UNIVERSITY OF COPENHAGEN

A thesis presented to the Faculty of Humanities in partial fulfillment of the requirements for the degree

Master of Science in IT & Cognition

An Explorative Study on Creating a General Purpose Text Encoding Applied to Multiple Tasks

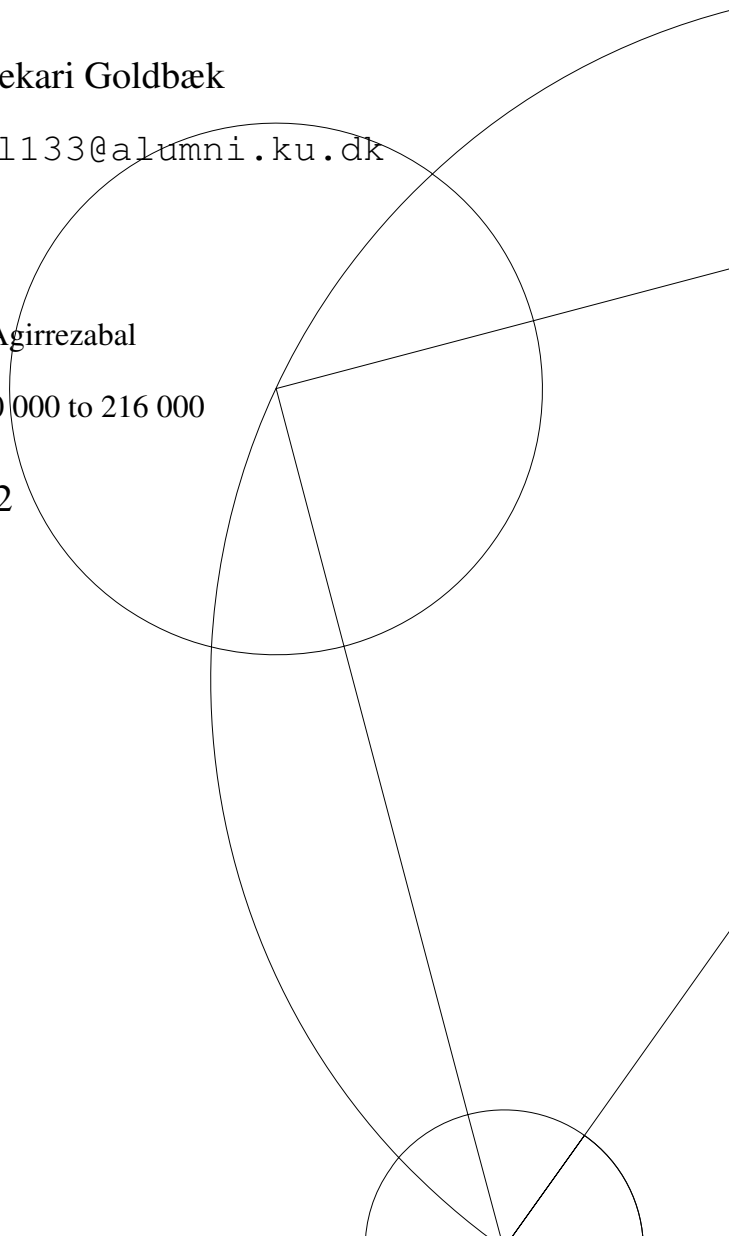
Niels Krog, Yasmin Shekari Goldbæk

lkt259@alumni.ku.dk, vsl133@alumni.ku.dk

Supervisor: Manex Agirrezabal

209259 characters of 180 000 to 216 000

May 2022



ACKNOWLEDGEMENTS

We would like to thank Copyfighter IVS[1] for providing us with data and Ankiro ApS[2] for offering us an office space along with insightful discussions. Especially, we thank Esben Alfort for giving feedback and advice during the project.

1 https://copyfighter.com/
2 https://ankiro.dk

ABSTRACT

The aim of this thesis is to apply a general-purpose encoding of Danish newspaper articles to authorship attribution, newspaper attribution, and headline generation with the aim of assessing the scalability of the encoding across tasks. From a novel dataset of 800k Danish online news articles, we test manually and automatically extracted features separately, as well as a combination of the two. Literature on attributing newspapers and using a handcrafted feature encoding for headline generation is scarce, creating incentive for an investigation of these fields. We found that manual features and a combination of manual and automatic features scale well across classification tasks, with performances consistently above baseline. The results obtained from headline generation were inconclusive as the training time for the model exceeded the thesis deadline.

The source code for this project is available online[3].

3 https://github.com/kroglkt/articlencoding

CONTENTS

1

INTRODUCTION

With the continual growth and availability of text online, the relevance of language understanding for text analysis has increased. Language understanding methods often rely on a feature extraction process which can be done in a manual or automatic manner and has wide range of applications, e.g. machine translation, information retrieval, and text summarization. A feature is a piece of information which represents certain characteristics from a text and can be at different levels of abstraction, i.e. from counting the words in the text to estimating its sentiment. Features are often vector representations of specific text characteristics, and a collection of features is commonly referred to as a text encoding.

Manual feature extraction relies on hand-picking relevant features of documents and fine-tuning to specific tasks and corpora. While such features can be powerful, the process is time-consuming and often tailored to the task at hand and the rapid increase in textual data calls for more scalable, automatic methods. In an attempt to improve scalability and limit manual labor, researchers have used Recurrent Neural Networks (RNN), particularly Long Short-Term Memory (LSTM) networks to alleviate the dependency on manually extracted features. Such networks can, in an unsupervised manner, learn abstract text representations when used in an autoencoder architecture, where the goal is to reconstruct the input. Unsupervised learning refers to a problem where the target value is not explicitly stated, e.g. deriving topic from a text automatically, whereas in a supervised scenario, the target value is available, i.e. a human has annotated the topic of the text.

LSTM autoencoders can be used to encode texts automatically and considers long term dependencies between words which has shown to perform well on different Natural Language Processing (NLP) tasks, e.g. lexical utterance classification [78], named entity recognition [32] and machine translation [96].

Another approach is the use of transformers [98] which have, since their emergence in 2017, been utilized for a number of NLP tasks that build on language understanding. Transformers utilize the concept of attention in a different manner to RNNs which has shown promising performance on

many different tasks, e.g. machine translation [98], document generation [58], syntactic parsing [45], natural language inference [76], and question answering [21]. A transformer, like an LSTM autoencoder, can be trained to automatically generate a feature representation of a given text.

In this study, we will use a novel corpus consisting of Danish articles from different online newspapers, tabloids, and niche newspapers and websites. The corpus contains articles from online newspapers and websites such as Jyllands-Posten[4], Information[5], BT[6], Berlingske[7], Børsen[8] and Computerworld[9]. We will be using newspaper, source, and domain interchangeably throughout this paper. The data originates from the tool Copyfighter from the Danish company[10] of the same name. Copyfighter seeks to help journalists, bloggers, and other companies detect plagiarized articles from their copyrighted documents. Newspaper corpora is relevant to work with as a lot of online news sources are readily available for everyone and it is therefore worth providing insights into these sources as anyone can be influenced by them. Transparency with respect to publicly available knowledge is therefore an important ethical topic, although socio-political implications are out of the scope for this project.

To the best of our knowledge, only few studies [33, 63] have used Danish corpora for feature extraction which is therefore a novel aspect of our project that can potentially provide insights and aid to future language applications.

Our dataset contains metadata from which we have derived this study's focus in order to test the capabilities of the feature extraction. Each article entry contains an author, newspaper name, and a headline, making room for three different tasks: authorship attribution, newspaper attribution, and headline generation whereof the first two are classification tasks and the third is a generation task.

In order to circumvent the aforementioned problems of labor-intensive processes for feature extraction, we attempt to make a general purpose encoding which can scale across the three different tasks. In doing so, we examine the differences between using manually extracted features, automatically extracted features, and a combination of these two.

This approach will contribute with results that can extend the findings of previous studies in order to improve language systems further. By using the same encoding for all three tasks, we investigate scalability of commonly used features across targets and task types.

4 https://jyllands-posten.dk/
5 https://www.information.dk/
6 https://www.bt.dk/
7 https://www.berlingske.dk/
8 https://borsen.dk/
9 https://www.computerworld.dk/
10 https://copyfighter.com

Authorship attribution is a well-documented problem, which consists of identifying the author of a given text based on an assemblage of different known authors. Many studies have tackled authorship attribution [67, 6, 33, 84, 22], and several of those studies have obtained reliable results where models were able to successfully identify the author.
The task has several applications, e.g. finding the rightful authors of disputed works. It can also be an extension in the fight against cyber-bullying and online threats as the ease of anonymity has grown [24]. Authorship attribution has been researched and experimented with across different textual genres, e.g. the federalist papers [67], student texts [6, 33], and newspaper articles [84, 22]. Another practical application of authorship attribution is detection of plagiarism which is useful in different educational and publishing institutions. On a note of plagiarism, newspaper attribution has similar relevance.

Newspaper attribution is the task of attributing the newspaper from which a text originates, and it can help detect plagiarism on a domain level. Certain newspapers may be subject to a lot of plagiarism as their articles may be copied and distributed across different websites without consent. As mentioned, the newspaper corpus in this project was collected with the Copyfighter tool in which the purpose is plagiarism detection, and our project therefore also has potential practical use for journalists and businesses. Another use could be as a tool for freelance journalists. If they have written an article but are unsure which newspaper would be most interested in it, the newspaper attribution could advise the journalist on which newspapers to contact.
Newspaper attribution has yet to be well documented in literature, a gap we help bridge in this study. When identifying authors, stylometry, i.e. linguistic style, is especially important and has been documented by several studies as particularly useful in this task [13, 106, 1]. When identifying newspapers, however, there are more aspects to consider as individual newspapers have different target groups, different political attitudes, and different topical preferences. Newspaper attribution may thus benefit from utilizing some content driven features, features that are often omitted in authorship attribution because authors often are not bound to specific topics. Both author and newspaper attribution are classification problems, and as mentioned, the selection and extraction of features often requires manual hand-picking of features that are useful for the specific datasets which decreases the scalability and versatility of the models.

Our third task is a generation problem. Headline generation can be seen as a task within the field of text summarization where the objective is to generate coherent, informative summaries of documents. The field of natual language generation (NLG) is quite active, so there are many

different approaches and room for exploration. Some previous studies have focused on generating informative headlines [102], whereas others have paid more attention to generating sensational headlines [101], otherwise known as “clickbait”. To the best of our knowledge, no previous studies have attempted to use a manual encoding to generate headlines which is what we seek to do. Our focus, therefore, is not on generating exclusively informative or sensational headlines but rather to evaluate headline predictions from using a manual encoding like the one used for classification tasks, in place of automatically generated encodings.

Effective headline generation would be a useful tool for journalists, as a means of inspiration for a proper headline for their article. In a time where people are bombarded with information, getting a reader’s attention is difficult, hence headlines play a large role for attracting readers.

The possibilities within text processing tools increases with the substantial amount of available text, and text summarization can be a helpful means of providing readers with insightful summaries without putting too much strain on manual labor. Successful methods are also influential in the challenge of natural language understanding (NLU) because condensed representations of text ultimately need to encompass the core meaning which is not an easy undertaking when you consider the semantic diversity cross-linguistically. The field of text summarization is an active field of research, meaning it is not as established as authorship attribution. The task of headline generation, using extracted features that are useful for authorship attribution, is therefore an exploratory task.

As described, each of the three tasks we focus on has an array of applications and contribute to solving relevant issues ensuing from the surge of online texts and documents.

Considering all tasks from a collected perspective, i.e. using the same feature extraction process, is relevant as it allows us to interpret which elements of the language the different tasks benefit from and whether they, in fact, might benefit from the same features. Additionally, by using a manual text encoding for headline generation, we graze the task from a new perspective. In order to direct our project towards the shortcomings of the field, we propose the following research question:

How can we create a general purpose feature encoding for authorship attribution, newspaper attribution, and headline generation?

To answer our main research question, we pose the following sub-questions:

1. *Will manually extracted features, automatically extracted features, or a combination of both generalize best?*
2. *Which of the manually extracted features has the highest impact on the results?*
3. *How do the newspapers relate to each other? Does our feature representation reflect real world similarities between newspapers?*
4. *How will a manual encoding perform for headline generation?*

In answering this research question, we get to know how well features perform across different tasks, focusing on manually extracted features. The application of the same manual encoding for a wide range of tasks is uncommon, as manually extracted features tend to be tailored to a specific task. By performing the tasks using these encodings, we will be able to show whether commonly used textual features scale across types of tasks, whereof two of them have a notable novelty aspect: newspaper attribution and headline generation.

The rest of the paper is structured as follows: in chapter 2 we introduce the state of the art in NLP as well as previous works related to the aforementioned tasks and present different methodologies and features used. In chapter 3 we introduce our own methodology, including the pre-processing steps we took to avoid bias in the data, and design decisions prior to implementing the models. In chapter 4 we describe our implementation. Following implementation, we present initial classification results, and thereafter the final results from our models in chapter 5 in form of quantitative measurements, visualizations and qualitative evaluation of the generated headlines. Our findings are discussed and interpreted in chapter 6 along with the limitations of our methodology of choice and future perspectives. Lastly, we offer concluding remarks in chapter 7.

Summary 1 - Introduction

In this introduction we stated the motivation behind taking up the project of creating a general purpose text encoding for three different tasks: authorship attribution, newspaper attribution and headline generation. Our textual encodings will be in the form of manually extracted features, automatically extracted features and a combination of the two. The processes will be performed on a novel Danish newspaper corpus, which has not been done before to the extent of our knowledge.

The key points from the introduction are:

- The formal research question was established along with a series of sub-questions which help answer the research question.
- Using the same encoding for three different tasks will show how well common textual features generalize.
- Previous headline generations have not utilized commonly used manually extracted features, which we will attempt.
- Little previous literature has been revolving around NLP tasks in the Danish language.
- The literature on newspaper attribution is scarce, motivating the use of common textual features.
- Authorship and newspaper attribution can be used for plagiarism detection.
- Headline generation can be a useful tool for journalists for coming up with a fitting headline.

2

RELATED WORKS

In this chapter we present the findings of related works which lay the base for our study and expectations. We first present an overview on current state of the art (SOA) methods for text analysis and encoding in section 2.1. In section 2.2 we present previous studies on authorship attribution, after which we investigate studies relevant for newspaper attribution section 2.3. Studies on headline generation are presented in section 2.4. In section 2.5 we present the linguistic features from previous studies in more detail, and finally, the key points are summed up in Summary 2.

2.1 TEXT ENCODINGS

With the rapid growth of online texts, the need for being able to automatically analyze and interpret characteristics within texts has become an essential ability. From being able to detect spam to analyzing the public opinion of a brand, NLP is useful in a wide range of applications. Such applications often utilize machine learning (ML), which can be split into three learning styles: supervised, unsupervised, and reinforced. Arguably, supervised learning is the most common type, where an ML model is trained using labelled data, e.g. showing the model the real answer to its task. In some cases, this value may not be available, and unsupervised learning may be applied. In reinforcement learning, the model will engage a task on its own without knowing the end-goal, learning it through experimenting, e.g. teaching a robot to walk. This project has a general focus on supervised learning with some insight in unsupervised learning.

Box 1 - Supervised and Unsupervised Learning

Among machine learning methods are supervised and unsupervised learning. Imagine a task, where the machine learning model should classify an animal as either duck or rhinoceros based on weight and color. In a supervised learning scenario, the target value is known, i.e. it is known whether a given animal is a rhinoceros or a duck. The learning model can then make predictions and learn to be better at it based on the target values. In an unsupervised problem the target value is not available, but the model would learn to separate the two animals based on what it knows about them. A heavy animal with grey color differs from a light animal with various colors. It then knows there are two different animals but may not know exactly what type of animal. The large grey animal could have been an elephant for what we know. Supervised learning tasks require the data to be labelled, which is not always the case, and it may be a laborious process for more complex tasks than the animal example.

Most supervised learning algorithms need inputs of fixed lengths in order to learn from them, however many types of data do not follow this paradigm. In cases of e.g. audio, imagery, or text, supervised learning algorithms need abstract representations of the data in the form of fixed length feature vectors [96].

These vectors have higher level of abstraction than the raw data and is optimally smaller in size than the raw data, but may not be. For example, identifying a car in an image, the learning algorithm would have a hard time interpreting the raw pixel values. An example of a useful feature extraction could be the identification of tires, simpler objects often associated with cars. The learning algorithm will now perform better, as the feature vector may state that two wheels are visible in the image. Then surely, it is not an image of a frog. Such feature vectors are essential for NLP as well where features could be bags-of-words (counting occurrences of each word in a document) or n-grams. Previously used features for authorship attribution and other tasks are expanded on in section 2.5.

