarity space with a numerical target variable (similarity score). The conversion of the data to this form means that the data can now be used by the mentioned regression machine learning models, as well as showcase similarity on a feature-basis.

For experiment 3, BERT is implemented using its text classification module, which is a deep neural network model based on the theoretical properties as discussed in Section 2.1.4 and 2.2.3. We use a monolingual and a multilingual model for this purpose, as recent research shows that there can be a significant difference in performance between the two types of models (Vries et al., 2019). The BERT multilingual (also known as mBert) pre-trained model (12-layer, 768-hidden, 12-heads, 110M parameters NN) is used for the purpose of deriving BERT embeddings for the multilingual model, which supports Dutch text inputs and has empirically performed well (Pires, Schlinger, and Garrette, 2019). For the monolingual model the Dutch Bertje model is used, which is also a 12 layered cased tokenization model with an identical number of parameters and hidden layers as the multilingual model. Bertje is chosen for this purpose as it currently holds state-of-the-art performance on various NLP tasks when using Dutch texts (Vries et al., 2019). In a similar nature, we use the neural network architecture for fastText as described in the official paper (Grave et al., 2018).

[16] https://github.com/KBNLresearch/Demosaurus/tree/kinderboeken/ML_Nizar

For Word2Vec and TFIDF we build a text classification (multi class) neural network after deriving average word vectors and TFIDF features, respectively. For Word2Vec we choose to use the CBOW underlying neural network architecture as it operates more efficiently, while the Skip-gram architecture is too cost-intensive to use in practice. For extracting word embeddings a pre-trained neural network is used that trains on an extensive Corporafromtheweb (COW) [17] Dutch corpus. The COW corpus performs empirically well for classification and outperforms other Dutch Word2Vec based models, such as models trained on Wikipedia (Tulkens, Emmery, and Daelemans, 2016). We find these results as well, when replicating the used experimental setup in the cited study for testing the performance of the COW corpus in comparison to the other corpora for our specific task. Due to the size of the COW corpus the subsequent word embeddings lead to the smallest percentage of OOV-words when using children's literature as input. For the same reason the COW corpus leads to word embeddings with the most accurate relationships between semantically similar words. The word embeddings are consequentially used to create average word vectors of the raw publication text input. The average word vectors and TFIDF features are then fed through a feed-forward deep neural network with a multi class text classification softmax output layer. These architectures have been schematically drawn and can be seen in Figure 4.7.

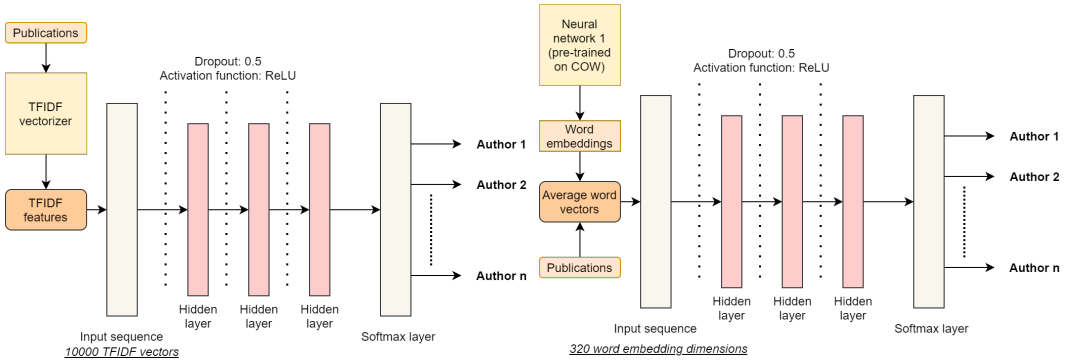

FIGURE 4.7: Implemented neural network architectures for TFIDF (left) and Word2Vec (right) based text classification.

4.5.4 Optimization

For optimization, we introduce the concept of using a validation set. The inclusion of a validation set helps us (1) find the best machine learning algorithm for the task at hand and (2) tune the hyperparameters of the consequential best performing model. We do this by applying K-fold cross validation, which is a model validation technique for analyzing the predictive performance of a model on one or more folds. We apply k-fold cross validation on the earlier defined training dataset, which means that this dataset is kept separate from the unseen test set (to prevent data leakage). We set k at 10 folds to reduce variability and give a more substantiated and accurate representation of performance. Accordingly, we combine the validation results to give an overall estimate of the predictive performance for different models and parameters, and ultimately the model and parameters that give maximal performance are kept. These will be used for further purposes as well as answering the research questions.

[17] https://corporafromtheweb.org/

Chapter 5

Results

In this chapter we compare the models within the experiments and observe their predictive performance. In these comparisons we also take a look at how the different types of (processed, engineered and added) features influence the predictive performance. We also consider how human experts perceive the importance of the features as the result of the conducted case study and survey. In addition, we make an inter-experimental comparison to seek which implementation methodology for learning as well as making predictions should get precedence when considering class imbalance and heterogeneous data problems. Finally, we also take a look at different text representations and how these compare with each other in a more NLP-oriented context that purely focuses on textual input.

5.1 Baseline performance

Two types of baseline models are created to establish a non-AI reference for comparison with machine learning models. These baseline models are based on the notion of disambiguating a set of authors for authorship attribution by using simplistic heuristics that could potentially pick the right author sufficiently on a consistent basis. For this purpose we implement models based on the following heuristics:

1. Picking the first author in a list of authors with the same name
2. Picking the author with the most publications in a list of authors with the same name

The performance for both models can be found in Table 5.1. We can observe that all weighted scores are higher than their macro counterparts. Furthermore, recall is always significantly higher than precision. Overall a F1 score of 0.36 is achieved for the first author model and up to 0.55 for the model that selects the author with the most publications. Regarding macro performance the baseline models perform drastically worse (F1 scores of 0.10 and 0.14). This difference in performance between the two metrics can be alluded to the fact that the used heuristics will get

TABLE 5.1: Performance of baseline models as measured in precision, recall (= accuracy) and F1 score.

Heuristic	**Scoring type**	**Precision**	**Recall/Accuracy**	**F1 Score**
First author	**Weighted**	0.33	0.45	**0.36**
	Macro	0.08	0.20	**0.10**
Most publications	**Weighted**	0.49	0.66	**0.55**
	Macro	0.11	0.20	**0.14**

biased towards getting classes with many instances right which will in turn lead to better weighted performance. At the same time the baseline models do not have a mechanism included that can predict authors with a lower number of publications well (as this can only happen if it is by chance) leading to lower macro performance. In addition, when considering all data as described in the experimental setup, publications with fewer potential authors (due to fewer authors sharing the same name) primarily define the test set, thus we also calculate individual performance on more difficult test sets. The results of this comparison can be seen in Figure 5.1 for both baseline models, for all metrics.

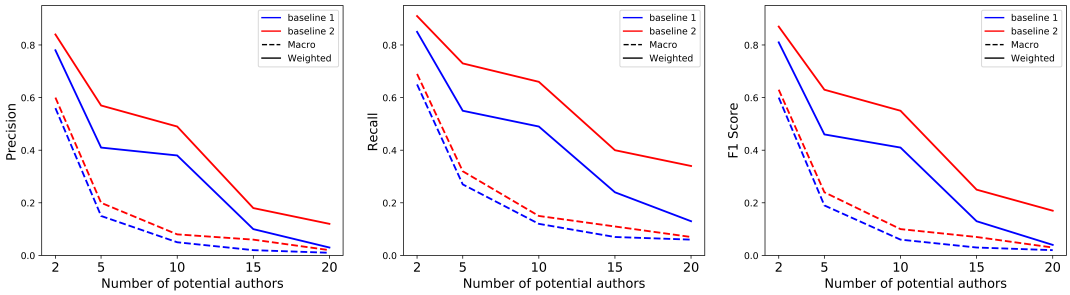

FIGURE 5.1: Performance of baseline model 1 and model 2 for different test sets, as calculated in precision, recall and f1 respectively.

The results of the evaluation for baseline model 1 and 2 show that performance significantly decreases for both models, as the test set tends to emulate more difficult authorship attribution choices. Gradually, F1 and precision reach scores of zero while the trend of recall being higher can also be found here (however also significantly declining). The baseline models perform better when considering weighted performance, however, the decline in performance when going from 2 to 5 potential authors is significant. Afterwards the models tend to keep declining in weighted performance as well. Figure 5.1 showcases that the patterns between the two baseline models are identical. Although, we can observe that baseline model 2 tends to perform better on all test sets for both macro and weighted performance. The difference between the baseline models is especially noticeable regarding weighted performance while difference between macro performances is minimal for all test cases. This difference can logically be attributed to the fact that baseline model 2's heuristic will statistically lead to more accurate predictions. When selecting an author with more publications than all other authors in a list of potential authors this will naturally increase the chance to pick the right author. Which becomes apparent when comparing with baseline model 1 that picks an author without any real statistical substantiation. This is due to the notion that the first author in a list of potential authors (in this context) is purely random.

5.2 Experiment 1: Author classification

The first experiment consists out of modelling the problem as an author classification task. For this experiment mainly textual input substantiated with contextual metadata is used and the models as described in Table 4.6 are used. For deriving the best performing model in this case, the set of machine learning models have been compared with the previously described 10 fold cross validation procedure to find the best performing model. The results of the comparison can be found in Figure 5.2.

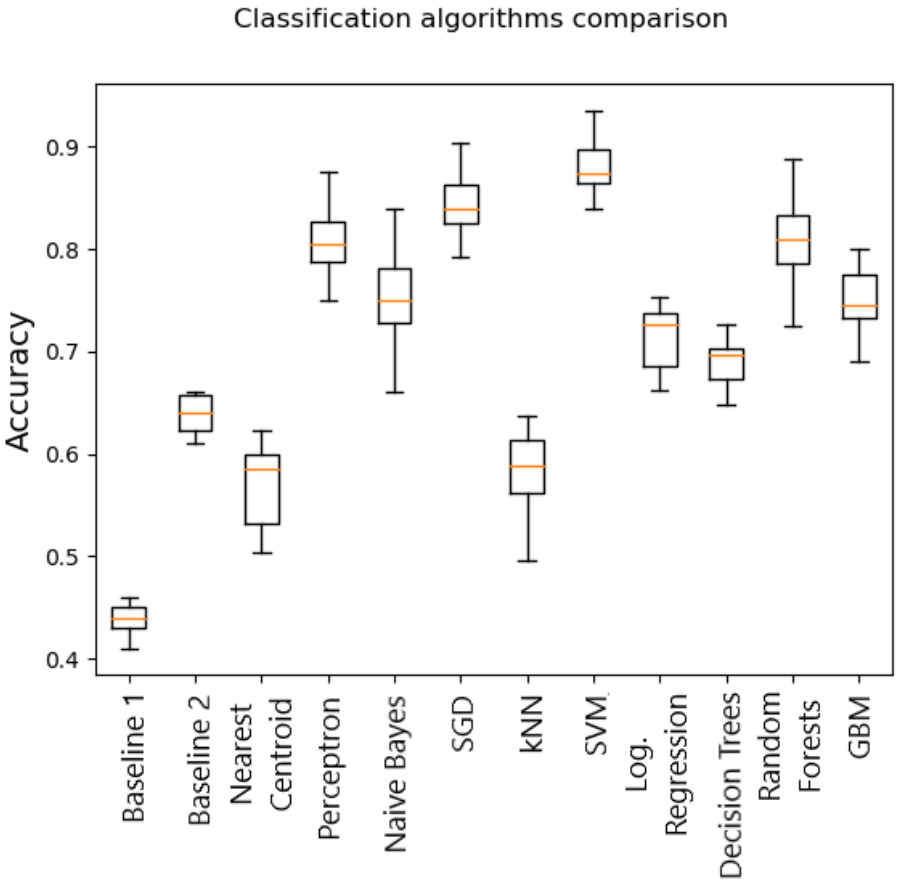

FIGURE 5.2: Comparison between the used machine learning algorithms (see Table 4.6) based on 10 fold cross validation.