Box 2 - N-grams

N-grams is a common feature in NLP and consists of token collocations of various lengths, N. Splitting the sentence *open the pod bay doors*[a] into word n-grams with *n=2* would yield *(open, the), (the, pod), (pod, bay), (bay, doors)*. N-grams are not limited to words, but can also be Parts-of-Speech (POS) tags or characters. The n-grams with low N are commonly referred to as unigrams (1-gram), bigrams (2-gram), trigrams (3-gram) and quadgrams (4-gram).

a Not taking padding into consideration. Notice, that the word *open* should occur with a start token and doors with an end token.

Years of research has resulted in clever and essential hand-crafted features for NLP. However, the process of identifying and extracting such features is labor-intensive and does not necessarily scale well. Features from texts are often task-dependent, meaning one feature might be useful for one project and not for another. Luckily, this process can be automated with autoencoders, which automatically learns feature representations of e.g. text [70].

Autoencoders were first introduced in 1986 by Rumelhart et al. [80], where they were some of the first to investigate unsupervised learning. The autoencoder is a neural network trained to create abstract, condensed representations of its input. The model will reduce the dimensions in its hidden layers and then reconstruct the input in its output layer. The concept of such architecture for encoding of an image is seen in Figure 1, showing the general concept of autoencoding used on images for easy visualization. The leftmost image is the input which is encoded using the encoder. The decoder reconstructs the image from the encoding, creating the rightmost image. The reconstruction is not perfect, but the general shape of the digit has been preserved.

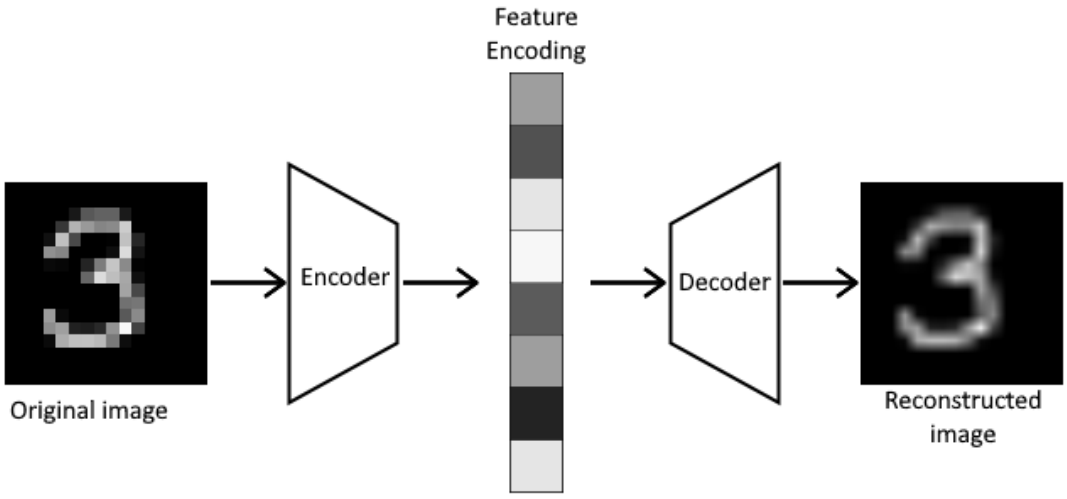


Figure 1: The general concept of autoencoders, reconstructing an image of a hand drawn digit.

The autoencoder is an instance of the encoder-decoder architecture. Where an encoder-decoder will map an input to an output (e.g. English input, Cantonese output), the autoencoder will reconstruct the input in its output (e.g. English input, English output) [66]. The encoder and decoder parts are neural networks and if the autoencoder were to use linear operations, the encoder would produce the same representation as principal component analysis (PCA) which is a common method for feature reduction [9, 88]. However, the neural networks are able to learn non-linear representations using non-linear operations. An autoencoder generally has difficulties reconstructing random input data as it will learn correlations within the input data [70]. This caveat is found in all types of compression, where the more random the input is the less compressed the data can be.

The learning of the feature representation is an unsupervised learning process, meaning none of the training data is labeled, as the input data is the target data itself, i.e. the goal being to reconstruct the input data as accurately as possible. The purpose of the encoder is to downscale the input, creating a bottleneck in the architecture. The decoder will then upscale the feature representation into an output, as seen in the left network in Figure 2. Were the shape of the feature vector not reduced or even upscaled, the network would likely not learn an abstract representation of the input, but rather pass the input through to the output, learning only the identity function [9]. However, as will be explained in subsection 2.1.1, in some architectures, expanding the size of the hidden layer may be beneficial.

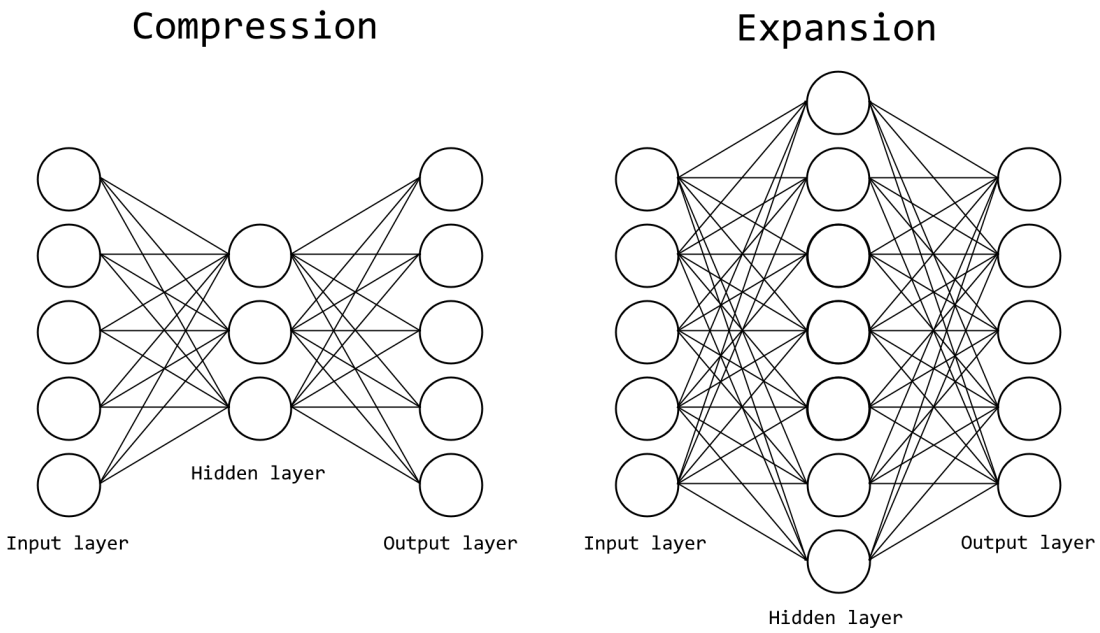


Figure 2: The architecture of autoencoders with a compressed layer and expanded hidden layer, i.e. feature encoding. The right model would be able to pass through the input data to the output layer, unchanged, not learning features.

The features learned from the encoder are in a latent space and each variable in the feature is a latent variable. Such variable may be of high abstraction and can be correlated with one or more input variables [88]. An example of such variable for encoded text could be the sentiment.
When the autoencoder has been sufficiently trained, the decoder can be removed, leaving only the encoder, which produces the abstract feature encoding. This encoding can then be used as an input vector for various learning models.

2.1.1 *Types of Autoencoding*

The autoencoder architecture described so far can be referred to as the simple autoencoder [88]. Were the hidden layer (i.e. feature representation) to be larger than the input data, a dropout layer or activation penalty can be used to create a sparse autoencoder [70, 88]. In this case, a large hidden layer will be made, but only a small layer will be passed through to the next layer. This process will often result in better feature representations, as it forces the network to learn redundancy [88]. In other terms, the network is still unable to memorize the input data but is allowed to extract

even more features than with the bottleneck[11]. Autoencoders can be stacked to create even higher abstraction features [88, 9].

Adding noise to the input data (and leaving the target data untouched) will create a more robust autoencoder which can be used for error correction [9, 88]. Such model is called a denoising autoencoder which can be extended into a contractive autoencoder if explicit regularization is added [88], i.e. penalize instances where small changes in the input leads to large changes in the encoding[11].

One very successful autoencoder is the variational autoencoder, which follows variational Bayes inference and generates data through a probabilistic distribution [9]. For example, with the decoder recreating a face, the feature vector may have a latent variable stating how much the person is smiling in a space from -1 to 1. With a variational autoencoder, this value is a probability density function (e.g. normal distribution), rather than a discrete value[11].

Autoencoders have a wide range of applications. They can be used for generation, clustering, and classification of data [9]. Furthermore, it can be used in anomaly detection, recommendation systems and general dimensionality reduction [9] (or compression). In general, it can be used as an unsupervised feature extraction tool to create input vectors for other models.

2.1.1.1 *LSTM Autoencoder*

One successful variation of autoencoders is the Long Short-Term Memory (LSTM) autoencoder which works well on sequenced data such as text, where long range temporal dependencies often occur [96], e.g. referring to a person in the end of a sentence, who was introduced in the beginning of the sentence. The LSTM is a type of Recurrent Neural Network (RNN) which uses time steps as inputs. The LSTM was first introduced by Hochreiter and Schmidhuber in 1997 [34], addressing the exploding and vanishing gradient problem, which refers to the error signals degrading during back propagation, i.e. the common method of training neural networks. The LSTM is designed to overcome such errors and is able to bridge time intervals which are a thousand time steps apart [34], however, whether this is the case in practical scenarios is up for debate. In 2013, Graves et al. employed a bidirectional LSTM (Bi-LSTM) to speech recognition [107]. A Bi-LSTM encodes the sequence from both directions, which allows for better context understanding. For example, given the start of a text: *The boys went to...* predicting the next word would be guesswork. But adding the back to front sequence: *... and then they got out of the pool*, adds context to the first sentence, allowing for a much better prediction[12]. In 2014, Cho et al. [17] proposed the gated recurrent unit

11 https://www.jeremyjordan.me/autoencoders/ (visited March 23rd)

12 https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0 (visited April 3rd)

(GRU) as a simplification of the LSTM, which performed well on translation tasks. While GRUs are simpler to implement and compute than LSTMs, LSTM autoencoders are still widely used. Such networks can have an **attention** mechanism applied to them, a set of weights, which dictate which input elements are the most important for the task. Attention is useful as a long input to the autoencoder may not be able to fit in the latent representation and attention would better capture relevant information in the limited space.

Box 3 - Attention

Attention is the ability to control where limited computing power is allocated and is found in humans and other mammals. The definition of attention is not clear-cut, as it depends on the field of study [57]. It is often related to focusing on the most salient elements in the sensory input, where the most salient element is attended first [36]. Applying attention to ML models will make the model generalize better and while the artificial attention is inspired by biological attention, it is not always the same concept [57]. Attention in ML was first introduced with an encoder-decoder architecture, where attentional systems determine which encoded features should be used for decoding. This does not apply to structures with bottlenecks, but rather architectures such as the *sparse autoencoder*, where the attention system selectively determines which weights to use [57]

As mentioned in subsection 2.1.1, the encoder and decoder are neural networks. Were these networks LSTMs, an LSTM autoencoder would be created. Sutskever et al. [96] used an LSTM encoding to translate text from English to French, where it performed well on both short and long sentences and outperformed the SOA statistical machine translators in 2014.

LSTM autoencoders have also been used for anomaly detection in NLP tasks. Saumya and Singh [83] used an LSTM autoencoder to detect fake online reviews. The method was implemented as unsupervised learning, as labeled data in this domain is sparse. To solve this, their model consisted of an LSTM autoencoder followed by clustering. The model was then trained to detect anomalies, successfully excluding reviews that did not follow ordinary review patterns. Similarly to this approach, Kumari et al. [52] used an LSTM autoencoder on online comments to detect cyber-aggression with anomaly detection. Their results were obtained with a decision tree as a classifier using the LSTM encoding as input.

To sum up, autoencoders are models which can automatically perform feature extraction in various inputs, including text. They are capable of producing abstract feature vectors, which can be used as

input for ML models. One type of autoencoder, the LSTM autoencoder, has proven useful for NLP tasks, as they perform well on sequenced data, making it a good candidate for this project.
However, in 2017, Vawani et al. [98] introduced the transformer, an attention-based model following an encoder-decoder architecture, originally created for machine translation [107].

2.1.1.2 *Transformers*

In recent years, transformers have been the subject of interest in the field of NLP as the performance of transformer models have been shown to beat several NLP benchmarks on different tasks [21]. They are considered state of the art and are particularly useful for sequence-to-sequence problems, e.g. translation tasks. Transformers were first introduced by Vaswani et al. [98] where the goal was to circumvent the sequential constraints that are dominant in recurrent models. The sequentiality of recurrent models encounter memory problems when input sequences become longer as there may be an increase in long-term dependencies between words. As mentioned, the LSTM seeks to bypass the memory problem, however, they still suffer from memory constraints in longer sequence lengths. Transformers avoid the sequential computation constraints by foregoing recurrence and relying on an attention mechanism to get global dependencies between input and output.
Transformers rely on a self-attention mechanism that creates sequence representations by relating positions in a sequence. Positional information is given through a positional encoder which outputs a vector (a word embedding) that includes context depending on where a word appears in a sentence. For example, in the sentences *Widow Tweed's fox is red* and *Widow Tweed looks like a fox*, the word *fox* would have different word embeddings due to its position in the sentence. To achieve this, the input is passed to the transformer in parallel, i.e. the whole sentence is passed as input, as opposed to sequentially, word by word, which is the case with RNNs.
In a standard attention mechanism scenario for working with text, each word has an attention score, i.e. which word should the model pay attention to? With self-attention, each word in the input has attention scores for its relation to other words, i.e. the word *him* has a strong attention score to a previously mentioned man, *Jimmy*.
The transformer then uses an encoder layer comprising of multi-head self-attention and a feed-forward network. The positional encoding is passed to the encoder layer where attention vectors for each word in a sequence is computed multiple times in parallel, making it multi-headed. The feed-forward network is then applied to each position in parallel. The parallel operations of transformers also allow for faster training compared to RNNs.

Since transformers were introduced, they have been used for a number of NLP tasks including machine translation [98], text generation [58], and natural language inference [76].
Vaswani et al., who introduced transformers, tested the architecture on English-German and English-French machine translation tasks. For both tasks, the transformer outperformed previous SOA models on BLEU scores [72], underlining the capabilities of an attention-based model. The BLEU (Bilingual Evaluation Understudy) score is an automatic evaluation of machine translation, based on precision, that utilizes n-grams, the size of which is configurable. The metric compares n-grams of the candidate machine translation with the provided references, i.e. gold standard, and counts the matches, where after it divides by the total number of n-grams in the candidate. The candidate translation is better the more n-gram matches it has with the reference.

Box 4 - Precision and Recall

Two commonly used metrics to evaluate binary classifiers are precision and recall. Precision measures which proportion of positive identifications were in fact positive, while recall measures which proportion of true positives were correctly identified[a]. A more manageable example is fishing in a lake with a net. You know there are 100 fish in the lake and catch 80, resulting in an 80% recall. However, you also catch 80 rocks, resulting in a precision of 50%. Changing the net to a smaller one, getting 20 fish and 0 rocks means you have 20% recall, but 100% precision[b]. The metric F1 is also commonly used for evaluating classifiers, as it is based on precision-recall. An F1-score spans from 0.0 to 1.0 and a value of 1.0 means 100% recall and precision, with 0.0 being either 0% recall or precision.

a httpd://developers.google.com/precision-and-recall (visited May 20th)
b https://old.reddit.com/r/datascience/recallprecision(visited May 20th)

In 2018, BERT was introduced by Devlin et al.[21]. It is a transformer model using **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Researchers utilize pre-training where BERT is trained on unlabeled, bidirectional text data, i.e. language representations from both the left and right, alike the bi-LSTM. BERT is then fine-tuned to specific tasks using labeled data. This approach results in the model having general knowledge of language compositions prior to fine-tuning to a specific task, resulting in less computation time when the model is applied. This concept is referred to as transfer learning.
For example, Delvin et al. fine-tuned the model to fit different NLP tasks, including binary classification tasks from the General Language Understanding Evaluation (GLUE) benchmark. BERT outperformed previous SOA on a binary linguistic acceptability classification task, i.e. classifying whether a sentence was acceptable from a grammatical perspective. Additionally, it

performed better than SOA on a task of classifying whether two questions were semantically equivalent. Another study [29] used a transformer-based model for multi-class classification of propaganda techniques and successfully classified which of 14 propaganda techniques was used in a text. The previous achievements in both binary and multi-class classification tasks make transformers a worthwhile approach to use for classification tasks.