From Figure 5.2 we can observe that the different classifiers have accuracy scores between 0.55 and 0.90, on average. The neighbors-based (kNN and Nearest Centroid) classifier models have the worst performance in comparison, with scores circulating around the 60% mark. This means that these types of models, while having better performance than the first baseline model, fail to overcome the second baseline model's performance. All other models have significantly better performance than both baseline models. For example, Decision Trees and Logistic Regression give performance with a median around the 70% accuracy which beat the baseline models significantly. The ensemble learning models (Random forests and GBM), Naive Bayes and Perceptron have on average approximately 10% better performance than the previous cluster of groups. The AdaBoost algorithm is omitted from the graph as it performs as an outlier model with drastically worse results (below 10% accuracy).

Ultimately, we can see that two classifiers have the best performance and reach a median score above 80% accuracy. These are the SGD and SVM (liblinear implementation) with respectively median scores of 84% and 87% accuracy. Since the results are based on experiments conducted on various test sets as the product of splitting the data in folds, the models tend to have some variance in performance, with the maximal performance gravitating around the 90% mark for both of these models. This will require some further exploration using the precision, recall and f1 metrics to determine which model is the best performing model on test sets that become increasingly difficult.

For this purpose, in accordance with the baseline models, we use different test

sets of 2, 5, 10, 15 and 20 potential authors. The SGD and the SVM classifier models were accordingly tested on each of these test sets and compared with each other. The results can be seen in Figure 5.3.

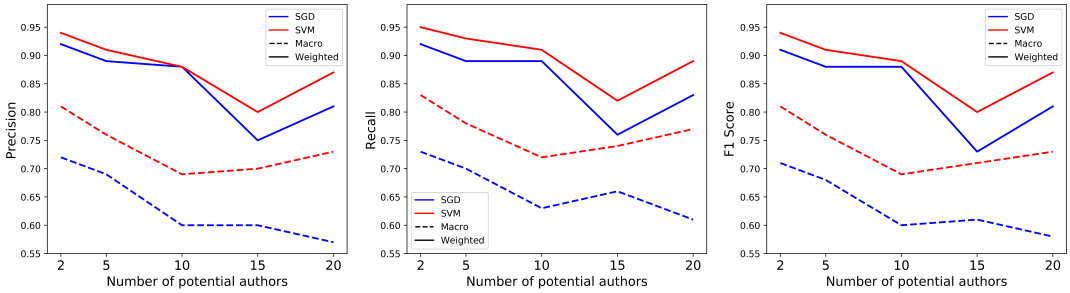

FIGURE 5.3: Comparison between the two best classifier models for precision, recall and F1 score for test sets that become increasingly difficult.

From Figure 5.3 it becomes apparent that the liblinear SVM implementation outperforms SGD on all test sets for all three metrics, for both weighted and macro performance. The only exception to this observation is at precision (weighted) for 10 potential authors, where the classifiers have equal precision. Both classifiers showcase similar patterns as performance in both cases peaks at 2 authors and gradually decreases afterwards until 10 potential authors, at which subsequently a relatively bigger decrease in performance can be found. When going from 15 to 20 potential authors the models perform better again. This could be caused if publications with 15 potential authors are less distinctive (either by chance or by being an intrinsic trait of publications with 15 potential authors) than publications with 20 potential authors. In addition, there are not many publications that can be linked to 20 potential authors which can also skew the results (see Figure 4.6). Considering all the results, we will thus use the SVM model for the goals and research questions of this thesis.

Unlike with the baseline models, precision and recall (and thus f1 score) are in balance here, which can indicate that models are inherently robust. The models outperform the baseline models significantly from 5 to 20 potential authors while reaching relatively consistent performance. Especially noteworthy is the considerable poor macro performance on the baseline models, while the classifier models consistently stay around the 0.6-0.8 range on all metrics for macro and above 0.75 for weighted performance. In addition, the classifier models do not fall to drastically low performance scores as with the baseline models.

The final step of the author classification task is tuning the SVM in order to find the best performing parameters. This is done by implementing a grid search that considers a set of values within probable ranges for different parameters and tests the model accordingly to find the parameter values that give the best performance. As a result, the following parameter values show to have the best performance:

- penalty: l2
- loss: hinge
- dual: True
- max_iter: 10000
- C: 100

- Tolerance: 0.01
- fit intercept: True
- intercept scaling: 10

The according performance difference of using standard parameter values versus tuned parameter values can be seen in Table 5.2. Hyperparameter tuning only leads to a marginal increase of approximately 1-1.3% performance depending on the metric. However with this tuned variation, we get the final model. This model achieves good results with respectively weighted and macro F1 scores of 0.92 and 0.76.

TABLE 5.2: Performance of the SVM classifier on the test set using standard parameter values and tuned parameter values (on all publications that can be linked to 5 to 20 potential authors).

	Standard			**Tuned**		
	Precision	Recall	F1	Precision	Recall	F1
Macro	0.75	0.77	0.75	0.76	0.78	0.76
Weighted	0.91	0.93	0.92	0.92	0.93	0.92

One of the main research questions in this thesis is whether adding metadata can alleviate the heterogeneity problem. Thus, different models with different feature selections were explored starting from a basis of only using the content of the publication. Ultimately, this showcases to what degree adding descriptive contextual publication metadata as well as adding the features that were engineered affects the model. The results of this experiment can be found in Table 5.3.

TABLE 5.3: Performance of the classifier for different feature inputs, with a focus on the addition of the contextual publication metadata (Chapter 3) and created features (Section 4.3.1).

	Weighted			**Macro**		
Features	Prec.	Recall	F1	Prec.	Recall	F1
Content + all metadata + created features + (ambiguous) familyname author	0.92	0.93	0.92	0.76	0.78	0.76
Content + all metadata + created features	0.76	0.77	0.76	0.54	0.55	0.54
Content + all metadata	0.73	0.75	0.73	0.52	0.53	0.51
Content + publisher + genres + themes + created features	0.72	0.74	0.72	0.52	0.52	0.50
Content + publisher	0.68	0.71	0.68	0.48	0.49	0.47
Content + genres + themes + created features	0.64	0.67	0.64	0.43	0.44	0.42
Content + genres + themes	0.63	0.66	0.63	0.43	0.43	0.41
Content + created features	0.61	0.64	0.61	0.41	0.41	0.41
Content only	0.59	0.61	0.58	0.38	0.39	0.37
Best performing baseline model	0.49	0.66	0.55	0.11	0.20	0.14

Table 5.3 shows that addition of different metadata features as well as created features incrementally improves performance. Ultimately, using all contextual information in combination with created features yields the highest scores. This can indicate that a multitude of different heterogeneous input features could, when selected, processed and combined accordingly, lead to a performance increase of approximately 27% for weighted performance and 41% for macro performance (not considering ambiguous author surnames as a feature). The specific influence of features will be further explored in Section 5.4, where they are also compared with author related metadata.

5.3 Experiment 2: Similarity learning

For determining the best model for similarity learning, we undergo the same k-fold cross-validation procedure with k set at 10. As these are regression models, we use the (negated) mean squared error (MSE) instead for determining the best model. The results can be found in Figure 5.4.

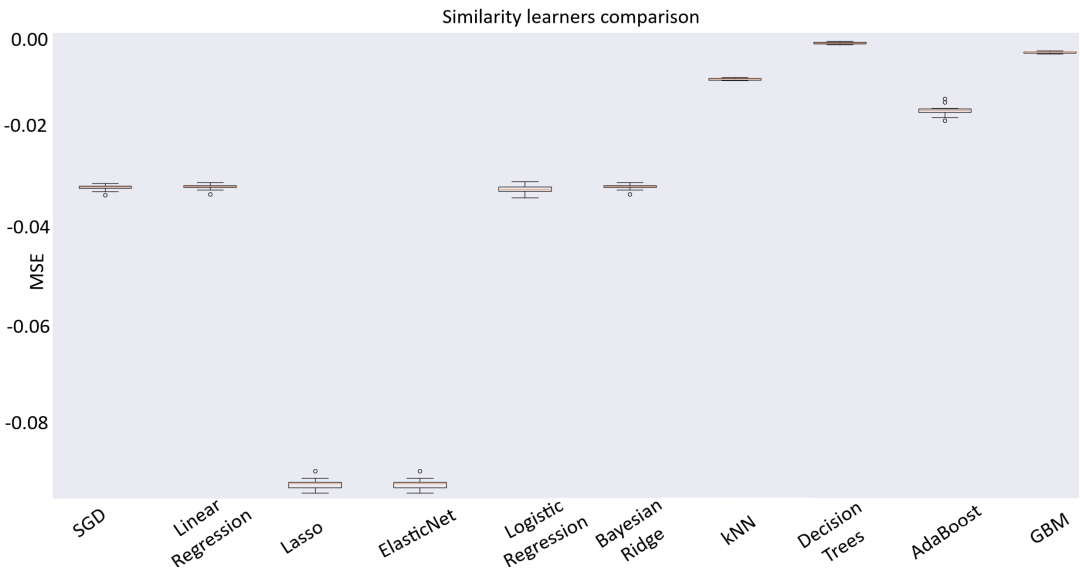

FIGURE 5.4: Comparing different types of similarity learners (see Table 4.6) based on MSE scores.

Figure 5.4 shows that there are clusters of groups to be found regarding performance. Lasso and ElasticNet have the worst performance with identical scores, followed by a group of models who approximately have a negated median MSE value of around -0.03 (linear regression, logistic regression, bayesian ridge and SGD). This could indicate that these types of models tend to fundamentally learn the same patterns with this data input. From the ensemble learning methods AdaBoost has the lesser performance, while also only having somewhat better performance than the previous mentioned cluster of machine learning algorithms. kNN performs relatively well, especially considering its classifier counterpart in Section 5.2. Ultimately the best performing models are GBM (ensemble learning) and Decision Trees (DT). Their scores are approximately similar, so they will be explored further. It becomes also apparent from observing the ranges of the boxplots in Figure 5.4 that the similarity learners do not have much difference in performance over different test sets. Unlike the classifiers, which relatively have much more variance in performance, this could be an indication that the similarity learners are more robust models.

GBM and DT will both be explored through converting the regression output to a ranking of different authors based on the predicted similarity scores. We will explore their precision, recall and F1 score for k=1 and seek which algorithm performs better for further use. The results for the test sets with different number of potential authors can be seen in Figure 5.5.

Figure 5.5 shows that for some metrics on some test sets performance only marginally differs, but overall GBM has the better performance on different test sets. These differences become more noticeable as the test set becomes increasingly difficult. We will thus use the GBM model for the goals and research questions of this thesis as the found superior performance applies for both weighted and macro performance. Noteworthy are also the patterns found in the graph. The models behave more consistent and stoic than the classifiers when using more difficult test sets. There is not much performance difference between 5 and 20 potential authors, for example. Conversely, across the board the similarity learners do have lower performance scores than the classifiers.

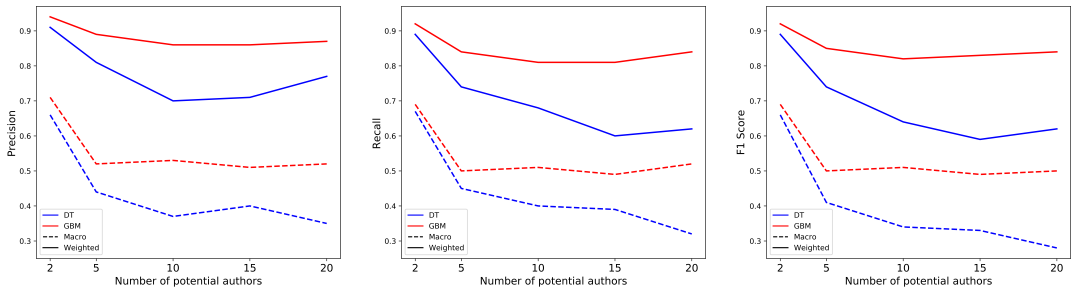

FIGURE 5.5: Comparison between the best similarity learners (GBM and DT) for precision, recall and F1 score for test sets that become increasingly difficult.

As with experiment 1, the similarity learner is also accordingly tuned using a grid search implementation. Consequentially, the best model performance is achieved with the following parameter values:

- number of estimators: 1000
- min samples split: 5
- min samples leaf: 4
- subsample: 0.7
- max depth: 13
- learning rate: 0.1

The results can be seen in Table 5.4. Tuning in this case has a significant effect on the performance of the model, with approximately a 14% increase for macro F1 performance. For weighted F1 performance, an increase of 3.7% performance is reached.

The similarity learner also makes it possible to use the (externally retrieved) author information. As, in line with Table 5.3 for Experiment 1 we will look at the role of different feature selections and their impact on the model. The results can be found in Table 5.5.

TABLE 5.4: Performance of the similarity learner on the test set using standard parameter values and tuned parameter values (on all publications that can be linked to 5 to 20 potential authors).

	Standard			**Tuned**		
	Precision	Recall	F1	Precision	Recall	F1
Macro	0.44	0.41	0.42	0.50	0.48	0.48
Weighted	0.86	0.79	0.81	0.87	0.82	0.84

TABLE 5.5: Performance of different feature inputs for the GBM similarity learner.

	Weighted			**Macro**		
Features	Prec.	Recall	F1	Prec.	Recall	F1
Title + publisher + year of publication	0.91	0.87	0.88	0.65	0.63	0.63
Content + all author information + metadata + created features	0.87	0.82	0.84	0.50	0.49	0.48
Content + metadata + created features	0.87	0.82	0.83	0.48	0.46	0.46
Content + all author information	0.81	0.77	0.77	0.44	0.43	0.42
Content	0.81	0.68	0.72	0.37	0.34	0.34
Autobiography + age + role author (All author information)	0.67	0.69	0.66	0.32	0.35	0.32
Role author	0.57	0.65	0.59	0.24	0.32	0.26
Autobiography + age author	0.61	0.46	0.49	0.22	0.23	0.21
Autobiography author	0.50	0.30	0.33	0.13	0.14	0.11
Best performing baseline model	0.49	0.66	0.55	0.11	0.20	0.14

Table 5.5 showcases the influence of adding author related information. While the impact of the individual author related data components differs, ultimately the combination of author related information only increases performance marginally. On average a 3% increase for macro performance is found while weighted performance only has a 1% increase for F1 score when compared to using all metadata. However, the best performance for the similarity learner is reached when using the top 3 most important features (which will be addressed in Section 5.4). Thus, best performance is reached when only using publication metadata, which might indicate that the addition of (the available/linked) author information is redundant.

To seek how the similarity learner performs when modelled as a ranking architecture, we also calculate recall values at higher k values. The results for higher k values can be seen in Figure 5.6. We can observe that recall increases with higher k values, most apparent when going from a top 1 to a top 3 ranking. After k = 3 recall still increases but at a significantly slower rate. This could mean that providing the results in a ranked matter can be useful, especially when considering a top 3 ranking.

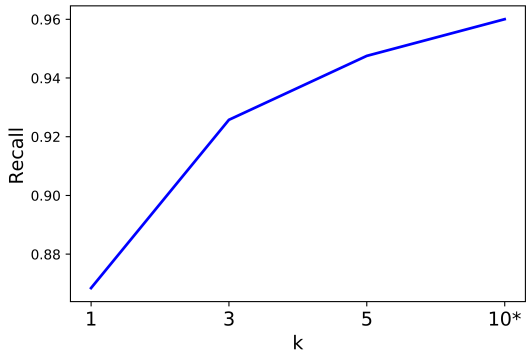

FIGURE 5.6: Performance for recall@k set for different values. Recall@k = 10 is only calculated for rankings with at least 10 potential authors.

5.4 Feature importance and relationships with machine learning

How models arrive at their predictions is the result of the found learning patterns in the data inputs. In this section we take a look at the features and their relationship with the target variable (the correct author), as well as their importance.

We compute the Pearson correlation coefficient for determining the correlations of features and the target variable. This has been computed for the top 15 features with the highest correlations and can be observed in Figure 5.7.

Figure 5.7 showcases that a few features stand out in terms of correlation with the correct author. This can be observed when inspecting the first row or column. Thus, namely the publisher, year of publication, genres and title features seem to stand out. They have a relatively higher correlation coefficient, up to the considerable high correlation coefficient of 0.70 for the publisher feature. A majority of the features have a correlation of around the 0.33-0.36 mark, mostly consisting out of the created features that convey statistical information. We can also see that the concatenated content features (both TFIDF and W2V) have a lower correlation with the target variable when compared to solely using the title of the book. However, when observing all correlation coefficients, we can see that title has a high correlation with the content feature (0.76). This could possibly indicate that using both is redundant and precedence should be given to using the title solely. Other features do not seem to imply collinearity, as most of the matrix consists out of low correlation coefficients. Furthermore, features not included in Figure 5.7 failed to have a correlation score of 0.25 or above with the target variable.

To determine the importance of different features for accurately predicting the right author, we compute the relative feature and permutation importance. The relative feature importance indicates the importance of a particular feature for the model when compared to all other features and how much it influences the model's predictions as such. The feature importance can be computed by considering the underlying architecture of the trained and fitted machine learning models which provide insight regarding the weighted coefficients the model uses to make its predictions. Furthermore, the permutation importance for a feature is an indicator of how much the model relies on the feature for making an accurate prediction. This is done by shuffling the values of a feature and then subsequently searching how much this increases the prediction error. If after doing so, prediction error increases consecutively for different publication instances, then the permutation importance

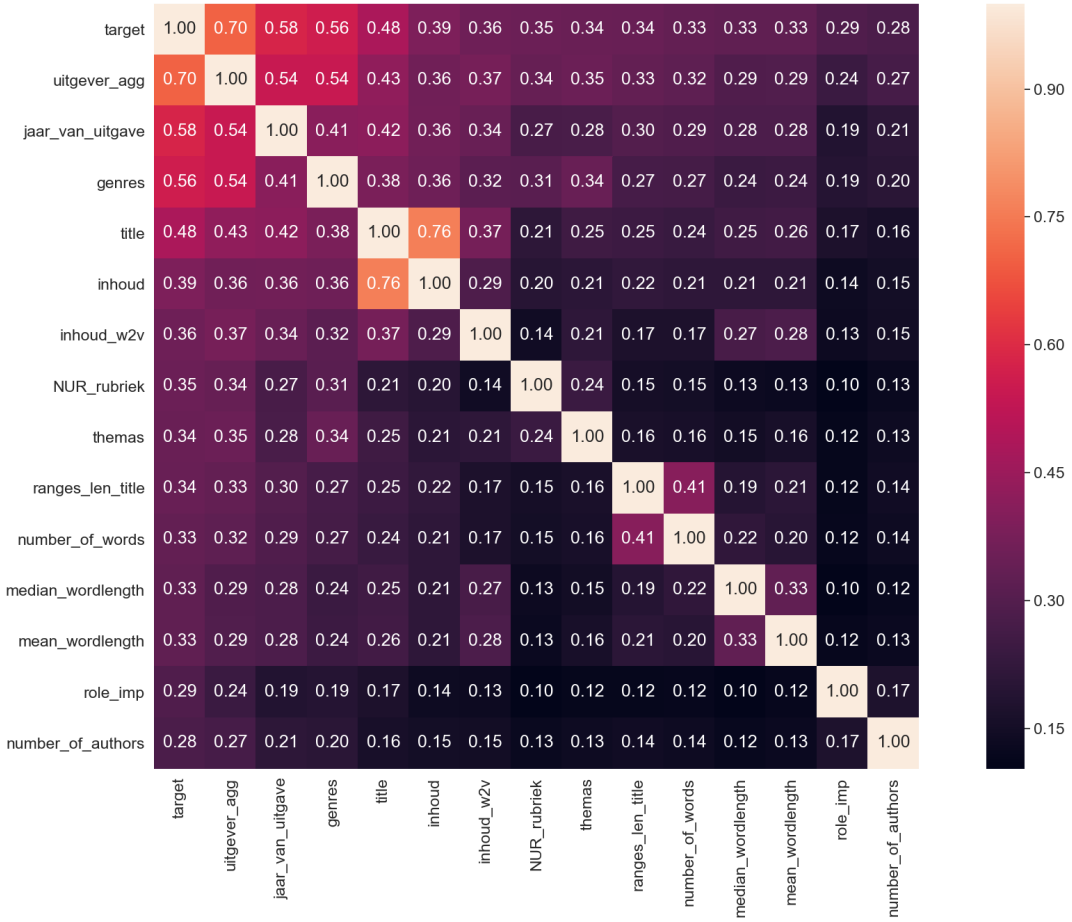

FIGURE 5.7: Correlations between input features and target variable (correct author) for the top 15 features with the highest correlations.

value will reach higher values and thus indicate an important feature. The results of these importance metrics for the different features can be observed in Figure 5.8.

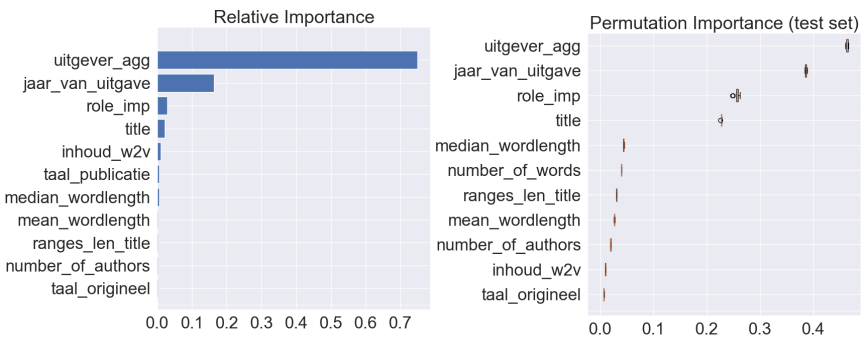

FIGURE 5.8: Relative feature and permutation importance.

Figure 5.8 showcases that some of the higher correlated features also have the highest relative and/or permutation importance (publisher, year of publication and title). The publisher feature is significantly dominant regarding relative importance, which diminishes the relative importance of other features. A surprise found in the top 5 features is the role of the author feature, which despite its lower correlation score in Figure 5.8 has the third highest relative and permutation importance. Furthermore, the absence of the number 3 most correlated 'genres' feature is also

remarkable. When looking at permutation importance we can see that the current input does in fact affect the model for a variety of features. To get more insight in how the model uses other features in a different setting, we also seek feature importance with the omission of the dominant features. The results can be seen in Figure 5.9.

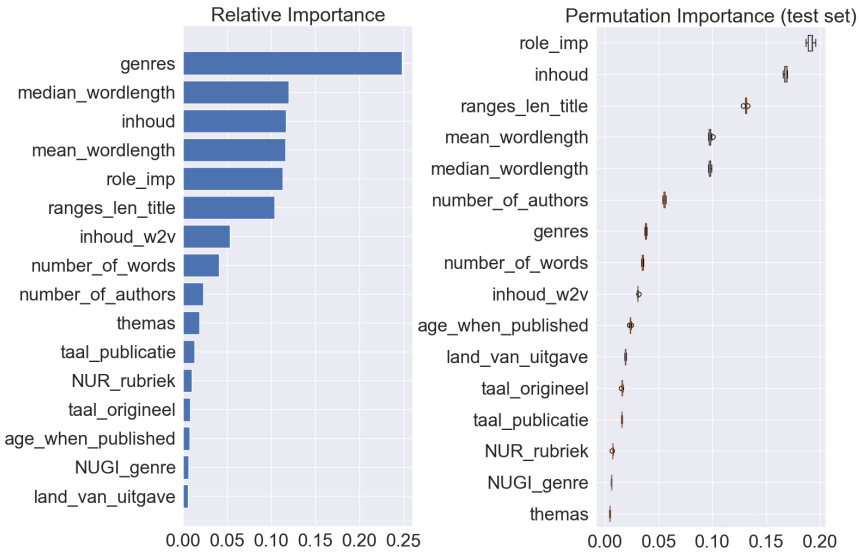

FIGURE 5.9: Relative feature and permutation importance without title, publisher and year of publication.

When omitting the most dominant features we get a more balanced importance distribution of the remaining features. Genres, while not scoring well with inclusion of the dominant features, now has the biggest relative importance. At the same time there is not a considerable discrepancy in importance with the other features. Following genres, a selection of features have approximately the same value in terms of importance, such as the statistical features, content and the role of the author feature. Role has degraded, however, in terms of relative performance while simultaneously having the highest permutation importance. Less important features now also seem to play a role in the model and do not get completely diminished from having a role in the model's predictive decisions.