So far, we have gone over the SOA models in NLP, including RNNs, LSTM autoencoders, and transformers. Both RNNs and LSTMs take input data in sequence, i.e. word by word, because the current state requires previous states to perform operations. Transformers, on the other hand, allow input to be passed in parallel. They use positional encoding and self-attention which allows the model to include information about words' positions as well as the saliency of each word with respect to other words in the sequence, thereby including long term dependencies. These parallel operations also make transformers faster than traditional RNNs.
In the next section we will introduce some previous studies on the three tasks we focus on, including how some studies have utilized the aforementioned models.

2.2 AUTHORSHIP ATTRIBUTION

Authorship attribution or authorship identification is the process of identifying the author behind a given text. Attribution, identification, and classification will be used interchangeably throughout this work. This task can be dated back to the Middle-Ages [50] and is used to determine the trustworthiness of the text. Today it can be used for a wide variety of tasks, e.g. determining the author of historical texts [82, 91], cyber security [24] or plagiarism detection [82].
The two most common types of tasks revolving around authorship insights are attribution and verification tasks. Where verification is a question of *"Did the same author write X and Y texts?"*, authorship attribution asks, *"Which of our known authors wrote X text?"*. The authorship verification problem is a subset of authorship attribution [50], the latter being the focus of this study.

2.2.1 *Types of Problems*

According to Koppel et al. [50, 49], excluding verification problems, there are three authorship attribution problems:

- Simple authorship attribution problem
- Profiling problem

- Many-candidates problem

The simple authorship attribution problem is the most basic form of authorship attribution with clean data. The author candidate set is limited to a handful and every candidate has multiple texts written. This problem is well understood, and high accuracy can be achieved [49]. However, this scenario rarely occurs with real-world data.

In the profiling problem, there is no candidate set of authors. The task is to create an author profile from the texts available using demographic and psychological analysis of the language used in the texts [50].

The final problem, the many-candidates problem, also known as the needle-in-a-haystack problem, is when the size of the candidate set is in the thousands and the authors may have varying amounts of text written. In this scenario, it is more challenging to achieve good accuracy due to the large number of candidates. With virtually unlimited text available online, this task is highly relevant and is exactly the case with the data used in this project. To solve this issue, Koppel et al. [50] used a combination of a similarity-based method and Support Vector Machines (SVM) for classification and found an increase from 56% up to 94% (at 30% recall) in accuracy on 10 thousand author candidates when allowing the model to classify a text as *Don't know*.

2.2.2 *Features*

There are two different types of features to extract from text, which reveals characteristics of the text: **stylometric features** and **content features**.

Stylometric features reveal the writing style of the author [91]. These features concern the style of the text, whereas content features reveal something about what the text contains. According to Stamatos et al. [91] the stylometric features consist of:

- Lexical features: Word/sentence length, vocabulary richness, word frequencies, word n-grams, spelling mistakes
- Character features: Character types, character n-grams, compression methods.
- Syntactic features: POS-tags, sentence structure.
- Semantic features: Synonyms, semantic dependencies.
- Application specific features: Structural, content-specific, language-specific.

This list has overlapping features with content features. For example, word n-grams will capture the content of the text along with stylometric tendencies. Content features consist of word frequencies, word and character n-grams, hapax legomena etc. This overlap is not of concern, however, as Sari

et al. [82] show, using content features is beneficial when performing authorship attribution of news articles because journalists often have certain topics they prefer writing about. They argue that using only stylometric features is beneficial when attributing authors to texts of the same topic or genre, e.g. law text or movie reviews.

In relation to topics, Seroussi et al. [85] explored the use of topics as a feature for attributing authors. Topic modeling is a statistical approach for identifying which variables are found within a data set [99]. The authors compared results from three topic models: Latent Dirichlet Allocation (LDA), Author-Topic, and Disjoint Author-Document Topic (DADT). They found DADT to yield promising results which is a testament to the usefulness of topics as a feature for authorship attribution.

Bozkurt et al. [12] performed authorship attribution on Turkish newspaper articles using stylometric features, vocabulary diversity, bag-of-words and frequency of function words. For stylometric features, they used the number of sentences and words in the article, the average number of words in a sentence and the whole article, the vocabulary size and frequencies of symbols used (periods, exclamation marks, etc.). They weighted their features using Term Frequency-Inverse Document Frequency (TF-IDF).

Box 5 - TF-IDF

TF-IDF is a weighting system often used on words. It consists of two parts: term frequency (TF), which counts how often a term occurs in a document, and inverse document frequency (IDF) which measures the term importance, as it compares how often the term occurs in a corpus[a]. The system is useful for measuring how important a term is for a document. For example, the word "the" will often occur in English texts, making it not important even though it occurs frequently in a document. However, were the term "orca" to appear a lot in one document but not in the others, this term would be weighted as more important for the document. TF-IDF has on many occasions been used in authorship attribution [68, 10, 77] and can be used on different terms, such as characters, symbols or n-grams.

a http://tfidf.com (visited April 19th)

Every year, the research group Webis hosts the PAN shared tasks on digital text forensics and stylometry[13]. Contestants will solve various tasks concerning NLP and on multiple occasions, authorship attribution was a part of the tasks. The methodologies used in these tasks are useful resources for feature and model selections.

13 https://pan.webis.de/ (visited April 18th)

In the overview paper of the authorship attribution task of PAN 2019 [43], the 12 best performing models are compiled. All features involve character n-grams and other sorts of n-grams. Other popular choices of n-grams contain words, POS (Part-of-Speech) tags, punctuation, and distortion, which is a method for masking topical contents of the text before feature extraction, focusing only on stylometric features. Most of the participants use TF-IDF weighting and SVMs as classifiers. On another note, at PAN 2018, Halvani and Graner [31] used a compression method, which is not actual feature extraction. Their approach is to compress the text (often using a method called prediction by partial matching or PPM) and then use a distance metric on these compressions to classify the author. While interesting, this project focuses on explicit feature extraction methods.

2.2.3 *Models*

Though the focus of this project is the encoding itself, a model has to be chosen for the classification and headline generation. There are different models available for performing authorship attribution. Stamatos [91] divides the models into profile-based approaches and instance-based approaches. In profile-based approaches, each author's training data is concatenated into one large text which is used to create a profile for said author. This approach has a very simple training process. However, the majority of authorship attribution approaches are instance-based which is the classic scenario where a classifier is trained on individual texts from the author.
Studies on authorship attributions are abundant and different methods are used. Koppel et al. [50, 49] used cosine similarity to attribute the closest author, space-wise and an SVM to determine whether this distance was reliable or not, only attributing good guesses. Similarly, Bozkurt et al. [12] used various models to classify 500 articles from 18 authors. They used PCA on their feature set to reduce its dimensions and obtained high-accuracy results using an SVM. Fang et al. [24] performed authorship attribution on emails with a Bi-LSTM for feature extraction, and a dense neural network for classification and Sari et al. [82] uses a simple logistic regression and a feed-forward neural network. Wu et al. [100] used a newly proposed model, a multi-channel self-attention network, which utilizes words, POS, phrase structures, and dependency relationships. Consequently, a wide range of different approaches for authorship attribution can be applied.
Authorship attribution is a well-researched field of NLP and has a wide range of use cases. The most common features for this task can be split into stylometric and content features, each with its own pros and cons. However, a more detailed categorization is given in section 2.5 where we expand on additional features previously used in NLP tasks.

2.3 NEWSPAPER ATTRIBUTION

Newspaper attribution or identification is the task of identifying from which newspaper an article originates given a list of known newspapers. Identifying the source of an article poses a relevant research problem as it can reveal significant differences between newspapers. From a socio-political perspective, finding differences between newspapers provides valuable insight. Other studies have used news corpora for opinion mining and sentiment analysis [51, 94, 11], as well as detection of political bias [53, 25, 15]. Successful detection of such differences can prompt a discussion as well as lay basis for further analysis of linguistic differences between sources. Investigating socio-political differences between newspapers is beyond the scope of this study. We instead focus on exploring which features are useful in successfully identifying newspapers.
The research on newspaper identification task is, to the best of our knowledge, sparse. Some studies, however, have created successful models for genre detection where newspaper articles were accurately classified from a number of genres [54, 93].

More closely related to our task, a study by Albakry [2] focused on comparing the frequency of five linguistic features in two different American newspapers, New York Times (NYT) and USA Today. The five linguistic features Albakry examined were prescriptive rules. In prescriptive linguistics there is a focus on defining which rules within a community represent the "correct" way of writing or speaking, e.g. not using contractions, e.g. *I'm, don't, haven't*, in formal writing or using *whom* after prepositions rather than *who*, e.g. *With **whom** do I speak?* Descriptive linguistic, on the contrary, is about merely describing how language is used within a community and formulating rules that describe the actual use of language. In his study of prescriptive usage rules, Albakry found that *NYT* and *USA Today* differed significantly from each other in three of the features: split infinitives (e.g. ***to** accurately **conclude***), starting sentences with major coordinators (e.g. ***And** in relation to...*), and clause-final prepositions (e.g. *after leaving, they came back **in***). While our study is not focused on studying usage of prescriptive rules in newspapers, the author's finding addresses notable differences between the two sampled newspapers. If these two newspapers differ significantly in these three features, it is probable that they would differ in other parameters as well and as would other newspapers. Additionally, Albakry provides important considerations when comparing sources within the same genre. *NYT* and *USA Today* were similar in popularity, and the collected articles from each newspaper were in the same topical categories to ensure reliable comparisons. These are important considerations as there can be many intervening variables. Some newspapers may lean predominantly towards one topic, e.g. finance or music, whereas

other newspapers may cover general national and international news. Another factor to take into consideration is editing. Newspaper articles may undergo editing to conform to industry conventions and possibly standards for each individual newspaper. In order to encode and analyze articles, it is worth considering which features may be particularly relevant for newspaper articles. In this regard, Maat [60] investigated conflicts between press releases and news reports. Press releases are stories submitted by individuals/organizations to journalists in hopes it will be passed on the the general public, whereas news reports are the articles published in newspapers. Maat explored which reworking strategies are used on press releases for them to fit journalistic conventions, and found that readability and neutrality are important factors in journalism. Readability is often improved by writing short and not so complex sentences, using everyday words, replacing numbers and symbols with written words, and inserting bits of background information. Features such as frequency of numbers and symbols, as well as sentence length could therefore be worthwhile for newspaper attribution.

A study by Banerjee [8] used Named Entity Recognition (NER) to compare three Hindi newspapers. The author found distinct NER variation between the three newspapers. NER could be a worthwhile method as significant differences in use of entities would further encourage a discussion of what drives linguistic differences between newspapers.

As the literature on newspaper identification problem is sparse, we aim to bridge this research gap by exploring the task using Danish newspaper articles. To the extent of our knowledge, there is no research done on newspaper identification using a Danish dataset.

Although research on newspaper attribution is limited, newspaper corpora have been used for other tasks. Several studies have used LDA on journalistic texts for topic modeling [44, 97, 37] which is the task of identifying the primary topics that a document revolves around. Topic modeling is commonly used for document classification [75, 37] and trend analysis [42, 41].

As our corpus consists of journalistic articles from news sources of varying popularity and with different target groups, topical trends and differences between newspapers can prove relevant features in attributing an article to a newspaper.

Another common text analysis strategy is opinion mining. Opinion mining, or sentiment analysis, is the process of detecting the emotional tone in text, e.g. whether an online review is positive or negative. Possible applications for this is, for example, public opinion on brands or economic predictions. One study used a Chinese news corpus to investigate opinion mining strategies [51] and applied the same strategies to a web blog corpus to compare the language use. The authors found that the news vocabulary was more varied, and the news language generally was more

objective. Studies have also investigated sentiment in news corpora in English and achieved better performance when sentiment extraction was applied to small text spans around entities rather than the whole text [7]. Godbole et al. [27] compared the sentiment around specific entities in news articles and blog posts and found that the entities discussed in news or blogs varied considerably. Additionally, the sentiment around overlapping entities was at times opposites, i.e. news articles and blog posts had opposite opinion about the same entities. These observations further encourage the consideration of entities in our analysis of newspaper attribution.
Other studies used news corpora to detect political bias [53, 25, 15]. Chen et al. analyzed how political bias is manifested linguistically and found that bias tends to be more apparent in the latter part of articles as the beginnings of articles are usually reserved for providing the reader with background information. The authors used an RNN to classify bias and reported promising results. As mentioned in subsection 2.2.3, Sari et al. did a study on authorship attribution in which they analyzed the usefulness of features across four datasets representing different genres that have been commonly used for authorship attribution. Two of the four datasets were documents of online news, one dataset consisted of legal judgments, and the last was movie reviews from IMDb. Using the Jensen-Shannon Divergence (JSD), a metric for computing similarity between two probability distributions, Sari et al. found the highest topical divergence on the online news datasets meaning that there was most dissimilarity of topics between authors in these datasets. This is no surprise as journalists often specialize within a certain area of expertise or have certain topical interests. Related to newspaper identification, the JSD could quantify the topical similarity across sources by computing the divergence between topical probability distributions. This can give insight into the dataset as well as prompt methodical considerations and manage expectations.

As mentioned, the substantial amount of online text has motivated an interest in different text analysis methods for different purposes, e.g. attending the need for plagiarism detection, and studying differing parameters of newspapers in order to discern socio-political opinions. Useful features for newspaper attribution remain largely unexplored, a research gap our study aims to bridge. Another relevant task that has become prominent with the increase and availability of text is automatic text summarization, which we present in the following section.

2.4 HEADLINE GENERATION

Generating headlines can be seen as a sub-task within the field of automatic text summarization which is the practice of generating informative, meaningful summaries containing salient aspects

of a document within a length limit [86, 69]. Within text summarization, there are two primary categories: **extraction** and **abstraction**. Extractive methods rely on extracting important sentences, based on statistical and linguistic features of the document, and concatenating them in a shorter form [30]. One caveat of this approach is, that it is incapable of generating comprehensible summaries shorter than one sentence. Additionally, the extracted sentences may be incoherent and misleading due to their disconnection from the article [39]. Abstractive text summarization relies on understanding the original text, e.g. through semantic representations, and re-telling it with fewer words [86, 30]. As our task is to generate headlines based off feature encodings, our task arguably fall under the category of abstractive text summarization. A third method in text summarization is compression which acts a step towards abstractive summarization and relies on selecting sentences and removing negligible constituents to make room for informative words that would have otherwise been dropped due to a length constraint [55].

In today's fast-growing information age, having access to short, accurate summaries of relevant information is valuable. The difficulty of the task lies in getting a model to build an understanding of the central theme of the given document and summarizing meaningful content based on that understanding. Headlines commonly bear an informative element as well as an element of persuasion in order to both grasp the main topic of the article and convince the reader to read the article. The rules that govern the structure of English front page headlines was studied by Mårdh [61] in 1980, and later in 2002 by Reah [79], who both noted that headlines serve a function to inform and evoke interest in the reader. Related to the structure of headlines, Reah notes that headline writers tend to omit function words to make room for words that are more information bearing. Additionally, the author notes that there are graphic features of headlines, however, these features are related to physical newspapers and are therefore not considered in this study. The shortness of headlines is a feature that has carried over from the switch from physical to online newspapers and is thus still a feature worth exploring. Mårdh remarks that the length of headlines has an impact on readability. It needs to be read quickly and thus needs to be relatively short but still long enough to convey the key information in a way that makes sense to the reader. As Mårdh's analysis is based on physical newspapers, it is likely that the overall structure of headlines may differ in online news, and additionally there may be other linguistic rules that govern Danish headlines. With the increase in online news, an increase in sensational headlines, commonly referred to as "clickbait", is seen. The main purpose, as the name suggests, is to bait the reader to click the articles. Clickbait headlines have been investigated by previous studies [101, 38], although the focus of this study is to explore the use of a manual encoding for headline generation. We therefore do not focus on

generating informative nor sensational headlines exclusively, but rather on evaluating headlines predicted from use of a manual encoding.

Headline generation has been studied in different languages, e.g. Russian [26], Arabic [3], and English [102, 103, 81, 18, 23, 95, 19].

A number of these studies focus on generating informative headlines [102, 19, 23] rather than eye-catching ones. Some found that extending statistical models with linguistic knowledge improved the construction of informative abstracts substantially [102, 23].