5.5 Human perception of feature importance for authorship attribution

Regarding how the features are perceived by different types of human experts, we can see the results of the conducted survey (as described in Section 4.1.3) in Figure 5.10. Figure 5.10 shows that experts mainly attribute importance to author information related features such as age of the author, autobiographical notes about the author and role of the author. These three features all get 10 votes. Other relatively high scoring features is the year of publication feature with 7 votes and the title with 5 votes. In contrast, most of the contextual publication metadata get around 1 to 3 votes and are not deemed as important. 'Others' in the pie chart are the publication language and country of publication features which both only got one vote. Three features failed to get any votes (and are thus not displayed in the pie chart), which are the (1) number of authors associated with the publication, (2) the engineered statistical features and (3) extra annotations about the publication.

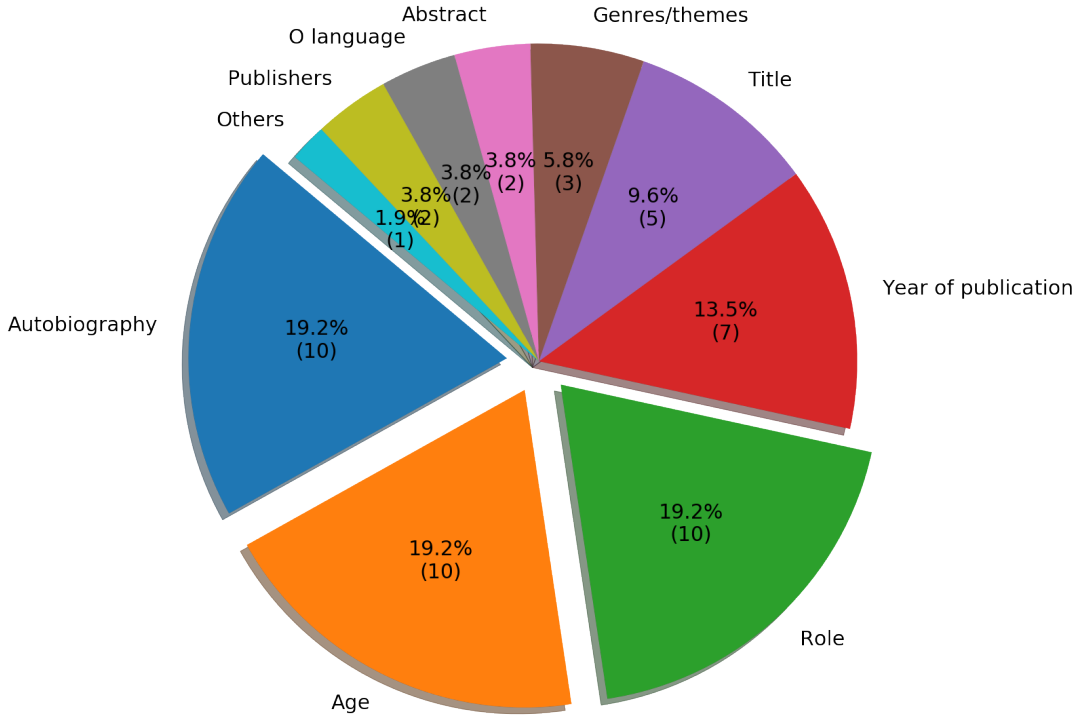

FIGURE 5.10: Perceived importance of the features as indicated by the respondents (N=18) of the survey.

When comparing the results of the survey we can see that there is a significant discrepancy between machine learning attributed importance and human attributed importance. Human experts primarily attribute importance to author information, while the machine learning model primarily attributes importance to publication metadata. Furthermore, it is remarkable that the autobiography feature that is highly deemed important by human experts does not even have a spot in the top 15 most important features (with or without omission of the dominant features) of the machine learning model. Similarly, the age feature also ranks near the bottom of the most important features (and only after omission of dominant features). A middle ground can be found in the role feature, which is both deemed important by human as well as machine learning model. Further discrepancies can be found when considering the machine learning model's perspective, which marks the publisher feature as the most important feature. Nonetheless, the publisher feature only gets 2 votes and therefore is ranked as one of the least important features by human experts. The machine learning model also deems the year of publication and title features as some of the most important features. While year of publication and title are not as highly rated by the human experts as other features, they still do get a sufficient number of votes (especially year of publication).

5.6 Experiment 3: Comparing textual representations with deep learning

The actual text of a publication is paramount to the publication's core characteristics. Where previous experiments focus more on different methodologies for learning, dealing with heterogeneous metadata and added atypical data, experiment 3 focus

purely on text. As previously indicated, available text of the publication is sparse and in approximately 75% of the cases there is only a title available. In this section we take a look at which of the discussed text representation types ultimately give the best predictive performance.

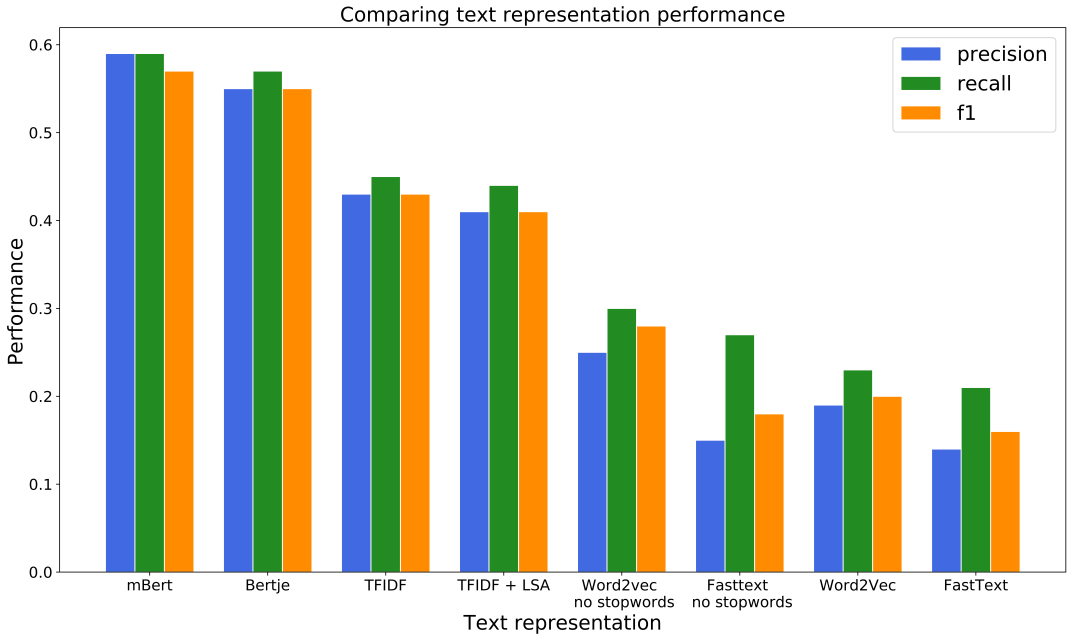

FIGURE 5.11: Performance for different types of textual representations for machine learning.

The results of the compared text representations can be seen in Figure 5.11. We can see that ultimately both variations of BERT have the best performance. There is not much difference between mBert (multilingual model) and Bertje (monolingual model), although, mBert performs slightly better with an approximately 2% increase in performance. Both BERT implementations reach around a F1 score of 0.57-0.60 with balanced precision and recall scores. With approximately a lower predictive performance of 15%, we find the TFIDF models place second after the BERT models. TFIDF with LSA applied does not lead to a significant (positive) change in performance, conversely it leads to slightly worse performance. This could be the result of the corpus not having sufficient variance for mapping words into a lower dimensional representation, and in fact important information is lost when doing so. Taking the last places are all the variations of the static word embeddings based methodologies: Word2Vec and FastText. The variations with stopword removal have a 15% worse performance than TFIDF. However, it shows that exclusion of stopwords leads in both cases to performance increase. To seek if performance increases with change of the vocabulary of the word embeddings, multiple other corpora (such as Wikipedia, Sonar500 [1]) were tried but ultimately Word2Vec reaches best performance with the COW corpus as discussed in Section 4.5.1 (of which the results have been included in the graph). When looking at the differences in metrics, we can see that the lower the scores become the more discrepancy we can find between precision and recall. Overall, recall seems to always be (slightly) higher than precision, except for mBert.

Outside of performance, there might also be the consideration of using computational resources. All models were trained with a maximum of 20 epochs and a

[1] https://github.com/clips/dutchembeddings

batch size of 16. Some models converted earlier to optimal performance than others, with early stopping approximately detecting conversion to optimal performance at around 15 epochs on average. While BERT has the best performance it is also the most computationally expensive. BERT takes the longest time to train, and also is expensive in memory usage, even when input is confined to solely the titles of the publications. In comparison, while TFIDF and the word embeddings based models have significantly lower performance they also take significantly less time to train, validate and test.

Chapter 6

Discussion

In this thesis an extensible pipeline is created for dealing with heterogeneous textual and non-textual metadata. This involves various different defining pre-processing steps, feature engineering and selection, integrating linked data, converting data to different spaces and considering different text representations. Consequentially, the best performing machine learning models on the processed input data are obtained through comparison and hyperparameter tuning. Outside of the technical aspect of this research, we also conduct human factors-based research directly tied with bibliographical experts. Metadata specialists are interviewed for gaining insights in current problems of manually attributing authors to publications, as well as the quality of the available data. These insights consequentially answer whether machine learning implementations can facilitate solutions for the named problems. From these insights we find that machine learning can be used for this purpose. This is mainly due to the considerable amount of reference data, available data being labelled (i.e. can be used for supervised learning) and the task having a high automation potential, which machine learning has historically been successfully implemented for. However, it also becomes apparent that we need to consider and adapt to the heterogeneous character of the available data as well as the class imbalance problem for a successful implementation.

These considerations culminated into three different experiments, as well as a follow up to the interviews. Namely (1) an experiment where the processed data is used for an author classification task, (2) an experiment where we implement similarity learning and seek whether it increases performance when we consider the prototypical problematic nature with author classification and (3) an experiment where we compare different text representations to seek if BERT also achieves its state of the art performance on sparse/heterogeneous textual data and thus should be recommended for usage. Furthermore, a survey was taken accordingly to seek whether the outcomes of the machine learning models are in line with the expectations of human experts.

In this chapter we will discuss and evaluate the results within the different experiments as well as what the results mean in the context of this thesis' research questions. As the two resulting models of the different learning spaces (experiment 1 and 2) ultimately have the same goal, it is insightful to discuss the results of the two experiments in conjunction. In addition, we also evaluate the methodologies and seek their respective advantages and limitations when compared to each other.

6.1 Machine learning algorithmic performance

How can we obtain the most optimal performance when considering sparse and heterogeneous data input for (1) author classification and (2) similarity learning machine learning tasks?

Experiment 1 shows that the SVM has the best performance when it comes to making predictions for a combination of text and metadata features. In experiment 2, GBM, an ensemble learning implementation has the best performance.

Ensemble learning models include redistribution of weights to samples in subsequent learning tasks. These samples are the 'hard-to-predict' samples that the models have difficulties with predicting correctly. This fundamental notion tends to lead to more intricate models and thus can generally lead to better performance. In addition, ensemble learning models are a multitude of many (weak) learners, which through combination lead to more robust models as they are instantiated with many different perspectives for learning relationships between predictors and target variable. Theoretically, this should lead to more substantiated models and thus better predictive performance. In the case of the classifier this might not be the case due to the many features created as the byproduct of the pre-processing steps in the pipeline. One hot encoding, count vectorization, TFIDF and so forth create many vectors that are in turn newly created features. The ensemble learning models can become too resource intensive when there are a significant number of features to learn upon, so the number of boosting stages or number of estimators has to be limited in the classification task. This could be seen when performing cross validation, while the singular linear models took a few hours at most, the ensemble learning models could take days even in the context of using a lower number of estimators (such as 100). As the data input in the context of the classification is not well handled by ensemble learning models, this means that the lower number of estimators is mandatory. In turn the lower number of estimators will make the model confined to a fewer number of possible stages for fitting the model and learning the patterns in the data, which will conclusively lead to lower performance scores.

A linear model that consists out of a singular model can produce better results in such cases as classifiers such as SGD and SVM can work well with many data points without being too resource intensive. In addition, the data can be well characterized as prototypical separable data due to the many categories the data can be divided into (and thus the resulting to-be-predicted authors). This is probably why we also see that, for example, an algorithm that works on the basis of dividing labels iteratively into smaller subgroups based on categorical characteristics such as DT performs reasonably well, while neighbor-based algorithms such as kNN and NCC algorithms perform relatively worse. This is also based on the notion that kNN and NCC typically work better with continuous values to determine distance between instances, while the majority of the metadata can be represented by boolean values. This might also be the reason why kNN performs significantly better in the context of similarity learning (experiment 2), where all features have continuous numerical values as input. The remaining algorithms can be all classified as eager learners, which can mean that differences in performance might be attributed to lack of optimization. Under different optimization procedures a bayesian classifier might outperform a support vector machine, for example. However, in the context of this thesis' research goals, the machine learning algorithm that grants the best performance is ultimately trivial as the goal is to find an algorithm that gives the best performance for answering the research questions, and not necessarily to investigate why a certain algorithm performs better than an other algorithm.

Since the similarity space in the context of experiment 2 creates a number of features equal to the number of different (raw) metadata features, this leads to a dataset that does not have a considerable number of features. The number of estimators can then be adequately set to a number that gives optimal performance, but does not overfit the model. In this case we see the true potential of ensemble

learning, namely that GBM significantly outperforms all other forms of regression learning using regression metrics, bar DT. However, GBM also considerably outperforms DT when converting the output data to a ranking (the desired output format) and getting a recall, precision and f1@k=1 score similar to classification tasks. Within ensemble learning algorithms, GBM outperforms the other included ensemble learning algorithm (AdaBoost). This is most likely the result of its more flexible nature and algorithmic difference of including approximation of gradients. Approximating gradients leads to the intricacy that GBM can find approximate solutions to the additive modeling problems more adaptively (and thus is not confined to a particular loss function such as with AdaBoost, which also only identifies shortcomings by adding high weights to problematic data points).

Ultimately, based on these results we would recommend using GBM when using a similarity learner and the liblinear implementation of a SVM when using an author classifier. Whether to use a similarity learner or an author classifier depends on other aspects as well and is also a matter of user preference. This will be discussed in Section 6.4, as investigating these methods' adaptation to the prototypical class imbalance problems with authorship attribution is another main research question.

In a general sense, machine learning performance between different algorithms should be judged on a case to case basis. As the no free lunch theorem applies it is usually recommended to test for a variety of algorithms, as it is challenging to determine beforehand what algorithm will work the best on the input data (Wolpert and Macready, 1997). Therefore the results in this thesis might not apply to other problems that involve a combination of textual input and associated metadata.

6.2 Addition of contextual publication metadata and author information

Does the addition of (1) contextual author information or (2) contextual publication metadata to heterogeneous data increase prediction accuracy and robustness, and how do these types of metadata compare to each other when used as input for a machine learning model?

While the author classifier can not use author information, the similarity learner can do so. This leads to one of the main research questions: does the inclusion of author information in fact add substance for making predictions? In a similar line, the question arises whether adding contextual publication metadata (which both types of learning methodologies support) in itself does change the model's behavior and predictive performance. As showcased in Section 2, many articles within the literature with similar goals only use the actual text of publications. Using additional metadata describing the publication for machine learning purposes is an under researched topic within the bibliographical domain, and is only sparsely researched across all domains.

6.2.1 Contextual publication metadata

When considering the contextual publication metadata, performance between individual features differs. The results as provided in Section 5.2 and 5.4 illustrate this, some features are dominant and actively increase predictive performance while others do not impact the model as much. The fact that the addition of (some) publication metadata does lead to considerable performance increase, means that we ultimately

can say that publication metadata does in fact increase predictive performance significantly. Performing according feature selection ultimately establishes which contextual publication metadata inclusion (or exclusion) leads to the most performance increase.

The differences between these individual metadata features can mainly be attributed to the quality of the information included in features. The best performing publication metadata features are publisher and year of publication. Both of these features are much more distinctive than other features, while also being some of the features with the lowest missing value rates. Processing the publisher feature is fundamental in this, as clustering of values that encapsulate the same publisher will mean that the model can correctly differentiate. If this is not done then the models will erroneously see the same publisher as completely different publishers based on variations found in the way they have been written down in the database, which drastically lowers the model's ability to generalize. Based on the notion that these processed features can successfully showcase differences in characteristics of publications (and thus their respective authors), then naturally these types of features will produce better results for a machine learning model.

Features such as publication language, original language and country of publication only have a marginal role in the model's predictions. This is probably due to lack of diversity found in their data as showcased in Chapter 3, which means that on this basis these features will fail to differentiate between different authors. Other features will have also have a marginal role due to the big rates of missing values, such as NUGI genres and NUR rubric. Accordingly imputing missing values leads to the same problem as with the language-oriented features, namely lack of diversity and characteristics to base differentiation on. These types of features thus, ultimately, are not well-suited for a machine learning task and should be omitted. This also reinforces the belief that contextual metadata should not always be automatically added under the belief system that using more information will lead to better results. Contextual metadata must be carefully examined regarding whether the data introduces valuable differentiation possibilities for a machine learning model, as not doing so can instead even decrease the performance of a model.

Surprisingly, the CBK genres and themes features can be seen as underperforming features. The fact that these features combined have a low rate of missing values, as well as a valuable ability to differentiate publications and authors based on their content should theoretically lead to a feature that can perform well in machine learning. The genres feature also has a high correlation with the correct author target variable, which should only reinforce this belief. However, performance is poor and when including all features they fail to be in top 10 regarding either relative or permutation importance. When looking at an explanation for this behaviour, we can perhaps deduct that this is due to inclusion of otherwise dominant features such as publisher, year of publication and title of the book. After omitting these features, the genres feature rises to become the feature with the most relative importance (see Figure 5.9). This can mean that there is a considerable correlation between genres and the three dominant features combined. Another possibility would be that there exists a correlation to a certain degree and in other cases the other features are preferred for basing predictions upon by the model, due to genres either being contradictory or not having enough predictive power as a feature for those cases. Either possibilities would make the usage of genres redundant and thus the model omits it from further use when considering all features.

6.2.2 Author information

The results of author information inclusion seem to indicate that author information in its current form is not useful. Both the autobiography as well as age of the author features perform poorly and have a low correlation with the correct author, as well as a low relative and permutation importance. While autobiography seems to be a promising feature (in theory and as mentioned by the experts in the interviews), its performance in real time can be attributed to a variety of things. Inspecting the autobiography data entries, it becomes apparent that next to a vast number of entry points having missing values many of the values are of low quality. Another significant portion of the feature includes codes that upon further consultation with different experts seem to not convey any important information, nor can they be traced back to words that convey useful information. Other data entries are either (1) one-word or small descriptions, (2) vague descriptions, (3) references to external websites, (4) references to a specific event in the author's life, (5) administrative notes (thus no actual autobiographical information) or (6) throwaway information. Ultimately, when evaluating this feature in depth we can conclude that the quality of the information is low. Which means that it is not possible in many cases to find semantic similarity between these biographical notes and the actual publications. Even in the cases of high quality information being inserted in this feature, there is also still the possibility that it is irrelevant to the publication's content, which will in turn also lead to no semantic similarity being found. The combination of all previously mentioned problems that arise with this feature, leads to a feature that will not perform well when used as a feature to base predictions upon in a machine learning.

Regarding other author related information, the age of the author (at publication) feature can in many occasions not be calculated due to mostly missing birth year values. Therefore in many cases it can not have a role in determining the correct author, which in turn explains it does not score well with the metrics that determine feature importance. With two out of three features considered not important, this only leaves the role of the author feature. Relatively, the role of the author scores high in the importance measuring metrics, even when it has a low correlation with the correct author. This might indicate that this feature mostly plays a role in lowering the number of potential authors, and not necessary at finding the right author. However, ultimately the model gets the best performance when using a feature selection that omits the role feature, as could be seen in Section 5.3. This might indicate that role only has a useful meaning in the context of a unfiltered feature selection input for the model, which is ultimately a model that gives lower performance overall. The unprocessed version of the role version has many missing values, and with domain knowledge approximately half of the missing values could appropriately be imputed. Whilst this significantly increases the quality of the role feature, this still leaves a relatively high number of missing values (approximately one third), which could only be imputed with a generic 'unknown' label. This resembles some of the earlier found problems with other features, namely a feature that can be hard to differentiate authors upon in a significant portion of the cases.

6.3 Differences between expert and machine regarding usage of information for authorship attribution to publication

Do bibliographical metadata experts differ in work approach and their perceived importance of information for authorship attribution in comparison with machine learning models?

In this thesis the role of the current manual attribution of authors to publications has been considered, mainly to get insights in current problems in manual attribution but also to get insights in how different types of information can potentially be used. Ultimately the goal of the KB is not to fully replace manual authorship attribution with automation, but rather to supplement it and make the attribution task significantly less laborious. From the interviews and survey results we can observe that there is difference between expert perceived importance of the available information and the importance of the features as determined by the machine learning models. In Section 5.5 this difference becomes mostly apparent, which could mean that implementation in the setting with the same people can lead to negative experiences. This might be the product of having to work with a supervising machine learning model that is conflicting to the human expert approach.

Experts attribute the most importance to the author information related features, while the machine learning models deem the contextual publication metadata features to be the most important. This could possibly be the result of the machine learning model's stoic and statistically determined importance of features, while humans have a perception of what features should be more important based on logical reasons. The main difference here could be that expectation differs from reality, as the author information related features tend to have low quality input as discussed in Section 6.2. If the author information related features were of high quality then this could potentially fulfill the expectations of the respondents. At the same time, the results can also indicate that humans can interpret data more effectively on a case-to-case basis, and thus understand when data is useful and when it is not. Furthermore, humans might have implicit knowledge about certain publications or authors that the machine learning model lacks (for example by not being directly available in the data that is trained upon). Another explanation could be that humans, such as the cataloguers, tend to work with more data than just the children's books database this thesis is focused on. It could be that in other datasets, features such as age and biographical notes about authors are richer and of higher quality, which then naturally will mean that they have more use in attributing authorship to publication.

Either way, the conflicting perceptions of feature importance lead to a new fundamental question regarding how the two approaches can be combined in a real-life setting. This could be solved, for instance, by combining the results of machine learning with what the respondents deem to be important information, and, not replace one or another. A visual indicator can in such cases showcase the confidence of machine attributed importance on a case-to-case basis, while simultaneously presenting the information the end users deem to be important. Through this way users can understand and consider the machine's attributed importance for possible new insights and contemplate their own expertise in conjunction. For some features it is also possible to mitigate the importance, as for example the age of the author implicitly can convey the same information as the year of publication feature. Both features convey certain information about the possibility that a certain author wrote a publication based on the timeline of an author's prior publications. However, the year of

publication feature is much more important for the machine learning models, while also being a feature with few missing values. In this case the year of publication feature can replace the age feature without any information loss and close the gap between human and the machine learning model's usage of information.

6.4 Methodology comparison: similarity learner versus author classifier

To what degree can the complexities of the prototypical authorship attribution class imbalance problem be alleviated by converting publication and author data to the similarity space?

Ultimately the question remains whether similarity learning is a better option for attributing authorship to publications, due to the previously discussed prototypical problems associated with author classification. One of the main benefits similarity learning provides is the integration of author information, however, as discussed in the previous sections this ultimately fails to increase performance in reality. A summary of the final performances of the different methodologies can be seen in Table 6.1. The results show that the author classifier has the best raw performance in terms of F1, recall and precision scores. Looking further however, we can see that similarity learning does provide some other benefits, namely (1) more robust predictions (2) possibility to convert predictions to a ranking architecture and (3) increased explainibility.

TABLE 6.1: Summary of the final performances of the best performing models from the differing implemented methodologies.