Dorr et al. [23] introduced Hedge (HEaDline GEneration) Trimmer, their linguistically motivated approach and reached a higher, albeit not significantly, BLEU [72] score than a previous system based on a Hidden Markov Model (HMM) approach. Despite the lacking significance in evaluation, authors note that with minimal linguistic knowledge, they were able to circumvent the need for large amounts of training data required by purely statistical approaches. The authors introduced linguistic knowledge by parsing the lead sentence of an article, which produced a parse tree that was then trimmed in accordance with a linguistically motivated algorithm. The algorithm was developed based on observations of human-produced headlines and would, for example, remove low content units, e.g. some determiners (e.g. *a, an, the*) and time expressions (e.g. *now, after, then*).

Zajic et al. [103] investigated Hedge Trimmer's sentence compression combined with topics, introducing the approach, Topiary. They use Unsupervised Topic Discovery for a corpus that does not have topic annotations and generated headlines by removing constituents of a parse tree of the lead sentence to fit a length threshold. The authors used ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [56] and BLEU scores to evaluate the combined approach. ROUGE is a metric, based on recall, for evaluating summaries. For ROUGE-N, the metric counts the number of overlapping n-grams between candidate and reference summaries and divides by the total number of n-grams in the reference summaries. ROUGE metrics are elaborated on in subsection 4.8.5. The authors found that Topiary scored significantly higher on ROUGE-1 (unigrams) compared to the Trimmer and the same finding applied for the BLEU-1. ROUGE scores has been used in numerous other studies on headline generation [26, 19, 3] and text summarization [18, 81], as has BLEU [102, 96]. Dorr et al. and Zajic et al. (2004) both incorporated linguistic knowledge to address the large amount of training data needed by purely statistical approaches.

Purely statistical approaches include hidden Markov chains, RNNs, and encoders-decoder models that do not make any linguistically motivated modifications to the output.

One study by Rush et al. [81] implemented a neural attention model for abstractive sentence summarization called Attention-Based Summarization (ABS). Additionally, they implemented

ABS+ where some extractive features were incorporated, although much less so than in Hedge Trimmer and Topiary. The authors found they were able to significantly outperform Topiary using the ROUGE metric. An extension of ABS was introduced by Chopra et al. [18] who used a convolutional attention-based conditional RNN including positional information of words and found it outperformed both ABS and ABS+ comfortably.

Another study using an RNN by Lopyrev [59] used LSTM units and attention mechanisms to generate headlines from news articles. The author found that the model performed well on some of the used newspapers but not on others. This finding was attributed to the fact that articles from the newspapers in question differed considerably compared to the training data. This relates to our considerations on newspaper attribution and provides further motivation for discerning which parameters newspapers might differ on.

Several statistical and combined approaches have been investigated for headline generation and a number of studies have achieved promising results, some reaching SOA performance. Some of the studies [23, 102] found that a combined approach, i.e. combining a statistical approach with some hand-picked features, performed better than the purely statistical approach. Importantly, however, the linguistically motivated modifications of these approaches were not feature extraction from texts as has been done for authorship and newspaper attribution. Instead, linguistically motivated algorithms removed specific parts of sentences, words and phrases which are commonly omitted in headlines.

With this in mind, it is worthwhile to explore whether a handcrafted text encoding can be utilized for headline generation.

2.5 FEATURES

In the previous sections we have gone over some of the previous studies relating to authorship attribution, newspaper attribution, and headline generation. In this section, we summarize and expand on the features mentioned and tie their relevance to our study.

Linguistic features have been widely considered for a variety of NLP tasks and can be divided into some overarching categories. Features can, for example, be considered lexical, character-based, syntactical, or content-based.

Note that this categorization is rather loose as there is a lot of overlap in terms of which aspects of a text a feature adds information to.

2.5.1 *Lexical Features*

Lexical features pertain to features related to the vocabulary of a language or its words, as opposed to the structural aspects of a language. Word frequencies is a commonly considered feature as it is relatively easily implemented across languages and a weighted distribution of words can shed light on the vocabulary used in a text. Previous studies [4, 14, 67] have found that the most common words, function words, are a particularly good stylistic indicator for authorship and have therefore been used in several studies pertaining to authorship tasks. For one, Koppel et al. [47] found that function words were the most varied feature between authors, i.e. differed most. The authors attributed this finding to different stylistic preferences and thus regarded function words a distinctive feature for identifying authorship. Function words are therefore a highly relevant feature for this study. As opposed to the most common words, hapax legomena (words occurring only once) have also been considered as stylistic markers [20] for authors in that counts of hapaxes can be suggestive of an author's vocabulary. It is worth noting that hapaxes overlap with content features because unique words in a text can be indicative of topics which ties in well with newspaper attribution as some domains are more topic specific than others.

To extend the knowledge of the vocabulary, measures for lexical diversity is also often considered. Lexical diversity or richness (or vocabulary richness) measures give a scalar value relating to the diversity in the vocabulary. These measures are often dependent on text length [91, 62] which is a caveat to consider when using lexical richness features. It is therefore not recommended to use only these features to represent texts.

Another way to represent words is through word embeddings in which words are represented as vectors in an embedding space. In this embedding space, words of similar semantic content are grouped together, e.g. *ostrich* and *emu* has similar word embeddings. Notable studies in this regard are ones by Mikolov et al. [65, 64] who introduced Word2Vec, a neural network that learns word associations. A commonly used architecture for learning word associations is continuous skipgrams that predicts the surrounding context of an input word.

Weighted word frequencies, function words, hapaxes, and lexical diversity measures are simple and easily implemented features, however, no contextual information is considered. To add contextual information, word n-grams can be beneficial which we will elaborate on in subsection 2.5.3.

2.5.2 *Syntactical Features*

Syntax concerns the way in which elements of the language are assembled into constituents. Where lexical features relate to words and vocabulary, syntactical features relate to the way in which words are structured in relation to each other. In order to capture syntactical information, POS trigrams have been considered in text categorization of magazines and newspapers [5] which is highly relevant to our newspaper attribution problem. POS tags are grammatical tags assigned to tokens (words), e.g. nouns, verb, adverb, etc. In a study by Koppel & Schler [48], they found POS bigrams to be more useful than other n-grams in attributing authorship.

Not only has POS bigrams been useful but skipgrams of POS tags have been found to yield promising results in discriminating between authors of blog posts [74]. They have also previously been used by Argamon et al. [5] to distinguish magazine articles from newspaper articles. They found that POS trigrams were a distinctive feature for news attribution. For authorship attribution, POS bigrams have likewise been beneficial in uncovering stylistic preferences between authors [48] Skipgrams, like regular n-grams, are collocations of elements, however, in computing skipgrams, a skip distance, k, is defined which indicates the amount of words (or characters or POS tags) to skip over. Authors also found that selecting the 250 most frequent POS patterns gave best results which is testament to the fact that a small number of patterns can still characterize authors well.

2.5.3 *Content Features*

Content features concern the parts of text that has to do with the theme or topic. As mentioned in subsection 2.5.1, word n-grams add contextual information on a lexical level. These word collocations allow for the capture of both contextual, content-based, and syntactical information. Homonyms (words spelled and pronounced the same but different in meaning) will appear in different contexts which word bigrams would capture, e.g. ***to*** *count* vs. ***the*** *count*, thereby adding contextual information as well as syntactical in that it indicates the lexical class of the following word. Hapaxes were also mentioned, and it is important to note that these also offer information about content because unique words can be allusive to topic. Content features can in some cases bias a classification because authors may not always write about the same topics. Sari et al. [82] analyzed the usefulness of style and content features respectively and found that stylistic features are more effective when authors write about similar topics, whereas content features were effective

for datasets where there was more dis-similarity in topics between authors. Seroussi et al. [85] also achieved results that indicated topics were a useful feature for attributing authorship.
Another content related feature is extraction of named entities. Named entity recognition is the process of identifying which words or constituents in a phrase make up an entity, e.g. person, location, or company. Extracting entities can aid in uncovering topics but a mere count of entities in a text can also be considered a stylistic feature in that a higher or lower number of named entities can allude to the source of a text.

2.5.4 *Character Features*

Character-based features relate to measures that are concerned with considering text from a character-level perspective. For this, character n-grams have been considered as they can capture lexical information and contextual information, and are additionally tolerant to noise, e.g. *cheetah* vs *cheetha* would yield the trigrams | *ah_* |[14] and | *ha_* | respectively. Were these two words represented lexically, they would yield two different representations, whereas with the character-based n-grams, the two words would have several common character n-grams (| *_ch* |,| *che* |, | *hee* |, | *eet* |). As character n-grams have been used in multiple previous studies [106, 92, 73] to discern authorship it is a relevant feature to consider in light of our authorship task.

The features described in this section is not an exhaustive list of features that are used in classification of authors and newspapers but merely provide an overview of commonly used features as well as features used in the aforementioned studies. In the next chapter we reiterate the focus of our project, and describe our methodological approaches, hereunder an elaboration on the features we decided to include in our study.

14 _ marks a space, i.e. the end or beginning of a word.

Summary 2 - Related Works

Throughout the literature review, four essential topics were covered: automatic feature extraction, authorship attribution, newspaper attribution, and headline generation. Not much literature was found on newspaper attribution, so papers from related fields were used to gain an understanding of the SOA. The usefulness of NLP features is task-dependent and the final choice of features will be determined in section 3.6 based on the findings of this chapter.

The key points from the literature review are:

- Autoencoders can automatically generate feature vectors of text (section 2.1).
- The LSTM autoencoder and transformer has proven useful for sequential data, herein text input (subsection 2.1.1).
- The features used for authorship attribution are split into stylometric and content features, where both are proven useful for the newspaper genre (subsection 2.2.2).
- Newspapers have been found to vary significantly with respect to some prescriptive usage rules (section 2.3).
- Journalistic articles undergo editing to improve readability, e.g. use of everyday words, replacing numbers and symbols, and more (section 2.3).
- News corpora has been used for different NLP tasks such as topic modelling, opinion mining, and authorship attribution. Studies show that news corpora have a tendency for topical and linguistic variation (section 2.3).
- Headline generation is a sub-task of text summarization of which there are three predominant methods: extractive, abstractive, and compressive (section 2.4).
- Incorporation of linguistic knowledge for text summarization is found to yield better results and can bypass the need for large amounts of training data (section 2.4).

3

METHODOLOGY

In this chapter we will first reiterate the focus of our project in section 3.1 as well as establish our expectations for the results. We then describe which hardware we used for computation in section 3.2, the data used in the project in section 3.3 and how we pre-processed it before extracting features in section 3.4. We will investigate the distributions of data in section 3.5, and present which features we chose to extract and why in section 3.6. We will argue for the decision of creating multiple subsets of the data in section 3.7, present the models used in section 3.8 and finally, failed attempts and design flaws in section 3.9.

3.1 FOCUS

The primary aim of this thesis is to explore the use of a general-purpose encoding of Danish newspaper articles and applying the same encoding on authorship attribution, newspaper attribution, and headline generation. Authorship and newspaper attribution are classification problems whereas headline generation is a generation problem. To the best of our knowledge, newspaper attribution is very sparsely documented, and this thesis therefore contributes with information on which features can aid source attribution within the same genre as well as which features may overlap with an authorship attribution task in the newspaper genre. Headline generation can be seen as a sub task of text summarization which is an active field of research. Several studies utilizing transformers for machine translation [98] and other NLP tasks [21] have been done to illustrate the versatility of transformers. A novelty aspect in this project is to use commonly used text features as input for the headline generation, which, to the best of our knowledge, has not been attempted before. For all tasks we use a novel Danish dataset which is described briefly in section 3.3. Furthermore, we focus on comparing our manually extracted features against automatic encodings from a transformer.

As our focus lies on the encoding, we use established classification models for the classification tasks such as random forest and logistic regression. Another option would be to implement a neural network structure, but it is a laborious process to find the right hyperparameters and training time is often longer than other types of models and they require large quantities of training data. For headline generation, we will utilize a ready-made model as well. By using a pre-existing model, we can ensure that the model indeed works. If our results are not as expected, we can rule out the model being faulty, and the issue must lie within our encoding or data of choice. The exact models used will be elaborated upon in section 3.8.

In terms of feature extraction, we focus on interpretability rather than optimization and have thus limited the time spent on feature and model optimization, although we discuss optimization possibilities briefly in section 6.3.

The headlines produced by the headline generator will be manually selected for showcasing when presenting the results, as this practice is common in headline generation papers [59]. One can interpret this as the task being difficult, i.e. only successes are shown, as the overall expression of the results is poor. However, for the sake of transparency and proper academic contribution, we will present the best and the worst results based on a qualitative analysis of the headlines.

3.1.1 *Expectations*

We have various expectations regarding the results obtained by the models and which traits we can interpret from them. Our expectations are focused around the main research question, *How can we create a general-purpose feature encoding for authorship attribution, newspaper attribution, and headline generation?* and its sub-questions:

1. *Will manually extracted features, automatically extracted features, or a combination of both generalize best?*
2. *Which of the manually extracted features has the highest impact on the results?*
3. *How do the newspapers relate to each other? Does our feature representation reflect real world similarities between newspapers?*
4. *How will a manual encoding perform for headline generation?*

In regard to our main research question, we expect the classification tasks to return more convincing results than the headline generation. Although these two tasks are not directly comparable, we expect the classification tasks to outperform the baseline model and we expect the majority of generated headlines to be nonsensical. This is due to several factors: authorship attribution is a

much more established field than headline generation and validated models are readily available. Furthermore, the metrics used for classification may not necessarily be applicable to generation, which makes the comparison between these two tasks more dependent on qualitative evaluation.

With respect to which encoding types will perform better, we expect the performance of all models to increase when combining manually and automatically extracted features as seen in the literature [81].

Regarding features, we expect that topic features are among the most important features for both classification tasks and we expect the vector features to outperform scalar features, due to their popular usage in the literature. Although many previous studies used English for development, we expect a fair amount of the features will perform well for Danish as well as the two languages belong to the same language family. Furthermore, a multitude of the features presented so far are language-resilient, e.g. character n-grams or sentence length.

We do not expect feature importances to be applicable for the headline generation, due to different factors: firstly, we would need to train it several times, and perform ablation studies by omitting various features to see how it affects the results, alternatively permutation importance could be used to speed up this process, however we did not perform such process due to time constraints. Secondly, this requires good results in the first place, i.e. having bad results and removing a feature will most likely also give bad results. Due to excessive training times of the headline generator, this approach is not feasible, even if the results with all features seem good.

In terms of how we expect the newspapers to relate to each other, it is worth noting that our data comprises of domains that are tabloids, mainstream or local news, or niche topic news (see subsection 3.3.1), e.g. IT, finance, or music. For this reason, we expect that tabloids will likely resemble each other in a feature space, as will mainstream news media outlets. With respect to the niche newspapers, we expect that papers covering similar topics will be alike each other in their topic features and might then be difficult for a model to discriminate between.

To the best of our knowledge, there have been no attempts at generating headlines from features extracted in a purely manual manor, which we are attempting in this study. The lack of studies attempting this makes it worthwhile exploring the possibilities, though we do not expect the results to be overwhelmingly good.

3.2 HARDWARE

Multiple systems were used for extracting features and training models. The cloud computing platform, UCloud[15] was utilized for some computationally heavy tasks and training times stated in this report are achieved with one of the systems stated in Table 1.

Name	Processor	Base Clock Speed	RAM
PC 1	Intel Core i5-4200U	1.6GHz	16GB
PC 2	Intel Core i5-8300H	2.3GHz	16GB
PC 3	Intel Core i7-4771	3.5GHz	16GB
PC 4	Intel Core i7-7700K	4.2GHz	16GB
Cloud 1	Intel Xeon Gold 6130	2.1GHz	$\leq$ 384GB

Table 1: The specifications of the hardware used for computation.

Due to the PC systems' comparable specifications, we will only distinguish between cloud and PC processes, as the cloud resources utilize a different series of processors from the personal computers.

3.3 THE DATA

The data used for this project is a product of the Danish company, Copyfighter, which uses it for plagiarism detection. The raw data consists of *808,066* Danish articles by *66,994* different authors across *268* domains. The meta data available for each entry are: unique ID, the domain, article body, article header, the publication date, the URI (web link) to the article, the author, and a text hash. The publication date, ID and text hash are unused except for managing and sorting data. As the data consists of many different domains, we establish four main categories that the newspapers fall into.

3.3.1 *Types of Newspapers*

There are a multitude of different types of newspapers, depending on how one divides them. In this section, we focus on four types of newspapers: **mainstream**, **local news**, **tabloid**, and **niche**. All

15 https://cloud.sdu.dk/ (visited May 6th)

four types are present in our database, and we expect them to differ in content and style of writing. We also expect each type of newspaper to be more confused for each other than across types of newspapers.

Mainstream newspapers covers a broad range of topics and type of articles, e.g. news, investigative articles, analyses, reviews, etc. [89] Examples of domains in our database belonging to this group are Berlingske, Politiken[16], and Jyllands-Posten.