	Weighted			**Macro**		
Best performing models	Prec.	Recall	F1	Prec.	Recall	F1
Author classifier (Section 5.2)	0.92	0.93	0.92	0.76	0.78	0.76
Similarity learner (Section 5.3)	0.91	0.87	0.88	0.65	0.63	0.63
Baseline model (Section 5.1)	0.49	0.66	0.55	0.11	0.20	0.14

The similarity learner has more robust predictions as we can observe in the variety in its performance on different test sets, compared to author classification. Essentially this means that the number of ambiguous authors matter less when using a similarity learner, which can be beneficial in increasingly more difficult cases for authorship attribution. This behaviour can be most likely attributed to the fact that basing predictions in the similarity space is less dependent on a specific publication-author combination instance, but rather on the calculated cosine similarity values. A machine learning model learns the relationship by seeking whether the similarity values for each feature are high enough to consider the author in the comparison to be the correct author (or predict a high similarity score in case it is an incorrect author). This is fundamentally different from the author classifier where the combinations of many different types of numerical, categorical and textual features make predictions much more dependent on the type of publication as well as the number of potential authors associated with the publication. The latter space models a more difficult and intricate relationship between predictors and the target variable, while the similarity space models a relatively easy relationship.

In addition, the similarity space provides the possibility to model the predictions in a ranking fashion. This is in contrast with the author classifier, which provides a singular prediction (as is typical with machine learning classification). Providing the

predictions in a ranking matter is beneficial for a production setting, as more options are presented to the user. This is useful in the case of harder-to-predict instances, namely when multiple authors can be the correct author as the result of minimal differences being present in the calculated similarity scores. As can be observed in the results, recall actively increases when including multiple options (such as a top 3 or top 10 ranking) for presentation to an user.

Finally, the similarity learner provides explainibility that the author classifier lacks. In the case of the author classifier the underlying architecture for making predictions can be a black box in many instances, as there are many features that are specifically made for a machine learning model. This leads to a reduced explainibility and makes it difficult for users to understand how the model arrives at its predictions. In the similarity learner, this problem does not exist. As the general notion holds: the higher the score, the more probable it is that an author wrote a certain publication. The similarity learner provides similarity scores for every feature, meaning users have insight in how similar authors' past works are compared to the current publication for all individual features. This is even further simplified as in the case of the similarity learner these features are identical to their raw metadata presentation. The latter notion will mean that people without a machine learning background, such as the actual users that currently manually attribute authorship to publication, can also understand the output.

Where the author classifier has better raw performance this could possibly not uphold in other situations. In turn the similarity learner might adapt better to those situations considering its robust nature. The variance found in the author classifier's predictions (and in general a bigger decline in performance when increasing the number of potential authors) might indicate that for publications with a larger number of potential authors than 20 the similarity learner will start to outperform the author classifier. The biggest boost in performance in the author classifier comes from the addition of the name of the author, of which the machine learning model will recognize that this is a fundamental differentiating feature and technically reduce the multi class classification task with thousands of labels to many sub-classification tasks with 5 to 20 labels. Thus, this inclusion of family name is defining and integral to combat the main problem found with author classifications tasks (too many labels) as defined in the literature. Ultimately, this means that modelling an author classifier in such a way that reduces the many labels in subgroups of small label selections can be an alternative solution to the problems with standard author classification and negate the necessity of converting the data to the similarity space.

In the comparison between machine learning and baseline models, both methods significantly outperform the best performing baseline model. The best performing baseline model is the model that picks the author with the most publications in a list of potential authors. Purely from a statistical point of view this can lead to picking the correct author in a significant number of cases, which is why weighted performance is relatively high while macro performance is considerably lower. The latter observation is the logical byproduct of picking an author a priori, based on solely the notion of what statistically would lead to the highest rates of picking the correct author. However, the baseline model quickly degrades as we can see for its performance for a bigger number of potential authors and shows that the lack of a more in depth reasoning for predicting or picking an author leads overall to insufficient results. Machine learning does however facilitate this more in depth reasoning and showcases this in terms of significant increases in performance (up to 62% for macro performance and up to 37% for weighted performance).

With these significant performance increases in mind, there is also the consideration left regarding how either similarity learner or author classifier adapt to the discrepancies found in instances per class, as part of the class imbalance problem. It becomes apparent that the author classifier directly suffers from the class imbalance problem when inspecting erroneous predictions. Authors with fewer number of publications are the biggest group of classes that get wrongly predicted. We also see a bias towards authors with more publications in the individual predictions, as in case of uncertainty the model tends to predict the author with the most publications. Regarding the similarity learner, this class imbalance in this form is mitigated. As all instances in the similarity space are not instantiated by labels, but rather by target variables in the form of scores (regression output). This removes the discrepancies found in instances per class, as with classification. However, the similarity learner can also indirectly be affected by class imbalance. As authors with fewer publications tend to give less substantiated similarity scores, which in turn, also lead to faulty predictions. These problems could potentially be solved by implementing confidence scores. Confidence scores can offset authors with a few publications by giving the model information that the calculated scores are perhaps not of high quality for certain authors.

6.5 Text representations

How do different types of text representations for heterogeneous and sparse textual data affect model predictions?

In this section we discuss the results of experiment 3, that focuses on researching which text representation grants the best performance. From the results it becomes apparent that BERT significantly outperforms the alternative text representations and upholds its state of the art performance from the literature on sparse textual input.

Within BERT we test for a monolingual (Bertje) as well as a multilingual model (mBERT). While difference is minimal, the multilingual model grants higher performance with the same instantiated parameter values. If this is not attributable to randomness or noise, then this could be the result of the available publication text not being Dutch in all cases. This could also explain the small difference as only a small minority of the publications are in other languages (up to 8%), which potentially could be any language. mBert is constructed through internal representations that have shared syntaxis of different languages, which is primarily the product of learning natural syntactic categories that have cross-lingual overlap. This leads to an ability to generalize cross-lingually due to parts of information language being encoded in the same embedding space (Pires, Schlinger, and Garrette, 2019). In the context of this experiment this could explain that mBert can also learn non-Dutch publications (when set to Dutch detection) and accordingly correctly classify them, especially with other languages that share lexical similarity (such as Germanic or Romance languages) (Libovický, Rosa, and Fraser, 2019). On the contrary, Bertje is solely pre-trained on Dutch text and thus purely grants contextualized Dutch embeddings, which will lead to a smaller chance of correctly classifying non-Dutch publications. However, the latter notion should lead to more intricate embeddings for Dutch publications and thus potentially also produce better results (as previously discussed) for the 92% majority Dutch publications. We test for this assumption in a confined experiment, where we only use these Dutch publications. This confirms the assumption, as Bertje does in fact have a higher performance than mBert in this case,

although, performance only marginally differs. In another experiment, we try to observe the differences in performance when using the 8% non-Dutch publications as input. In this case mBert has a significant higher score, while Bertje grants drastically low performance scores. From the literature we previously could observe that monolingual models tend to perform better for language-specific tasks. However, the fact that it is not happening in this case could also be caused due to the Dutch language having one of the biggest Wikipedia corpus to train upon in the multilingual model, which in turn produces more enriched embeddings (Libovický, Rosa, and Fraser, 2019; Wu and Dredze, 2020). Thus, the more intricate embeddings would lead to better text classification results, in opposition to other languages included in the multilingual model.

From the literature we could get an indication that BERT would outperform both TFIDF and the word embeddings based models. Theoretically, this should be expected as BERT produces more intricate embeddings due to a deeper contextualization of words. For example, BERT produces multiple embeddings for the same word, which in turn maps different meanings of a word to the context they are used in (such as synonyms). In word embeddings the meaning is static, meaning all different usages of a word will get mapped into a singular embedding. Naturally, the former would lead to better results when trying to learn upon textual input. Semantic meaning is not learned in the case of TFIDF, which in itself is already a lack of a fundamental feature that would lead to worse results. TFIDF produces a statistic that measures topical importance, where the input corpus is formed out of the publications themselves. The other text representations variations learn from external corpora, usually significantly bigger and richer than the corpus of only the publication titles. The latter could lead to problems associated with out-of-vocabulary (OOV) words, but at the same can model relationships between meaning of words that are in the corpus more accurately. BERT outperforming TFIDF thus makes sense, as modelling actual semantic meaning has many uses that gets omitted by TFIDF.

However, TFIDF does outperform static word embeddings such as Word2Vec and FastText which in fact model semantic meaning. The corpus of publication text being used as input for TFIDF, means that TFIDF can model relationships well between words that are only used by certain authors. This will then lead to easier to predict authors in such cases, in contrast to word embeddings. Additionally, a considerable number of authors indulge in publishing a book series. Book series result into a set of publications having very similar names, in which TFIDF can establish efficiently that these belong to the same author. This is amplified if the author uses specific words attributable to that author (such as a character name, i.e. 'Nijntje') in their title.

Word2Vec in this thesis uses the COW corpus as previously mentioned, which leads to 9% of the records having OOV words when generating average word vectors for the publication text input. Due to the size of the COW corpus, we theorize that in these cases OOV words would most likely be either names, neologisms, archaic words or made-up words. TFIDF would thus in these cases have better results on these types of words which will lead to a better performance overall. Other explanations might not necessary be found in the intrinsic properties of the text representations themselves, but rather in the underlying text classification neural network architecture. As the text classification neural networks are instantiated with exactly the same layers and neurons for comparison purposes, it might be that TFIDF is better optimized with this type of architecture. Simultaneously, the neural networks for word embeddings might need to be optimized and use different parameter value, as well as the layering of the network to have better performances.

Between fastText and Word2Vec there is not much difference in the underlying architecture as both use the CBOW neural network model. The main difference is that fastText has a better way of dealing with OOV words as fastText represents words as n-grams of characters (parts of the words) which can lead to better generalizability. However, unexpectedly, in this experiment fastText leads to worse performance in both cases (with or without stopword removal). This might be the byproduct of the unique approach to dealing with OOV words, which paradoxically might reduce performance for some cases if OOV words are not reducible to more useful subparts of words. Other explanations might be the same as the difference found with TFIDF versus word embeddings; lack of individual model optimization. Since the difference is not considerable, we can also possibly think of randomness as possible cause.

6.6 Applicability beyond the KB

This thesis focused on a case study catered to the KB, as all data was facilitated by the KB. Albeit not directly a research question, we have to wonder to what degree the findings of this thesis are applicable to other institutions or domains that either work with (1) heterogeneous data or (2) have ambitions for automating authorship attribution. In this section we will discuss to what the degree the described techniques and findings are universally applicable. We showcased in Chapter 1 and 2 that data problems similar to the problems found in the KB's heterogeneous bibliographical database are common. In a general sense, other institutions or domains that work with heterogeneous data can use the techniques described in this thesis for improving the quality of their data, with ultimately the goal to increase predictive performance for a machine learning model. This should also be a fundamental step in the pipeline before using any machine learning model for such tasks. However, it should be judged on a database basis regarding what types of processing or engineering is needed to increase the quality of the data. This thesis can efficiently supplement this process, as it showcases the types of techniques that can be applied to different types of problems found within heterogeneous data (for example, the described different pre-processing steps for textual data versus categorical data). Furthermore, the addition of contextual information should also in general be considered by any institution working with heterogeneous data, as this thesis showcases the significant role it can have in increasing predictive performance.

The magnitude of applicability increases for specifically the task of authorship attribution with heterogeneous data, as the scope of the problem gets reduced. Institutions or research groups working with authors as labels, will have a multi-class problem with an atypical higher number of labels. Thus, the practice of converting data to the similarity space can be universally recommended in those cases, as this showcases to be an effective solution for this type of class imbalance problem. Furthermore, as this thesis focuses mostly on Dutch publications, we also have to consider whether the findings apply in an international sense. As the language is only relevant for the textual features of the publication, these findings stay applicable to all other types of (contextual) metadata. Even in the cases of textual features, language ultimately only plays a marginal role. This is due to the fact that both a multilingual as well as monolingual BERT model outperform other text representations significantly. mBert (the multilingual model) showcases to have the best performance in regards to the semantic representation of the text. Thus, institutions using non-Dutch publications could use the same model. As either variation

of BERT showcases a significant increase in performance over other types of text representations, other institutions could also still consider the possibility of using a monolingual BERT model. As the types of monolingual models are different per language, we can not give an indication whether mBert will also outperform non-Dutch monolingual models. As previously discussed, we could see that for some languages monolingual models outperform mBert, thus we recommend to explore this on a case-to-case basis.