Tabloid newspapers focus on sensations and entertainment, including scandals, gossip, and sport. The name stems from the smaller physical size of the newspaper [90] and domains of this type include BT and Ekstra Bladet[17]. From the nature of these types of newspapers, one would also expect an increased frequency of clickbait headlines.

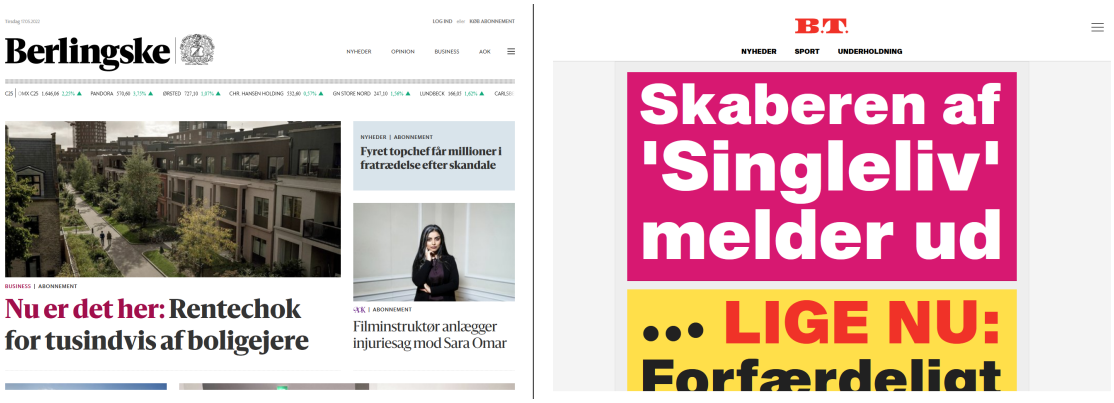


Figure 3: The mainstream newspaper Berlingske (left) and its tabloid counterpart, BT (left) (visited May 17th).

Though Berlingske and BT (Berlingske Tidene) are related, there is a clear visual difference between the two, as seen in Figure 3.

Local news is a type of newspaper, that unsurprisingly focuses on local news and events. This type of newspaper is related to its geographical area of coverage and may be seen as a sub-type of mainstream newspapers. Such newspapers in our dataset include SN[18], Fyens[19], and Nordjyske[20].

Niche newspapers cover a focused set of topics, e.g. culture, finance, or technology, and thereby have a narrower target group than the mainstream newspapers. Examples of such newspapers in our data are Information, Børsen and Computerworld.

16 https://politiken.dk/
17 https://ekstrabladet.dk/
18 https://www.sn.dk/
19 https://fyens.dk/
20 https://nordjyske.dk/

3.4 DATA PRE-PROCESSING

As the database is real-world data, it is bound to have imperfections. Some entries have missing fields, and some are malformed or duplicated. In this section, we describe the process of cleaning the data before extracting features and applying it to any learning models. In the running text, exact numbers of articles, domains and authors will be omitted or rounded to ease the reading process. Please refer to Table 2 to see the exact numbers and amounts of data affected by the pre-processing steps. Each of the subsections correspond to a row in said table.

3.4.1 *Removing Duplicates, Missing and Malformed Entries*

The data had a large quantity of duplicated bodies which were removed. Afterwards, entries with missing data were discarded. If an entry did not have either a body, author, header, or domain it was discarded, as these fields are necessary for the learning. We contemplated saving the entries with missing data that had an intact URI and then scraping the body, header, or author from the website, depending on which fields were missing. However, we deemed this too time consuming relative to how necessary it was, as plenty data was left after cleaning up missing entries. On multiple accounts, anomalies were found which were removed. Early entries in the data would have incorrect values in the author field, e.g. an URI instead of author or test data from the developers, like *"testtest"* and *"awdad"*. After removing these, $\approx$*7,800* unique authors remained.
The author fields were titleized, i.e. making every word start with a capital letter. The reason for this being that the capitalization of the authors' names were not consistent. In one instance the author would be *bjørn lomborg* and in other cases *Bjørn Lomborg* which would be identified as two separate authors.

3.4.2 *Removing Entries with Multiple Authors*

Some entries had more than one author. Using these entries unaltered would have repercussions in the authorship classification: firstly, it is unknown how large a part each author had contributed with and with what exactly they contributed. Secondly, some authors appear in the database alone in some entries but also with a co-author in other entries, which would introduce additional noise to the classification. Finally, a lot of the combinations of authors appear only once, as some authors may be guest authors.

We considered possible solutions to this issue as we wanted to keep the focus on single authors, and because detecting style change of authors in a single document is another task in itself [104]. One solution would be to remove guest authors from articles with multiple authors, i.e. authors not occurring by themselves anywhere in the database. However, only 68 authors qualified as guest authors, so we decided to discard the entries with more than one author as it was affordable in terms of lost data.

Most entries with multiple authors were formatted with `,` or `|` as separators. However, it is not as simple as splitting the strings by these characters and counting the authors, removing them if more than one is present. Some authors would have cities or job titles included in their fields, e.g. *Anette Claudi, redaktionschef for forbrug* or *Lone Theils, London*. As these entries should not be removed, the actual author name should be isolated and used in the author field. Doing this would identify the author Lone Theils even though the field could contain both *Lone Theils, London* or *Lone Theils, Warszawa*. Both articles should be attributed to Lone Theils.

Some rules were set up to filter entries with multiple authors. Firstly, entries with `|` as separators could safely be removed, as they were only used with multiple authors. Secondly, the author field was split by the comma separator and if the second name only contained one name (i.e. no space) we can safely assume it is a city or job and not a person. Lastly, to capture entries with multiple words for job titles or cities, a handcrafted list of words were used for filtering. These words include *redaktør, York, Ritzau, Universitet, etc.*

Before pre-processing, the database had ≈800K entries with ≈67K unique authors. After the pre-processing, this was reduced to ≈330K entries with ≈3K unique authors. Exact numbers are seen in Table 2.

3.4.3 *Removing Entries with Domain/Author Occurring Only Once*

In terms of domain names and authors, each entry needs to occur at least once in the training set and once in the test set. Were an author to be present in the test set and not the training set, the model would not have learned this author's writing style. With a class occurring only once, it is not possible for a model to both train and test on this class. Therefore, non-repeating entries in these fields were removed.

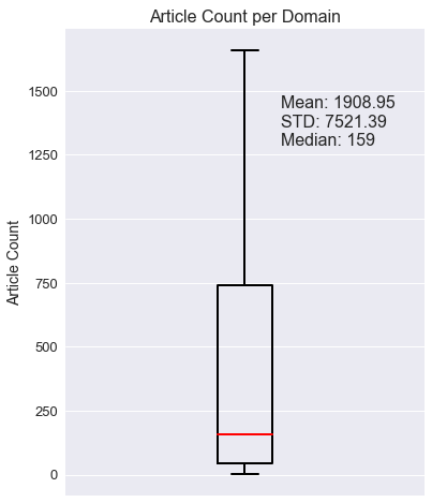


Figure 4: Article count per domain boxplot. Outliers removed for readability.

The database originally had 207 unique domain names, but 11 occurred only once, leaving 196 domains. After removing unique entries, the distribution seen in Figure 4 was obtained.
There are ≈*1,200* authors that only occur once in the database. A multitude of these authors are anomalies, e.g. not authors (incorrectly formatted cells), or multiple authors with uncommon separators. The authors were once again titleized, as the multiple authors cleaning step left some entries with capitalized author names. Performing these operations reduced the number of unique authors to ≈*1,900*, while converting *34* authors with titleize. This operation is also indicated in Table 2.

3.4.4 *Removing Entries with Foreign Domain and Duplicate Headers*

An issue with the headers was unearthed. ≈*29K* entries had headlines, that were not unique. In some cases, it was justified, e.g. *Briefing Morgen: Her er dit overblik på 2 minutter* which is a headline concerning brief daily news. By investigating the different types of articles associated with the headers, a threshold of 20 occurrences were set, i.e. any articles with headlines occurring more than 20 times were dropped, removing ≈*1500* articles. Note, that while this operation is still applied to the data used for classification tasks, any duplicate headlines for headline generation were removed.
Additionally, non-Danish articles were present in the database. These were removed by domain endings, which worked with few issues, e.g. Danish and Swedish domains both using the extension *.nu*. In total, *31* domains were removed with ≈*28K* entries and *39* authors.
However, ≈15K articles still had the same header, *Din profil er oprettet* ("Your profile has been created"). These headers had identical bodies and were mainly found in the domains BT and Berlingske. It was found, that these articles were incorrectly saved in the database and made up the entirety of the articles from the domain Berlingske which is a prominent Danish newspaper.
Along with this issue, it was found, that some authors had written thousands of articles. Investigating this, revealed many cases where these authors were incorrectly attributed. The author Anne Sophia Hermansen had apparently written ≈*14K* articles and Thomas Treo ≈*6K* with the median author having written only 3 articles. The true authors of miscredited articles were found by random sampling and by finding article counts on online author profile pages if available. It is probable that multiple cases of miscredited articles are present in the database which were not discovered. However, investigating all authors and domains, would be incredibly laborious, so we settled on the

top 10 authors according to articles written. In total ≈35K articles had to be reevaluated which we did with a web scraper.
Some authors, such as Karim Pedersen and Mads Elkær, had also written thousands of articles which seemed suspicious. However, it was revealed that Mads Elkær is one of the founders of Copyfighter and Karim Pedersen has been in charge of testing the system. As they have used the Copyfighter system a lot, their articles naturally have a higher chance of occurring in the dataset.

3.4.4.1 *Web Scraping*

With the URIs available for the malformed articles, we made a web scraper to fetch the header, body and author and replace the missing data. This was only done on the domains Berlingske, BT, and Ekstra Bladet as they were the domains where issues occurred. As mentioned in section 3.4, data was discarded if entries were missing instead of being scraped. The reason for not scraping these articles, is the sheer number of articles across many domains. This would take excessive time to implement, as every domain has one or more different structures which the scraper should be tailored to. The implementation of this process is described in section 4.2.
After scraping, ≈*5K* articles were manually removed as they still had the header, "Din profil er oprettet". These were not automatically filtered because of timing out. *40* authors were added, due to the incorrectly attributed authors being fixed. However, this shrank to *18* after removing unique authors once again.

Process	**Entries**	**Authors**	**Removed Entries**	**Removed Authors**
Load the data	808,066	66,994	-	-
Removing duplicates, missing and malformed entries	348,751	7,849	459,315	59,145
Removing entries with multiple authors	336,788	3,103	11,963	4,746
Removing entries with domain/ author occurring only once	335,583	1,906	1,250	1,197
Removing entries with foreign domain and duplicate headers	307,953	1,867	27,630	39
Scraping articles	301,416	1,885	3,995	+ *18*

Table 2: Quantities of data removed with each process of pre-processing. Please note that 18 authors were added after scraping.

3.4.5 *Data Leakage*

Data leakage occurs when the classification model has access to revealing data which was not accounted for. For example, having a model predict housing prices based on various metrics, such as size, year built, location and price per square meter. The target value is the price, but the price per square meter is also included and not hidden from the model when training. This of course gives the answer away to the model, resulting in overly ambitious result metrics.

In the case of this project, data leakage could be if the author or domain persist in the article body. If that is the case, predicting either of these two target values would be trivial.

Therefore, we used regular expressions to remove the author and newspaper names from both headlines and article bodies. Along with this, we searched domains for commonly occurring text, such as the domain DR[21] writing *Mere end 30 dage gammel* (more than 30 days old) if the article is not new. Such phrases were detected by comparing each article in a subsample from the domain in question to a randomly selected article from the same domain. The three longest matches are found using the built-in Python library, `difflib`, and all matches occurring more than 3 times with a length above 20 characters are removed from the articles. Had these texts not been removed, the

21 dr.dk

model would more likely than not associate the phrase *Mere end 30 dage gammel* with DR instead of generalizing on the actual content of the articles. Such strings will henceforth be referred to as junk.

Figure 5: A word cloud of the most common words, which were removed from the article bodies across all domains. Some domain names, were recognized as junk, such as Altinget[22] and Gaffa[23]

In Figure 5, the most common words removed across all domains are presented. Note, that this accounts for the most common words, not entire strings. For instance, the article bodies from the sports magazine Bold[24] were littered with code strings, such as JavaScript and CSS which the automatic junk removal did not capture sufficiently, so we decided to exclude the domain all together.

3.4.6 *Limitations*

As described above, we took several steps to pre-process and clean the data to avoid data leakage. Despite the many steps we took in an attempt to remove junk and avoid bias, there were still oddities to be found within the bodies that could potentially bias the classification. Due to the sheer size of the dataset, we were not able to account for every occurrence of repeating patterns. For example, some domains were behind a paywall or had sponsored content. Remnants thereof was thus found in

24 bold.dk

different variations throughout domains, e.g. *Køb abonnement*, *Læs mere*, *Annoncørbetalt indhold*, and *Få Nyhedsbreve*. Considering time constraints, we decided to not hard code rules for all the recurring patterns we found within the different domains and instead treat results with account and reservations for these potential biases which will be further discussed in section 6.1.

The text cleaning process was applied twice, to account for any missed parts in the first iteration, as the process is partially based on randomness. After searching for and removing authors, domains, and junk from articles twice and manually adding strings for removal, $\approx$*550k* substrings were removed.

After the final pre-processing, the amount of data left across articles, authors and domains are seen in Table 3.

	Count
Articles	301,416
Authors	1,885
Domains	160

Table 3: The final amount of data

3.5 DATA INSIGHTS

Besides getting insights in the raw data while cleaning it up, we also investigated the distributions and possible correlations in the database.

3.5.1 *Article Lengths*

First, we investigated the lengths of the article bodies to see if some articles were outliers in terms of length, e.g. some articles containing only a single word. The distribution of article lengths is seen in Figure 6 which has a smooth curve from start to finish. Note, that the unprocessed article count is scaled to have a similar area under the curve as the pre-processed data as a large quantity of data was removed, i.e. the black line would lie much higher on the y-axis. The scaling makes it easier to compare the distribution.

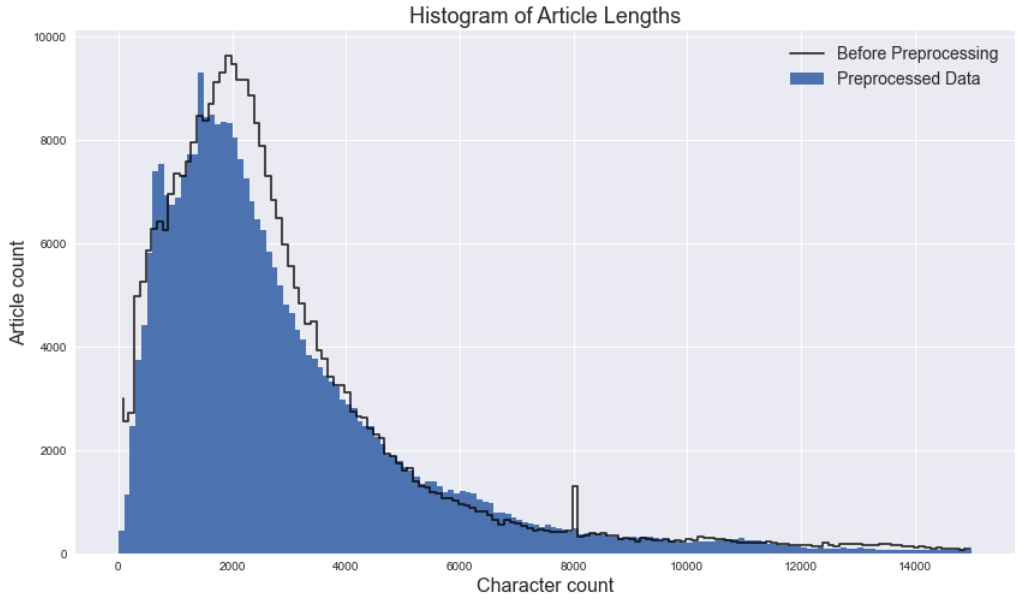


Figure 6: A histogram of the distribution of article body lengths. The pre-processed data is indicated by the blue histogram and the raw unprocessed data is indicated by the black line.

The left tail has changed due to removing empty bodies and a peak around 8000 characters count is removed due to the removal of duplicated articles. After pre-processing the distribution is as seen in Figure 7. The average article lengths per domain and author are available in Appendix A.