Chapter 7

Conclusion

This thesis shows that there are possibilities for replacing or alleviating laborious manual attribution of authorship to publication with heterogeneous data input. For this purpose an extensible pipeline can be used that ultimately leads to adequate performance, as this pipeline results into a model that can predict the right author in a substantial majority of the cases and significantly reduce the number of possibilities in hard-to-predict cases. Either way, this implementation reduces costs and time consumption for authorship attribution in a real-world setting and thus facilitates more efficient work procedures. Before applying machine learning, various techniques such as pre-processing data, feature engineering and selection, converting data to other vector space representations and integrating linked data can increase the quality of the input data. Consequentially, different machine learning methodologies can be considered for authorship attribution: namely author classification and similarity learning. Author classification grants the best performance, while similarity learning grants more explainability, options and produces more robust predictions as part of dealing with the class imbalance problem. Regarding supplementing the text of the publication with information, the addition of contextual publication metadata shows to actively increase predictive performance. On the contrary, author related information, as made possible by using similarity learning, does not result into increases in performance due to low quality input. For representations of text, the multilingual variation of the NLP language model, BERT, gives the best performance. Multilingual BERT produces the best performance due to a combination of more intricate embeddings and a wider coverage of the semantic meaning in the available publication text versus other text representations.

Based on interviews and a survey this thesis also gives insight in manual authorship attribution and human factors-based considerations for integrating machine learning in the current workflow. Investigation of human experiences showcases that the end users have a different perception of the importance of the available information when compared to machine learning models. However, this difference in perception should not lead to negative experiences. Some data features that are perceived to be important can either be replaced by other, similar information conveying, features that lead to better performance. In addition, it could be illustrated to an end user that the data feature deemed important is of low quality for a certain publication (and that other features might be more important).

For future work we recommend to research whether results stay consistent for publications with more than 20 potential authors, as well as publications that are not children's books. In addition, we also recommend to research whether non-primary authors such as translators and editors can also be correctly predicted by implementing the pipeline described in this thesis. Finally, we recommend to research whether implementation of the machine learning model in a real life setting will indeed affect current work approaches and the associated end users positively.

Bibliography

Abbasi, Ahmed and Hsinchun Chen (2005). "Applying authorship analysis to extremist-group web forum messages". In: *IEEE Intelligent Systems* 20.5, pp. 67–75.

Abuhaiba, Ibrahim and Hassan Dawoud (Apr. 2017). "Combining Different Approaches to Improve Arabic Text Documents Classification". In: *International Journal of Intelligent Systems and Applications* 9, pp. 39–52. DOI: 10.5815/ijisa.2017.04.05.

Adhikari, Ashutosh et al. (2019). *DocBERT: BERT for Document Classification*.

Alloghani, Mohamed et al. (Jan. 2020). "A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science". In: pp. 3–21. ISBN: 978-3-030-22474-5. DOI: 10.1007/978-3-030-22475-2{_}1.

Arcia, Yaritza Adame et al. (2017). "Author Profiling, instance-based Similarity Classification". In: *CLEF*.

Beel, Joeran et al. (2016). "Research-paper recommender systems : a literature survey". In: *International Journal on Digital Libraries* 17.4, pp. 305–338. ISSN: 1432-5012. DOI: 10.1007/s00799-015-0156-0.

Bermingham, M L et al. (2015). "Application of high-dimensional feature selection: evaluation for genomic prediction in man". In: *Scientific Reports* 5.1, p. 10312. ISSN: 2045-2322. DOI: 10.1038/srep10312. URL: https://doi.org/10.1038/srep10312.

Bhaya, Wesam (Sept. 2017). "Review of Data Preprocessing Techniques in Data Mining". In: *Journal of Engineering and Applied Sciences* 12, pp. 4102–4107. DOI: 10.3923/jeasci.2017.4102.4107.

Blanco, Alberto et al. (May 2020). "Boosting ICD multi-label classification of health records with contextual embeddings and label-granularity." eng. In: *Computer methods and programs in biomedicine* 188, p. 105264. ISSN: 1872-7565 (Electronic). DOI: 10.1016/j.cmpb.2019.105264.

Boenninghoff, B et al. (2019). "Similarity Learning for Authorship Verification in Social Media". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2457–2461. ISBN: 2379-190X VO -. DOI: 10.1109/ICASSP.2019.8683405.

Boer, G. (2019). *Issues with the Dutch Snowball Stemmer*. URL: https://github.com/snowballstem/snowball/issues/1.

Breiman, Leo (2001). "Random Forests". In: *Machine Learning* 45.1, pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

Bühlmann, Peter (Jan. 2012). "Bagging, Boosting and Ensemble Methods". In: *Handbook of Computational Statistics*. DOI: 10.1007/978-3-642-21551-3{_}33.

Castro, Daniel et al. (Sept. 2015). *Authorship verification, combining linguistic features and different similarity functions Notebook for PAN at CLEF 2015*.

Castro Castro, Daniel et al. (Sept. 2015). "Authorship Verification, Average Similarity Analysis". In: *Proceedings of the International Conference Recent Advances in Natural Language Processing*. Hissar, Bulgaria: INCOMA Ltd. Shoumen, BULGARIA, pp. 84–90. URL: https://www.aclweb.org/anthology/R15-1012.

Chandra Sekharan, Sindhu (Oct. 2017). *Recent Approaches on Authorship Attribution Techniques-An Overview*.

Chen, Xiaoling et al. (Aug. 2011). *Authorship Similarity Detection from Email Messages*, pp. 375–386. DOI: 10.1007/978-3-642-23199-5{_}28.

Chen, Yihua et al. (June 2009). "Similarity-Based Classification: Concepts and Algorithms". In: *J. Mach. Learn. Res.* 10, 747–776. ISSN: 1532-4435.

Clark, Kevin et al. (Aug. 2019). "What Does {BERT} Look at? An Analysis of {BERT}{'}s Attention". In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Florence, Italy: Association for Computational Linguistics, pp. 276–286. DOI: 10.18653/v1/W19-4828. URL: https://www.aclweb.org/anthology/W19-4828.

Dash, M and H Liu (1997). "Feature Selection for Classification". In: *Intelligent Data Analysis* 1, pp. 131–156. ISSN: 1571-4128. DOI: 10.3233/IDA-1997-1302.

Denecke, K, T Risse, and T Baehr (2009). "Text classification based on limited bibliographic metadata". In: *2009 Fourth International Conference on Digital Information Management*, pp. 1–6. ISBN: VO -. DOI: 10.1109/ICDIM.2009.5356767.

Devlin, Jacob et al. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.

Diederich, Joachim et al. (2003). "Authorship Attribution with Support Vector Machines". In: *Applied Intelligence* 19.1, pp. 109–123. ISSN: 1573-7497. DOI: 10.1023/A:1023824908771. URL: https://doi.org/10.1023/A:1023824908771.

Domingos, Pedro (Oct. 2012). "A Few Useful Things to Know about Machine Learning". In: *Commun. ACM* 55.10, 78–87. ISSN: 0001-0782. DOI: 10.1145/2347736.2347755. URL: https://doi.org/10.1145/2347736.2347755.

Famili, A et al. (1997). "Data preprocessing and intelligent data analysis". In: *Intelligent Data Analysis* 1.1, pp. 3–23. ISSN: 1088-467X. DOI: https://doi.org/10.1016/S1088-467X(98)00007-9. URL: http://www.sciencedirect.com/science/article/pii/S1088467X98000079.

Farda Sarbas, Mariam and Claudia Müller-Birn (Aug. 2019). *Wikidata from a Research Perspective – A Systematic Mapping Study of Wikidata*.

Fathi, Mohamed, Noha Adly, and Magdy Nagi (July 2020). "Web Documents Classification Using Text, Anchor, Title and Metadata Information". In:

Fissette, M V M (2010). "Author Identifcation in Short Texts". In:

Freund, Yoav and Robert E Schapire (1996). "Experiments with a New Boosting Algorithm". In: *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*. ICML'96. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 148–156. ISBN: 1558604197.

Gannon, Dennis (Oct. 2019). *A "Chatbot" for Scientific Research: Part 2 -AI, Knowledge Graphs and BERT*. DOI: 10.13140/RG.2.2.22273.81768.

Gashler, M, C Giraud-Carrier, and T Martinez (Dec. 2008). "Decision Tree Ensemble: Small Heterogeneous Is Better Than Large Homogeneous". In: *2008 Seventh International Conference on Machine Learning and Applications*, pp. 900–905. DOI: 10.1109/ICMLA.2008.154.

Gaustad, Tanja (2004). *Linguistic Knowledge and Word Sense Disambiguation*. Groningen.

Ge, Zhenhao, Yufang Sun, and Mark J T Smith (2016). *Authorship Attribution Using a Neural Network Language Model*.

Gómez-Adorno, Helena et al. (Aug. 2016). "Automatic Authorship Detection Using Textual Patterns Extracted from Integrated Syntactic Graphs". eng. In: *Sensors (Basel, Switzerland)* 16.9, p. 1374. ISSN: 1424-8220. DOI: 10.3390/s16091374. URL:

https://pubmed.ncbi.nlm.nih.gov/27589740https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5038652/.

Goutte, Cyril and Eric Gaussier (Apr. 2005). *A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation*. Vol. 3408, pp. 345–359. DOI: 10.1007/978-3-540-31865-1{_}25.

Grave, Edouard et al. (2018). "Learning Word Vectors for 157 Languages". In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Gressel, Gilad et al. (2014). "Ensemble learning approach for author profiling". In: *Notebook for PAN at CLEF*, pp. 401–412.

Grieve, Jack (2007). "Quantitative authorship attribution: An evaluation of techniques". In: *Literary and linguistic computing* 22.3, pp. 251–270.

Grzegorczyk, Karol (2019). *Vector representations of text data in deep learning*.

Han, Yiqiu and Wai Lam (2003). "Exploiting Heterogeneous Features for Classification Learning BT - Intelligent Data Engineering and Automated Learning". In: ed. by Jiming Liu, Yiu-ming Cheung, and Hujun Yin. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 177–184. ISBN: 978-3-540-45080-1.

Hughes, Rachael A et al. (Mar. 2019). "Accounting for missing data in statistical analyses: multiple imputation is not always the answer". In: *International Journal of Epidemiology* 48.4, pp. 1294–1304. ISSN: 0300-5771. DOI: 10.1093/ije/dyz032. URL: https://doi.org/10.1093/ije/dyz032.

Imtiaz, Syed and Sirish Shah (Oct. 2008). "Treatment of missing values in process data analysis". In: *The Canadian Journal of Chemical Engineering* 86, pp. 838–858. DOI: 10.1002/cjce.20099.

James, Gareth et al. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated. ISBN: 1461471370.

Jiao, Shuming et al. (2020). "Does deep learning always outperform simple linear regression in optical imaging?" eng. In: *Optics express* 28.3, pp. 3717–3731. ISSN: 1094-4087. DOI: 10.1364/oe.382319. URL: http://europepmc.org/abstract/MED/32122034https://doi.org/10.1364/OE.382319.

Kale, Sunil and Rajesh Prasad (Apr. 2017). "A Systematic Review on Author Identification Methods". In: *International Journal of Rough Sets and Data Analysis* 4, pp. 81–91. DOI: 10.4018/IJRSDA.2017040106.

Kang, Hyun (May 2013). "The prevention and handling of the missing data". eng. In: *Korean journal of anesthesiology* 64.5, pp. 402–406. ISSN: 2005-6419. DOI: 10.4097/kjae.2013.64.5.402. URL: https://pubmed.ncbi.nlm.nih.gov/23741561https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3668100/.

Karen, Sparck Jones (Jan. 1972). "A Statistical Interpretation of Term Specificity and Its Application in Retrieval". In: *Journal of Documentation* 28.1, pp. 11–21. ISSN: 0022-0418. DOI: 10.1108/eb026526. URL: https://doi.org/10.1108/eb026526.

Khan, Jamal Ahmad (2017). "Style Breach Detection: An Unsupervised Detection Model". In: *CLEF*.

Kilinç, Deniz (2016). "The Effect of Ensemble Learning Models on Turkish Text Classification". In: *Celal Bayar Üniversitesi Fen Bilimleri Dergisi* 12.2.

Koppel, Moshe, Jonathan Schler, and Shlomo Argamon (2009). "Computational methods in authorship attribution". In: *Journal of the American Society for information Science and Technology* 60.1, pp. 9–26.

Kovaleva, Olga et al. (Nov. 2019). "Revealing the Dark Secrets of {BERT}". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 4365–

4374. DOI: 10.18653/v1/D19-1445. URL: https://www.aclweb.org/anthology/D19-1445.

Kraaij, Wessel and Ren Pohlmann (Apr. 1996). "Porter's stemming algorithm for Dutch". In: *Informatiewetenschap 1994: Wetenschappelijke Bijdragen Aan de Derde STINFON Conferentie*.

Lee, K.-Y et al. (Jan. 2017). "Comparison and analysis of linear regression & artificial neural network". In: *International Journal of Applied Engineering Research* 12, pp. 9820–9825.

Li, Bofang et al. (2019). "Scaling Word2Vec on Big Corpus". In: *Data Science and Engineering* 4.2, pp. 157–175. ISSN: 2364-1541. DOI: 10.1007/s41019-019-0096-6. URL: https://doi.org/10.1007/s41019-019-0096-6.

Libovický, Jindřich, Rudolf Rosa, and Alexander Fraser (2019). *How Language-Neutral is Multilingual BERT?*

Litvinova, Tatiana et al. (Dec. 2016). "Profiling a set of personality traits of text author: What our words reveal about us". In: *Research in Language* 14. DOI: 10.1515/rela-2016-0019.

Liu, Shigang et al. (2017). "Addressing the class imbalance problem in Twitter spam detection using ensemble learning". In: *Computers & Security* 69, pp. 35–49. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2016.12.004. URL: http://www.sciencedirect.com/science/article/pii/S0167404816301754.

Maalej, Walid et al. (2016). "On the automatic classification of app reviews". In: *Requirements Engineering* 21.3, pp. 311–331. ISSN: 1432-010X. DOI: 10.1007/s00766-016-0251-9. URL: https://doi.org/10.1007/s00766-016-0251-9.

Mason, Llew et al. (Jan. 1999). *Boosting Algorithms as Gradient Descent.* Pp. 512–518.

Mendenhall, Thomas Corwin (1887). "The characteristic curves of composition". In: *Science* 9.214, pp. 237–249.

Mikolov, Tomas et al. (2013). *Distributed Representations of Words and Phrases and their Compositionality*.

Mirończuk, Marcin Michał and Jarosław Protasiewicz (2018). "A recent overview of the state-of-the-art elements of text classification". In: *Expert Systems with Applications* 106, pp. 36–54. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2018.03.058. URL: http://www.sciencedirect.com/science/article/pii/S095741741830215X.

Mohsen, Ahmed, Nagwa El-Makky, and Nagia Ghanem (2016). "Author Identification Using Deep Learning". In: pp. 898–903. DOI: 10.1109/ICMLA.2016.0161.

Naili, Marwa, Anja Habacha Chaibi, and Henda Hajjami Ben Ghezala (2017). "Comparative study of word embedding methods in topic segmentation". In: *Procedia Computer Science* 112, pp. 340–349. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.08.009. URL: http://www.sciencedirect.com/science/article/pii/S1877050917313480.

Nazábal, Alfredo et al. (2020). "Handling incomplete heterogeneous data using VAEs". In: *Pattern Recognition* 107, p. 107501. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2020.107501. URL: http://www.sciencedirect.com/science/article/pii/S0031320320303046.

Ostendorff, Malte et al. (2019). *Enriching BERT with Knowledge Graph Embeddings for Document Classification*.

Otterbacher, Jahna (2010). "Inferring Gender of Movie Reviewers: Exploiting Writing Style, Content and Metadata". In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. CIKM '10. New York, NY, USA: Association for Computing Machinery, 369–378. ISBN: 9781450300995. DOI: 10.1145/1871437.1871487. URL: https://doi.org/10.1145/1871437.1871487.

Park, Youngtae (1994). "A comparison of neural net classifiers and linear tree classifiers: Their similarities and differences". In: *Pattern Recognition* 27.11, pp. 1493–1503. ISSN: 0031-3203. DOI: https://doi.org/10.1016/0031-3203(94)90127-9. URL: http://www.sciencedirect.com/science/article/pii/0031320394901279.

Pervaz, Ifrah et al. (2015). "Identification of Author Personality Traits using Stylistic Features: Notebook for PAN at CLEF 2015". In: *CLEF*.

Pires, T, Eva Schlinger, and Dan Garrette (2019). "How multilingual is Multilingual BERT?" In: *ArXiv* abs/1906.0.

Porter, Martin (2001). "Snowball: A language for stemming algorithms". In:

Provost, F (2008). "Machine Learning from Imbalanced Data Sets 101". In:

Qaiser, Shahzad and Ramsha Ali (July 2018). "Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents". In: *International Journal of Computer Applications* 181. DOI: 10.5120/ijca2018917395.

Qian, Tie-Yun et al. (2015). "Review Authorship Attribution in a Similarity Space". In: *Journal of Computer Science and Technology* 30.1, pp. 200–213. ISSN: 1860-4749. DOI: 10.1007/s11390-015-1513-6. URL: https://doi.org/10.1007/s11390-015-1513-6.

Ráez, Arturo Montejo, Luis Alfonso Ureña López, and Ralf Steinberger (2005). "Text Categorization using bibliographic records: beyond document content". In: *Procesamiento del Lenguaje Natural* 35.

Ramos, Juan Enrique (2003). "Using TF-IDF to Determine Word Relevance in Document Queries". In:

Raschka, Sebastian (July 2014). "About Feature Scaling and Normalization and the effect of standardization for machine learning algorithms". In: DOI: 10.13140/2.1.4245.1849.

Ratcliff, John W. and David Metzener (1988). *Pattern Matching: The Gestalt Approach*. URL: https://xlinux.nist.gov/dads/HTML/ratcliffObershelp.html.

Rexha, Andi et al. (2018). "Authorship identification of documents with high content similarity". In: *Scientometrics* 115.1, pp. 223–237. ISSN: 1588-2861. DOI: 10.1007/s11192-018-2661-6. URL: https://doi.org/10.1007/s11192-018-2661-6.

Richter, Georg and Andrew MacFarlane (2005). "The impact of metadata on the accuracy of automated patent classification". In: *World Patent Information* 27.1, pp. 13–26. ISSN: 0172-2190. DOI: https://doi.org/10.1016/j.wpi.2004.08.001. URL: http://www.sciencedirect.com/science/article/pii/S0172219004001061.

Rokach, Lior (2010). "Ensemble-based classifiers". In: *Artificial Intelligence Review* 33.1, pp. 1–39. ISSN: 1573-7462. DOI: 10.1007/s10462-009-9124-7. URL: https://doi.org/10.1007/s10462-009-9124-7.

Ruder, Sebastian, Parsa Ghaffari, and John G Breslin (2016). *Character-level and Multi-channel Convolutional Neural Networks for Large-scale Authorship Attribution*.

Salton, G, A Wong, and C S Yang (Nov. 1975). "A Vector Space Model for Automatic Indexing". In: *Commun. ACM* 18.11, 613–620. ISSN: 0001-0782. DOI: 10.1145/361219.361220. URL: https://doi.org/10.1145/361219.361220.

Schwartz, Roy et al. (2013). "Authorship attribution of micro-messages". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1880–1891.

Sebastiani, Fabrizio (2002). "Machine learning in automated text categorization". In: *ACM computing surveys (CSUR)* 34.1, pp. 1–47.

Seroussi, Yanir, Ingrid Zukerman, and Fabian Bohnert (Jan. 2011). *Authorship Attribution with Latent Dirichlet Allocation*, pp. 181–189.

— (2014). "Authorship attribution with topic models". In: *Computational Linguistics* 40.2, pp. 269–310.

Shi, Congying, Chaojun Xu, and Xiaojiang Yang (2009). "Study of TFIDF algorithm". In: *Journal of Computer Applications* 29.6, pp. 167–170.

Shrestha, Prasha et al. (Apr. 2017). "Convolutional Neural Networks for Authorship Attribution of Short Texts". In: *Proceedings of the 15th Conference of the {E}uropean Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 669–674. URL: https://www.aclweb.org/anthology/E17-2106.

Stamatatos, Efstathios (Mar. 2009). "A Survey of Modern Authorship Attribution Methods". In: *J. Am. Soc. Inf. Sci. Technol.* 60.3, 538–556. ISSN: 1532-2882.

Sudheep Elayidom, M et al. (Sept. 2013). "Text Classification for Authorship Attribution Analysis". In: *Advanced Computing: An International Journal* 4.5, 1–10. ISSN: 2229-6727. DOI: 10.5121/acij.2013.4501. URL: http://dx.doi.org/10.5121/acij.2013.4501.

Sulea, Octavia-Maria et al. (2017). *Exploring the Use of Text Classification in the Legal Domain*.

Suleiman, Dima, Arafat Awajan, and Nailah Al-Madi (Oct. 2017). *Deep Learning Based Technique for Plagiarism Detection in Arabic Texts*. DOI: 10.1109/ICTCS.2017.42.

Sun, Chi et al. (2019). "How to Fine-Tune BERT for Text Classification? BT - Chinese Computational Linguistics". In: ed. by Maosong Sun et al. Cham: Springer International Publishing, pp. 194–206. ISBN: 978-3-030-32381-3.

Tschuggnall, Michael et al. (2017). "Overview of the Author Identification Task at PAN-2017: Style Breach Detection and Author Clustering". In: *CLEF*.

Tulkens, Stephan, Chris Emmery, and Walter Daelemans (May 2016). "Evaluating Unsupervised Dutch Word Embeddings as a Linguistic Resource". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Ed. by Nicoletta Calzolari (Conference Chair) et al. Paris, France: European Language Resources Association (ELRA). ISBN: 978-2-9517408-9-1.

Vries, Wietse de et al. (Dec. 2019). "{BERTje}: {A} {Dutch} {BERT} {Model}". In: *arXiv:1912.09582 [cs]*. URL: http://arxiv.org/abs/1912.09582.

Wall, Michael E, Andreas Rechtsteiner, and Luis M Rocha (2003). "Singular value decomposition and principal component analysis". In: *A practical approach to microarray data analysis*. Springer, pp. 91–109.

Wang, Lidong (2017). "Heterogeneous data and big data analytics". In: *Automatic Control and Information Sciences* 3.1, pp. 8–15.

Wang, Pu et al. (2009). "Using Wikipedia knowledge to improve text classification". In: *Knowledge and Information Systems* 19.3, pp. 265–281. ISSN: 0219-3116. DOI: 10.1007/s10115-008-0152-4. URL: https://doi.org/10.1007/s10115-008-0152-4.

Wang, Sun-Chong (2003). "Artificial neural network". In: *Interdisciplinary computing in java programming*. Springer, pp. 81–100.

Wolpert, D H and W G Macready (Apr. 1997). "No free lunch theorems for optimization". In: *IEEE Transactions on Evolutionary Computation* 1.1, pp. 67–82. ISSN: 1941-0026. DOI: 10.1109/4235.585893.

Wu, Shijie and Mark Dredze (2020). *Are All Languages Created Equal in Multilingual BERT?*

Yuan, G, C Ho, and C Lin (2012). "Recent Advances of Large-Scale Linear Classification". In: *Proceedings of the IEEE* 100.9, pp. 2584–2603. ISSN: 1558-2256. DOI: 10.1109/JPROC.2012.2188013.

Yule, C Udny (2014). *The statistical study of literary vocabulary*. Cambridge University Press.

Zhang, Tong (2004). "Solving Large Scale Linear Prediction Problems Using Stochastic Gradient Descent Algorithms". In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML '04. New York, NY, USA: Association for Computing Machinery, p. 116. ISBN: 1581138385. DOI: 10.1145/1015330.1015332. URL: https://doi.org/10.1145/1015330.1015332.

Zhang, Zhengyan et al. (July 2019). "{ERNIE}: Enhanced Language Representation with Informative Entities". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 1441–1451. DOI: 10.18653/v1/P19-1139. URL: https://www.aclweb.org/anthology/P19-1139.

Zheng, Alice and Amanda Casari (2018). *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc."

Zhou, Zhi-Hua (2012). *Ensemble Methods: Foundations and Algorithms*. 1st. Chapman & Hall/CRC. ISBN: 1439830037.