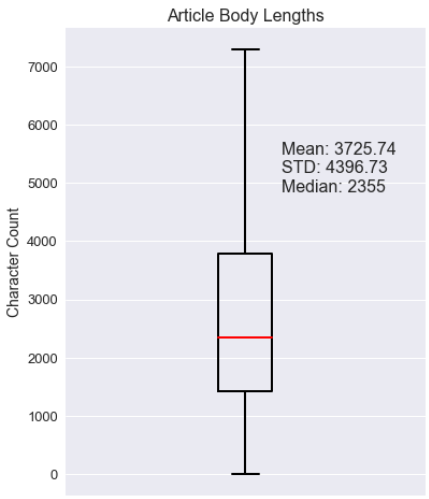


Figure 7: Boxplot showing article lengths in characters. Outliers removed for readability.

3.5.2 *Publication Dates*

To get an idea of when the articles were published, the distribution of publication dates are seen in Figure 8. The writing styles and subjects change over time, so an article from 1920 would differ from an online article from 2021.

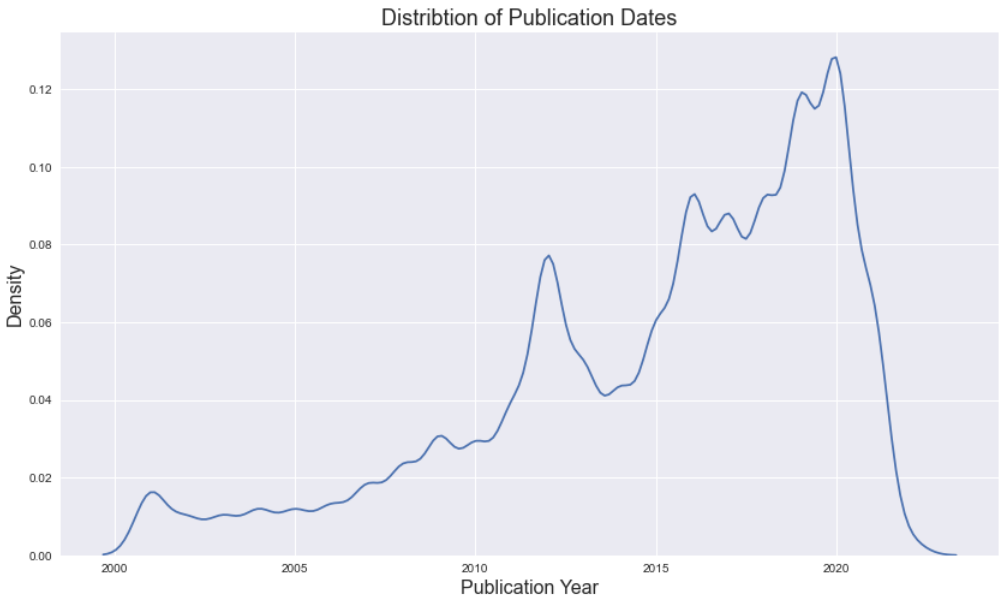


Figure 8: The distribution of publication dates between 1995 and 2022.

The publication date field of our dataset is, unexpectedly, imperfect. The figure is limited to the years 1995-2022, as they are gathered in the age of the internet. However, ≈5k articles were marked as being created after 2022, which of course is not possible. This is simply due to an inconsistency in how the publication data is formatted. The articles from the future are not included in Figure 8. As expected, the number of online articles grows over the years, as more and more articles get digitalized. However, keep in mind that this distribution may also, in part, reflect how frequently used the Copyfighter tool was, as journalists have access to the system themselves and can add their articles for plagiarism detection.

To see whether the average headline length decreased along with the publication date increasing, we plotted them against each other. This was done to see, whether headlines for new articles would be shorter than old articles, i.e. less descriptive headline and more along the lines of clickbait. However, the distribution seemed to a very similar structure as the publication dates seen in Figure 8, i.e. headlines actually increasing in length over time. The plot is seen in appendix section A.3. This finding will be further discussed in subsection 6.1.4.

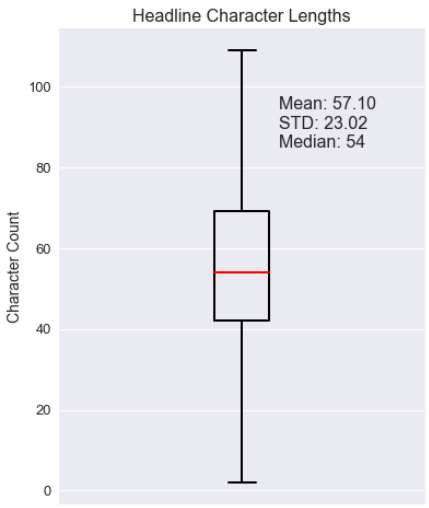


Figure 9: Headline lengths boxplot. Outliers removed for readability.

3.5.3 *Headlines*

The distributions of headline lengths were investigated as well. The mean character count, along with standard deviation and median is seen in Figure 9, and in Figure 10, the distribution of the headlines is seen along with a (not fitted) normal distribution on top of it. According to an Anderson-Darling test the data is normally distributed passing a significance a level of 0.01.

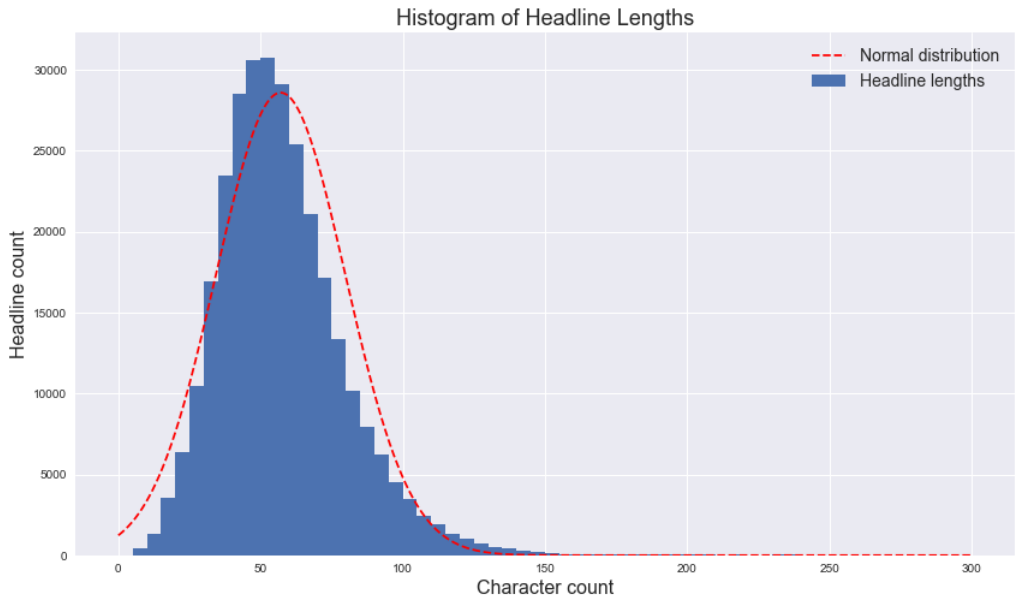


Figure 10: The distribution of headline lengths in character with a non-fitted normal distribution on top.

3.5.4 *Authors*

Secondly, we investigated the authors, namely the article counts for each author. We found that the database still had some notable outliers in terms of article counts for each author, as highlighted in Figure 11. Even though the articles by Anne Sophia Hermansen were revised, she has still written ≈7.5K articles.

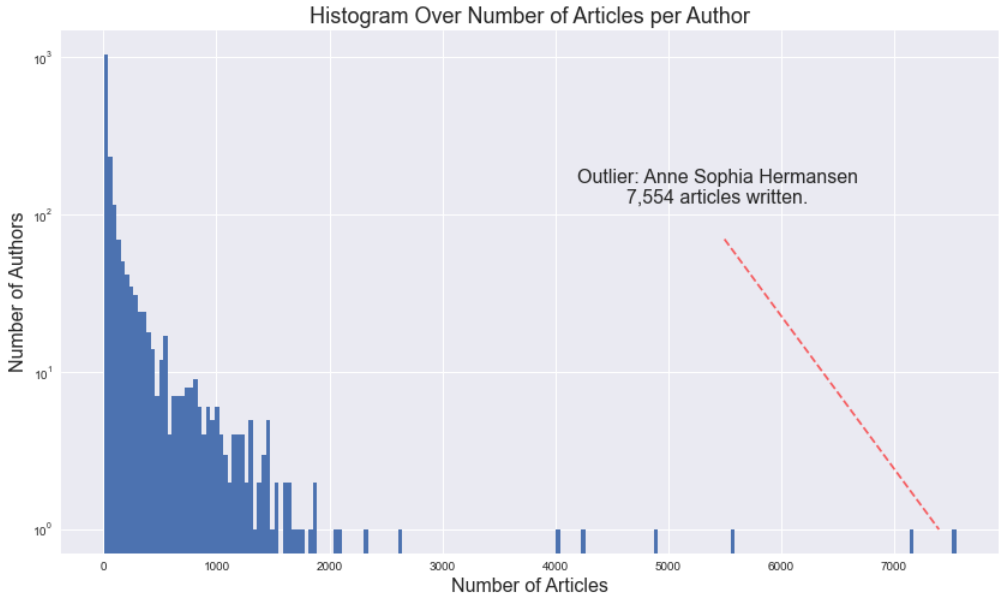


Figure 11: A histogram showing how many authors have written a certain number of articles. The bars at height 10^0 indicate individual authors with their article count on the x axis, highlighting outliers. Note, the y axis is logarithmic, hence the low volume compared to the previous plots.

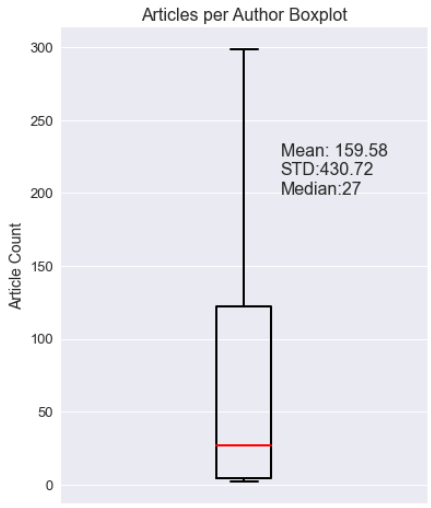


Figure 12: Articles per author boxplot with outliers removed for readability.

The mean, standard deviation and median along with the boxplot of the distribution is seen in Figure 12.

Finally, we investigated average length of articles for each author and whether this had a correlation between number of articles produced, i.e. having written a large quantity of articles, they would likely have a short length. Applying Pearson's correlation coefficient between the average article length and number of articles written for each author returned no correlation. The coefficient is *-0.012* with a p-value of *0.57*. Interpreting this, it means that we can confidently say, that there is no linear correlation between average article length and article count. Figures showcasing this step are available in Appendix A.

3.6 FEATURE SELECTION

Based on the literature covered in chapter 2, we here describe which features will be used for the manual encoding. The features are selected on the basis of authorship attribution as this task has

been documented over many years and with a wide variety of features. A few features documented as successful for newspaper attribution in particular also lay ground for our selection. We expect there will be a reasonable amount of overlap in which features are useful for the respective classification task. As headline generation has garnered attention with respect to an automatic approach, there were not any easily discernible features to extract. The linguistic knowledge of these approaches comprise of linguistic algorithms rather than extracted features. We therefore focused our selection around authorship and newspaper attribution. To the extent of our knowledge, there is no literature on headline generation using an extracted feature set, such as found in authorship attribution tasks. Consequently, no features are specifically selected for headline generation.
In Table 4, all feature types are presented.

Feature name	**Feature type**	**Amount of features**
Sentence statistics (count, lengths in characters and words)	Scalar	7
Word statistics (count, lengths)	Scalar	4
Hapax legomena count	Scalar	1
Lexical diversity	Scalar	8
Topic scores	Scalar	22
Word n-grams	Vector	3
Character n-grams	Vector	3
POS n-grams	Vector	3
Bi-skipgrams	Vector	3
Function word frequency	Vector	1

Table 4: The manual features chosen for each task. Each feature type may have different variations, with the amount showcased in the rightmost column.

The feature types, scalar and vector refer to single values and vectorized values, respectively. Each document is represented by all the features showcased in the table. As written in section 2.1, the feature vectors produced by autoencoders will be smaller in size than the actual text. This is not necessarily the case with manually extracted features, as using various n-grams will blow up the data size. However, this is not a concern, as generalization over compression is the focus of this project. Please note, that topic scores can be treated as both a vector (with length 22) or 22 individual

scalars. The reason for categorizing them as scalars are, that each topic is more easily distinguished than the other vector features. Topic scores will be expanded upon in subsubsection 3.6.1.8.
The features described in this section are initial, experimental features. Each feature type has multiple components, which may be reduced for the final implementation, choosing only the best performing features and also reduce the feature space with principal component analysis (PCA).

3.6.1 *Feature Motivation*

In this section, we present the reasoning behind the selection of features for our encoding, whereas the actual implementation of the feature extraction will follow in section 4.3.

3.6.1.1 *Sentence Statistics*

We decided to include sentence statistics in the form of sentence counts and lengths (in words and characters) as such features have commonly been used in authorship attribution tasks, and we presume they would beneficial for newspaper attribution as well.

3.6.1.2 *Word Statistics*

Alike the sentence statistics, we also use count, mean, median, and standard deviation for words. Word count can also be a measurement of general text length, and word length might give preliminary insight into word complexity. As noted by Stamatos [91], word and sentence measures are features that are more of less easily implemented across languages and is therefore a rather scalable feature.

3.6.1.3 *Hapax Legomena*

As mentioned in subsection 2.5.1, hapax legomena are words that occur only once within a text. Our reason for including it as a feature is that some domains may have a predominant focus on specific topics which could be useful in identifying the domain. Note, that the feature is a count of hapax legomena, not the actual words chosen.

3.6.1.4 *Lexical Richness*

Lexical richness or vocabulary diversity are measures of the range of words used in texts where a greater range indicates more diversity [62]. Although Stamatos [91] states that lexical richness measures are not reliable on their own, a combination of many such measures may be. Measuring

the range of word use can be indicative of style within authors and domains as it provides insight to the vocabulary and can thereby give a perspective on what the target group is and the quality of the writing. One downside to such feature is, that the vocabulary naturally expands, as the text length gets longer. Various formulas are created as an attempt to combat this issue. Specific details and implementation on the measures we used will be given in subsubsection 4.3.1.1.

The features, sentence statistics, word statistics, hapaxes, and lexical diversity, provide several scalar measures that give a decent exploratory overview of the general complexity of a text. Short articles might be taken as an indication of appealing to the general public whereas longer articles may imply in-depth discussion of a topic and may therefore have a narrower target group. However, we do not expect the scalar features to be the most important features, due to vector features being the dominant features in recent works (PAN features described in subsection 2.2.2). The word, character, and POS n-grams are the most utilized in SOA works so we expect these features to have high importance. They carry much more information than the scalar vectors due to their size and also include contextual and syntactical information.

3.6.1.5 *N-grams*

For some of our vector features we decided to include n-grams of words, characters, and POS-tags weighted by TF-IDF. Each n-gram type in Table 4 has three lengths: unigrams, bigrams, and trigrams.

Word n-grams have been included in studies of authorship attribution [82] as they are simple lexical features and can offer insights to common constituents used by authors as well as offer some content specific knowledge as mentioned in section 2.5. In using word n-grams, we gain insight into important word combinations of a document. Say you have an article about cheetahs with the sentence *Cheetahs run fast and cheetah cubs are cute*, there would be trigrams such as *cheetahs run fast, cheetah cubs are, cubs are cute*. It is important to note that word n-grams are content specific which can introduce bias in some cases of authorship attribution, e.g. if training text for one author contains a lot of words relating to cheetahs, a model may be inclined to automatically attribute a text containing similar words to said author. It can therefore be necessary to include stylometric features that are not biased by content and instead more directed towards style, e.g. character n-grams.

Character n-grams have been included in multiple studies [46, 35] and have also been found to be a useful feature in distinguishing authorship of texts. Due to the lack of semantic information in

character n-grams, it is possible to capture lexical, grammatical, and orthographic preferences [50]. We therefore included character n-grams to have a noise resilient feature.

POS-tag n-grams have previously been used to distinguish magazine articles from newspaper articles, as well as for identifying authors. As POS n-grams have been used for both authorship and newspaper attribution, this was incentive for us to include it

For all n-gram computations, we included unigrams, bigrams, and trigrams. Trigrams are considered long enough to capture valuable information while still being computationally workable [5]. Quadgrams and longer are, to the extent of our knowledge, rarely seen in such tasks.

The length of all vector features was set to maximum 1000 elements and were TF-IDF weighted. This value was chosen to be as long as possible, while still manageable in terms of memory usage. Choosing the longest vectors possible allows for reduction of the length if they turn out to be excessively long.

N-grams of different types have been found to be efficient feature in classifications tasks like ours, however, as noted by several studies [48, 28] a combination of these along with other features often perform better which is why have included several types of n-grams.

3.6.1.6 *Word Skipgrams*

As mentioned in subsection 2.5.2, skipgrams are word collocations in which k adjacent words are skipped. To the best of our knowledge, word skipgrams have not been tested in previous published studies but we know from prior experience that it performs well in an authorship verification task. The syntactical structure can vary from author to author and word skipgrams allow us to capture patterns between words that are not immediately adjacent to each other. With the sentence *the works of Shakespeare* where *k=1*, skipgrams would capture the bigram | *the of* | which can be a stylistic choice of the author in how they write the genitive case (i.e, possession). Regular word bigrams would result in *the works* and *works of* and thereby miss such information. Word skipgrams can therefore be useful in capturing syntactic information on a lexical level.

It further allows us to capture content information. With a sentence like *cheetahs run fast*, skip bigrams (*k=1*) would be <*start*> *cheetahs*[25], <*start*> *run, cheetahs run, cheetahs fast, run fast, run* <*end*>*, fast* <*end*>. Of these bigrams, some are more informative than others, e.g. *cheetahs fast* versus *cheetahs run*. In a document about cheetahs, one may find that *cheetahs fast* appear several times. Had only word bigrams been computed, *cheetahs run* would likely appear more

25 <start> and <end> are border symbols to indicate sentence beginnings and ends respectively. Border symbols are beneficial to include as a lot of important information can lie in which words tend to occur near the beginning or end of a sentence.

frequently, however, such a bigram is not as descriptive of the cheetah more so than it is a fact of a lot of animals.
With above considerations in mind, as well as our prior experience, we decided to include word skipgrams as a feature. To limit computation, we only included bigrams with different k.

3.6.1.7 *Function Words*

TF-IDF weighted frequency of function words were included as a vector feature because the use of function words has been proven a distinctive feature in a number of authorship studies [67, 47, 4, 105].

3.6.1.8 *Topics*

As mentioned in subsection 2.5.3, topics are useful for identifying newspapers as they tend to have topics of focus. Additionally, topics were shown to be a useful feature in identifying authors [85]. Although Seroussi et al. found the DADT model to yield better results, we decided to use LDA as it is commonly used for unsupervised topic modelling and was readily available for implementation

3.6.2 *Automatic Encodings*

In order to answer our first sub-question[26], we need to encode articles automatically to compare against our manually extracted features. Initially, the automatic features were to be extracted using an LSTM autoencoder, but this was changed to a transformer model instead. The reasons for this decision will further be elaborated in subsection 3.9.1 and the implementation specifics of the automatic encodings will be described in detail in section 4.4. The automatically generated features will be extracted in the same fashion as the manual, i.e. on the same data, using the same split. In further tests, the automatic and manual features will be combined into one feature set by concatenation, where we will uncover whether such combinations will prove better than each feature extraction standing alone.

3.7 DATA SUBSET

The pre-processed dataset had some shortcomings. Firstly, the distributions of domains and authors were skewed with both distributions having over- and underrepresented classes. Secondly, the

26 Will manually extracted features, automatically extracted features, or a combination of both generalize best?

number of unique authors is excessive (1,885), making a standard classification task difficult. As mentioned in subsection 2.2.1, this issue can be coped with by creating a vector profile on each author and classify based on cosine distance and also allowing the model to classify an article as having an unknown author. However, as the focus on this study lies on creating a general-purpose feature encoding and not maximizing classification metrics, we decided to create a subset of the dataset, limiting the number of classes and balancing the class distribution, enabling the use of conventional classifiers. Furthermore, a smaller dataset will allow for faster testing of models and feature extraction.

The question now is, should the database be compiled into one or more subsets? Choosing one subset would ensure, all tasks utilize the exact same input, removing one possible source of bias and making their metrics easier to compare directly. However, it would also result in compromising the ratio between domains and authors, i.e. the domain BT may have 1000 articles written by 800 different authors, each author only having written one or two articles. If the goal is to classify the domains, this is perfectly fine, however, it is not useful for identifying the authors, as the majority of authors do not have enough training data. Removing underrepresented authors in this manner would in the worst-case result in domains being completely pruned or only be represented by one author, which would ruin the ability to differentiate between newspaper and authorship.

On the other hand, creating multiple subsets would not suffer from this issue. Underrepresented classes in both domain and authors would be dropped and overrepresented classes randomly sampled. However, the data would need to be extracted once for each subset created, due to the TF-IDF weights being tied to the corpus. Finally, the data would not be exactly the same for each task.

However, the downsides of having a single subset surpasses the downsides of multiple subsets, as it blurs the distinction between the tasks, spoiling the actual goal of this project.

In regard to headline generation, the class distribution is not an issue, as it will not be used for a classification problem. Therefore, a headline subset will also be sampled, where domain and author are not of concern, but rather making sure no articles have the same headline. However, in subsection 3.4.4 we stated that repeating headlines of less than 20 occurrences were alright to keep. Though from initial tests with headline generation, we decided to remove all duplicates anyway.

The two subsets, author, and domain, underwent the same processes, albeit with different limits for sampling of classes. Underrepresented classes were dropped, and overrepresented classes were randomly subsampled with a maximum article count being at the third quartile of the distribution after dropping underrepresented classes. Some classification models may be more sensitive to

imbalanced data than others and a limit at the third quartile seems reasonable for a first iteration of tests. If the results are unsatisfactory and we deem the reason being the class distribution, we will re-sample the dataset. The lower limit for class representation was found based on the number of remaining classes after pruning, i.e. limit the number of classes and have a reasonable amount of articles per class. For the domain subset, domains with fewer than 1000 articles were dropped and for authors we simply took the top 100 most productive authors.

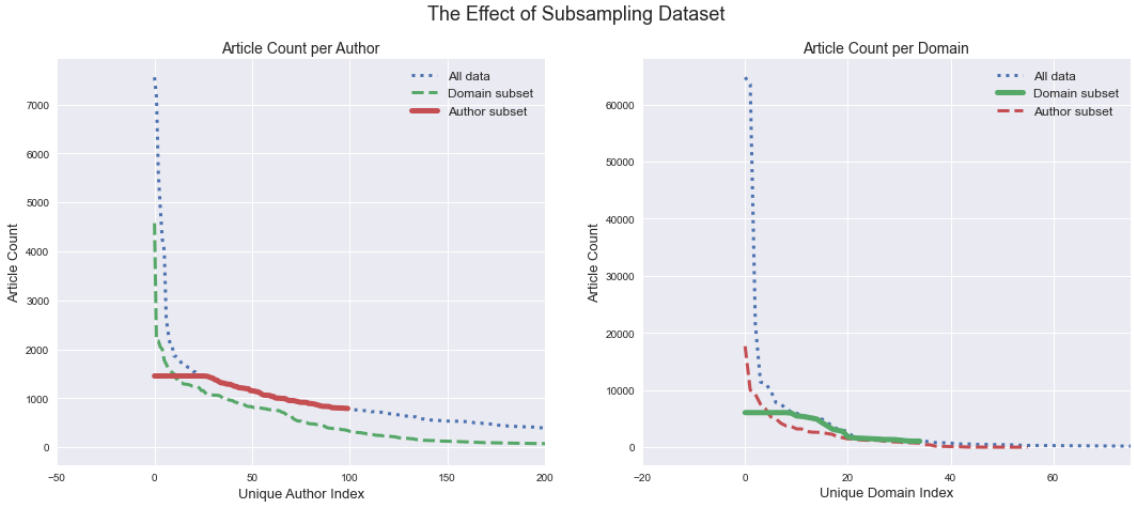


Figure 13: The distributions of the subsets, according to articles per author (left) and domain (right). Zoomed in to show relevant subsample.

In Figure 13, the distributions of articles per author and domain for each subset is seen compared against the full dataset. The highlighted subsets are the ones subsampled in the given class value, hence it looks significantly smaller in size compared to the other dataset. However, while article count has been greatly reduced, the difference between the two subsets in terms of article count is insignificant. The class count is what has been significantly altered as seen in Table 5.

Subset	**Authors**	**Domains**	**Articles**
Domain subset	1,458	35	124,058
Author subset	100	56	115,867
Header subset	1790	144	214,921
All data	1,885	160	301,416

Table 5: The value counts for all datasets.

Note, that the topic count was found by fitting the LDA model on the entire dataset. The training time of the LDA model is too long for fine-tuning the topic count to each subset, compared to our

expectations of performance gain. However, an LDA model with said topic count was fit to each subset to obtain the topic scores.

3.7.1 *Splitting the Data*

The subsets were split into train and test datasets prior to both manual and automatic feature extraction. The reason for doing it before extraction is, that the TF-IDF vectorizers in the manual extraction rely on the corpus available. Were the vectorizers to have access to the entire corpus, i.e. both train and test data, the weights in the test data would be biased by the training data. This procedure is not an issue with the automatically extracted features, but to ensure consistency between the feature sets, we performed the split before automatic extraction as well. The specifics of splitting the data is elaborated upon in section 4.5.

3.8 MODELS

As covered in section 3.1, the classifiers used are conventional models. We will utilize logistic regression and random forest for each of the classification tasks. Logistic regression (LR) is originally a binary classifier, fitting a sigmoid function to the data, separating the two classes. It assumes the data follows a binomial distribution, i.e. only two classes are available. It can be extended to a multiclass classifier by fitting it to a multinomial distribution and changing its loss from log-loss to cross-entropy. [27] Random Forest (RF) is an ensemble classifier, meaning it consists of multiple classifiers, which vote on an outcome. The random forest generates a series of decision trees. It has fast training time and convincing results, which makes it a good choice for a simple classifier. The Python library, Scikit-Learn will be used for these models, which will be compared against a dummy classifier. The dummy classifier is set to classify everything as the most frequent class, thereby having the greatest chance of being correct without taking anything from the extracted features into account.

For the headline generation, we will use the Python library, Headliner[28], which dramatically increases the ease of use of encoder-decoders. It was originally built for headline generation, hence the name, but can be used for other tasks, such as machine translation. As it, along with most other

27 https://machinelearningmastery.com (May 5th)

28 https://github.com/as-ideas/headliner (visited April 29th)

such libraries, by default takes text as input, we will have to set it up, such that our own encodings are the inputs. More on the process of implementing all our models in section 4.7 and section 4.8.

3.9 FAILED ATTEMPTS AND LIMITATIONS

The following subsections touch upon major implementation and design attempts that failed and the expected drawbacks of our approaches.

3.9.1 *The Usage of the LSTM Autoencoder*

The initial plan was to use an LSTM autoencoder for automatic feature extraction. We attempted this several times, using pre-made autoencoders and creating our own using Keras. However, the results produced by either were unsatisfactory. When using a ready-made LSTM autoencoder for text representation maintained by Eric Rocha Fonesca[29], we encoded only the headlines from Politiken and Information as a test, to see, if a random forest could classify the domain based on these encodings. The results were promising, as the model correctly classified all the headers correctly, no mistakes. Note, that the domain names were still present in the title, hence the suspiciously good accuracy. Comparing with the text representation by the transformer solution, the LSTM results were better. When performing the same operation on the bodies, the memory consumption was too large for our systems. We therefore abandoned the LSTM autoencoder approach and went with the transformer.

3.9.2 *Data Leaks*

As mentioned in section 3.4, we performed a thorough cleaning of our data. Heaps of articles were removed, salvaged, and cleaned. However, even after rule-based and manual cleaning, the dataset is still not completely sanitized. The amount of data is too large for us to comb through, so it is expected that some articles still suffer from data leakage. We expect the models to pick up on such instances, e.g. LDA topics containing biased words, such as CSS code if the cleaning still needs work. However, this will be further addressed in subsection 6.1.2 when the results are obtained. We expect the domain classification to be particularly affected by data leaks, as most junk is domain-specific, as explained in subsection 3.4.5.

29 https://github.com/erickrf/autoencoder

Summary 3 - Methodology

In this chapter we have described the choices we made for minimizing bias in our results. We have motivated our choices of features and presented the data along with the numerous steps takes to pre-process it. Finally, we settled on creating three subsets, one for each task, in order to avoid a skewed distribution of classes, and to give the headline generation model as much data as possible.

The key points from the methodology are:

- The focus of this project lies on the generalization of features across tasks. We therefore do not strive for as high accuracy as possible, but rather interpretability. (section 3.1)
- We set up our expectations for the results and described why we expect the classifications to perform better than headline generations (subsection 3.1.1).
- We cleaned up the data, removing over 500K articles in the process. We did our best of minimizing data leakages and other possible sources of bias (section 3.4).
- We investigated the distributions of the data in terms of classes and article counts, which led to the choice of separating the data into multiple subsets, due to class skew (section 3.5, section 3.7).
- The choices of features extracted were justified and explained in detail (section 3.6).
- We settled on the models logistic regression and random forest for classification and an encoder-decoder for headline generation (section 3.8).
- Finally, we described the expected drawbacks and failed attempts in the designing of the project (section 3.9).

4

IMPLEMENTATION

In this chapter, the implementation process is described in detail. First, in section 4.1 we present the programming tools used for implementation followed by how the web scraper for updating articles were created in section 4.2. In section 4.3, the implementation of all manual feature extraction processes are explained, followed by automatic features in section 4.4 in section 4.5 and section 4.6 we describe the processing of features and data. Finally, in section 4.8, we present the headline generation process.

4.1 SOFTWARE

All implementations are done in Python 3 with various libraries. The specific libraries will be explained in further sections when relevant and in appendix section B.1, a list of libraries used is available. Throughout the project, extensive use of Numpy, NLTK and Pandas was applied. Numpy is a library for scientific operations, especially useful for working with long lists of data, NLTK (Natural Language Tool Kit) provide useful functions for extracting features of natural language, such as n-grams, and Pandas allows for working with data in an Excel-like manner.

4.2 WEB SCRAPER

As written in subsubsection 3.4.4.1, a web scraper was implemented for updating incorrectly attributed authors and to fix malformed bodies of certain domains.
The scraper was made with a combination of the Python frameworks, `requests`, `Beautiful-Soup` and `selenium`. The latter opens up a virtual browser, which renders text generated by JavaScript, which is not possible with `requests`. The domains Berlingske and Ekstra Bladet would simply return JavaScript code instead of text when scraped with `requests`. While power-

ful, `selenium` needs to open up a webdriver for every request and close it again afterwards, likely due to spam protection. This process is slow and took upward 10 seconds per article.
The implementation of the web scraper was more laborious than initially expected. The domains had various methods of formatting authors for their articles and 404 errors had to be handled. In order to ensure that the scraper worked as intended, a series of unit tests were made, testing if various scenarios returned the expected values.
After 75 hours (PC) of scraping, *22,713* articles were salvaged (updated bodies and/or author), *8,100* articles failed to scrape or had multiple authors and were removed, and *12,523* articles timed out when fetching the author. The articles which could not be scraped due to a time out were kept, if the goal of the scrape was to update the author. The reason being that there is a chance that the originally attributed author was in fact the author of the article. Due to the long processing time, we decided not to retry scraping articles that timed out.

4.3 FEATURE EXTRACTION

Each article went through a series of processing steps in order to extract all the features described in section 3.6. In the subsections below, each process is explained, going in detail on the most advanced features.

4.3.1 *Scalar Features*

Seven scalar features concerning sentence length were extracted from each article body: the number of sentences, average sentence length in characters and words, the standard deviations of the same lengths and median lengths. Regex was used for splitting sentences at symbols used for ending sentences (e.g. full stop, question mark, exclamation mark) and also used for removing all symbols when counting characters. Word scalar features consists of four features: number of words, average lengths in characters, the standard deviation of the length and median length. Hapax legomena were obtained by creating an NLTK frequency distribution of words and using its built-in function, `hapaxes`.

4.3.1.1 *Lexical Diversities*

Eight different measures for lexical diversity were extracted. For seven of them, the library, `LexicalRichness` was used. Three versions of type-token ratios (TTR) were used. TTR

measures how diverse a vocabulary is by dividing the total number of words with the number of unique words. However, as Stamatos [91] stated, the length of the body will affect the vocabulary richness, which TTR does not account for. However, as series of other TTR versions are made in an attempt at solving this issue, in this case corrected type-token ratio (CTTR) and root-type token ratio (RTTR). Three more measures named after their inventors, Herdan, Summer and Maas are used. Finally, we implemented Lix, which is a commonly found measure using in the Danish elementary school system. The relation between the measures are plotted in appendix section B.2.

4.3.2 *Topics*

As the topic of the articles are not included in the meta data of our dataset, this feature was estimated using the LDA model from the Gensim library. An LDA model will find topics in an unsupervised manner with clustering, given a corpus as input. The issue with clustering is that the actual number of clusters (topics) is unknown. Having too few clusters will not capture all the actual topics while having too many will result in overfitting, i.e. one topic per document in extreme cases. To find a good number of topics, we used the `CoherenceModel` from Gensim, which gives a measure of, how coherent the topic keywords are. Testing from 8 to 29 topics, we found, that 22 topics performed best with a coherence score of 0.65, as seen in Figure 14.

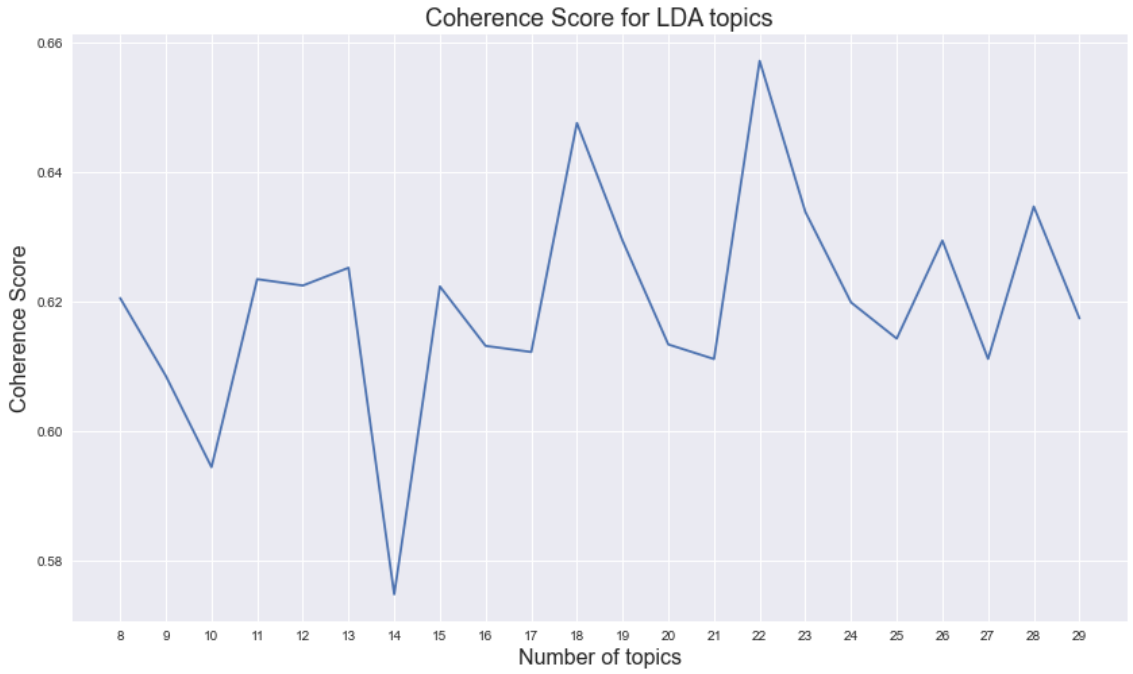


Figure 14: The coherence score of topic modelling as number of topics increase.

Further details on the LDA model is presented in subsection 4.3.2.
Topics can either be treated as vectors or scalars, as each topic has a score, i.e. one could make a vector containing all topic scores. However, as we are able to discern between topics, we decide to treat them as scalars. As described in subsubsection 3.6.1.8, the topics are obtained with the LDA implementation from the `Gensim` library. The model is trained on the training data of the respective subsets with the following hyperparameters: a chunksize (or batch size) of 2000, 20 passes (or epochs) with 400 iterations and 22 topics.
The topic count was later challenged by a repeated test. This issue will be discussed in subsection 6.2.1.

4.3.3 *Vector Features*

For all of our vector features we used TF-IDF weighted vectorization with a limited size of 1000 elements.

4.3.3.1 *Word and Character n-grams*

In computing the uni-, bi-, and trigrams we split the corpora into words and characters respectively and used the `ngram_range` parameter of the `TfidfVectorizer`.

4.3.3.2 *POS Tagging*

For the extraction of POS-tags we used the pre-trained `da_core_news_md` model for Danish through the Spacy library. The model was trained on a Danish news corpus and was therefore very suitable for our own corpus. The POS tagger from Spacy is based on the Universal Dependencies project, a multilingual treebank collection, which aims for consistent cross-linguistic annotations [71, 40].
We did a brief manual inspection of some handpicked documents and found that the tagger overall worked very well on the corpus with only a few mistakes, e.g. *selektivt* was tagged as ADV (adverb) in *selektivt udvalgte ting* although it is an adjective, and *Køb* was tagged as PROPN (proper noun) although it is a verb inflected with imperative. The errors were so few and far between that we did not consider it an issue.

4.3.3.3 *Function words*

In vectorization of function words, we used a list of function words provided to us by the IT company Ankiro ApS. The list contains the lexical classes outlined in the table below.

Lexical class	**Examples**
articles	*den, det, en, et, nogle*
dummy subject	*der, her, det*
infinitive marker	*at*
conjunctions	*fordi, eller, og*
pronouns	*denne, hver, nogle*
personal pronouns	*du, ham, hans, sig, sit, os, vores*
reciprocal pronouns	*hinanden, hverandre*
interrogative pronouns	*hvor, hvorfor, hvordan, hvorhen*
prepositions	*forbi, gennem, i, inde, langs*
interrogative adverbs	*hvornår, hvorved, hvorhen*

This list was used instead of NLTK's Danish `stopwords` as the stopwords are not limited to function words, which is what we are specifically interested in due to the common use of function words in authorship attribution tasks [4, 14, 67]. The list from Ankiro ApS contains 242 words. To limit time spent on pre-processing, we excluded phrases that spanned over several words, e.g. *på grund af, i stedet for, med hensyn til*.

The feature was implemented by stripping articles from all words except function words, i.e. bodies were represented by only the function words that occur in the text. For this reason, we only computed TF-IDF of unigrams because bigrams, e.g. [*in we*], would not be very informative in that it was unknown how many words was removed in between those two function words. Anything beyond unigrams might then assume patterns that are not necessarily meaningful.

4.3.3.4 *Skipgram Bigrams*

In order to vectorize skip bigrams we used the `skipgrams` function from NLTK which takes the document, split into words, as input with skip distance and n-gram size as parameters. The function was applied to each document before vectorizing with TF-IDF. Skip bigrams were computed with k ranging from 1 to 3.

4.4 AUTOMATIC ENCODING

The automatic encodings were obtained with a `RepresentationModel` from the SimpleTransformers[30] library. The model generates a vector representation of the input texts using a BERT model, with the Danish fine-tuning, `Maltehb/danish-bert-botxo` which has been trained on Wikipedia articles in Danish. The model produces vectors with a size of 768 which we use as the automatic encodings.

4.4.1 *Combination of Manual and Automatic Encodings*

For combining the manual and automatic encodings obtained, the two types were simple concatenated, making each article be represented by manual features followed by automatic features.

4.5 DATA PROCESSING

As mentioned in section 3.7, we decided to split the data into subsets in order to limit the amount of classes for authorship and newspaper attribution, as well as avoid over- or under-representation of certain classes. The data for each subset was split into training and test sets using Scikit-Learn's `train_test_split` function. The test size amounts 20% of the total data and the random state is set to **42** in all splits to enable reproducibility. Each subset split is stratified according to its target value, except the headlines dataset, as its targets are not classes.
The article count for the train and test sets for each data subset are seen in Table 6.

Subset	**Train**	**Test**
Domain subset	94,572	23,644
Author subset	92,693	23,174
Headline subset	171,936	42,985

Table 6: The train and test sizes for each task subset.

30 https://simpletransformers.ai/ (visited May 28th)

4.6 FEATURE PROCESSING

As mentioned above, we split the data into train and test data prior to feature extraction. The features were saved, one file for training and one for test data and the same was done for target values. As the target values for the classification tasks were encoded to be class indices, dictionaries containing which index corresponds to which class were also saved, i.e. index 3 for domains corresponds to DR. This later allowed us to better interpret the data. We also saved the feature names of the extracted features according to their corresponding indices, which also will be used for later insights. The processes of manual feature extraction for each subset took between 6 and 7 hours (Cloud) and automatic feature extraction took 7 hours (PC). Exact values for each extraction was lost.

4.6.1 *Classifier Unit Test*

A classifier unit test was created to test if a feature combination would perform well. The script takes an Scikit-Learn classifier as input and compares it to a dummy baseline on a series of different metrics. The script produces true and false positive rates, which can be plotted against each other, creating a Receiver Operation Characteristic (ROC) curve for visually assessing classification performance. Additionally, the script outputs a confusion matrix and can be configured to return feature importances obtained with permutation importance. However, for the feature importances presented in this report, we used a random forest model, as it has the attribute `feature_importances_` built in, i.e. no additional processing is required, which would be the case with permutation importance. It must be noted that feature importance may vary between models, so our results are just one instance of such.

4.6.2 *Metrics*

Due to the high number of classes, and the imbalance found in the distribution of classes, some evaluation metrics will be less useful than others. Accuracy often results in misleading results, e.g. if the classes are imbalanced it can appear unrealistically high. To get the best comparison between models, we used many different metrics available in Scikit-Learn. Our main metrics presented in results are accuracy, F1, ROC AUC, and top K accuracy. The whole list is available in appendix section B.3. Top K accuracy will show in decimal how often the correct label is among the top K

predictions by the classifier. We decided to set K to 5. ROC AUC is the Area Under Curve for the ROC curve. This metric is tied to the ROC curve plots, and a perfect classifier has a ROC AUC of 1. Finally, F1 as mentioned in Box 4 is a commonly used metrics and takes recall and precision into account.

4.6.3 *Obtaining ROC Curves*

A ROC curve is a good method of visually interpreting the performance of a binary classifier, i.e. a classifier making predictions between only two classes. As none of our classifiers are binary, we cannot directly create a ROC curve for them, but it is still possible.
On the website for Scikit-Learn, a method for obtaining this is available[31]. This approach utilizes a `OneVsRestClassifier`, which fits one classifier per class, effectively converting the multi-class problem to a binary class problem, one can then plot the average ROC curve, across all classes. The downside to this, is an exploding training time, as classifiers now will be fit once for each class, which means 100 times longer training time, which is not feasible.
Our workaround is not as accurate but works for plotting and comes at no additional cost in processing time. The function for getting the true and false positive rates is showcased in Listing 1.

```
1    def fpr_tpr(preds_proba, y_test, name=None):
2        y_test = label_binarize(y_test, classes=np.unique(y_test))
3        fpr, tpr, _ = roc_curve(y_test.ravel(), preds_proba.ravel())
4        print(f"ROC AUC control for {name}: {auc(fpr, tpr):.2f}")
5    return fpr, tpr
```

Listing 1: Function for obtaining true and false positive rates of multi-class predictions.

The input parameters for the function are probability predictions and target values in arrays and the keyword parameter for printing the model name. On line 2, we binarize the target values, making one-hot-encodings. The true and false positive rate is then obtained on line 3 with the Scikit-Learn function, `roc_curve` using the raveled versions of the two arrays as inputs. The estimated ROC AUC will then be printed on line 4 for comparison with the actual ROC AUC score.

31 https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html (visited May 12th)

4.6.4 *Standard Scaling and Dimensionality Reduction*

We experimented with scaling the authorship attribution data to unit variance using the `StandardScaler` class from Scikit-Learn prior to training the models. However, as the random forest is robust in terms of varying unit scaling, scaling the data yielded worse scores. Please note, that this is only the case for a random forest model, as other models such as logistic regression are sensitive to the feature scaling[32]. Therefore, we will standardize the input for the logistic regression and not for the random forest.

In terms of dimensionality of the data, each article would consist of up to 10180 features, which means the training times for the models are impractically long. One way to speed up the training is to reduce the dimensions of the data using principal component analysis (PCA), where eigenvectors capture the dimensions of the data with the most variance. PCA will yield best results if the data is standardized prior to applying PCA[33] which we will do. 4300 principal components were needed to capture 99% of the variance in the authorship attribution training data. However, the results obtained when fitting the random forest to the reduced dataset were noticeably worse than the raw data, hence PCA will not be used when training models.

4.7 CLASSIFICATION MODELS

The models, random forest and logistic regression were both applied using the implementations found in Scikit-Learn. All models used the random state **42**, and had the parameter `n_jobs` set to -1, utilizing all 16 available parallel processes (Cloud), speeding up the training time. Additionally, the `verbose` parameter was set to true, to follow the training progress. All other parameters were left as the default options.

4.8 HEADLINE GENERATION

As mentioned in section 3.8 the headline generation model is implemented using the library, Headliner. The specific model used from this library is `BasicSummarizer`, which is an LSTM encoder-decoder model. Other choices are a summarizer with an attention layer, a transformer-based summarizer and a transformer-based summarizer on top of a BERT pre-trained model. All

32 https://scikit-learn.org/stable/auto_examples/preprocessing/plot_scaling_importance.htm
33 https://scikit-learn.org/stable/auto_examples/preprocessing/plot_scaling_importance.html

models use a masked crossentropy loss function. We experimented with all types of models, but the basic model provided the best results, likely due to it being simpler and needing less configuration. The authors of Headliner found the best results for headline generation to be produced by the BERT-based model. However, they state, that it is sensitive to the configuration of the hyperparameters[34]. The headline generation underwent a series of tests, before being applied to our own encodings.

4.8.1 *Validating the Model*

In the documentation of Headliner, they provide an example of a neural machine translation model[35] using the attention-based summarizer. The goal was to translate English phrases into German. We replicated this process, which was a success.

As the encoder-decoder model in theory can transform any sequence into another, we treated headline generation as machine translation, i.e. using the same model for headline generation as for the German translation. The next step in setting up the model was to apply it to an English newspaper dataset[36], generating headlines using the first 400 characters of the article, similar to the approach by Dorr et al. [23], who used only the first sentence of an article. The reason for using this dataset was that they most likely had more clean data than us, ruling out the possible bias of junk contaminating the results. The results were promising, so we decided to use our own data.

Again, to completely validate the model and rule out sources of error, we trained the different models using textual input in form of the first 400 characters of all the articles in the author subset. Using text as input means that the model itself vectorizes and learns encoded representations of the text. Were the results to be satisfactory, the encoder would be replaced with our own text encodings. In this experiment, none of the summarizer models, except the basic summarizer, would produce unusable headlines. Therefore, we went with the `BasicSummarizer`.

4.8.2 *Initial Headline Generations*

After training the model for 30 epochs on the textual input, the following results were obtained: in Table 7, manually selected examples of good headlines are highlighted, and examples of poor predictions are presented in Table 8. Both of these tables show results for training data, i.e. the

34 https://medium.com/axel-springer-tech/headliner-easy-training-and-deployment-of-seq2seq-models(visited May 27th)
35 https://as-ideas.github.io/headliner/examples/nmt_example/ (visited May 11th.)
36 https://www.kaggle.com/datasets/snapcrack/all-the-news (visited (visited May 9th.)

model has trained using these headlines, so we should expect them to be better than the ones found in the test data.

Target (train data)	**Prediction**
Spilforslag: Power Hour i Aalborg	spilforslag power hour i københavn
DMI advarer: Voldsomt vejr på vej	dmi advarer bilister på vej mod skybrud
Regeringen misser mål for ulandsbistanden	regeringen om at fremrykke mål
To biler kørte sammen	to biler stødte sammen

Table 7: Manually selected headline predictions using the training data, i.e. the model has seen these headlines before. Predictions obtained with textual input.

Target (train data)	**Prediction**
Indtægtstab: Saudi-Arabien kan leve længe med lave oliepriser - og måske nyde godt af dem	kan man huske , at der er for at være med til at være en god idé ?
Et dansk køretøj er angrebet med brandbar væske i Kabul	dansk supermarked er i fare for at voldtage
VIDEO Islamisk Stats tortur-fængsel for kvinder opdaget i Syrien | Udland	video mette frederiksens video af trump og en dårlig dag for en dårlig dag for en dårlig dag for en dårlig dag for en dårlig dag for en dårlig dag
Coronavirus	# .# kroner til at tage en halv milliard

Table 8: Examples of poor predictions in the training data, i.e. the model has seen these headlines before. Predictions obtained with textual input.

The predicted headlines of course vary in quality, but the results seen in Table 7 look promising. However, as these are based on training data, it is not representative of how well the model generalizes. In Table 9, manually selected examples of headlines generated on test data is presented. These headlines are certainly getting the gist of the article. However, the model also produced an abundance of poor results, as presented in Table 10.

Target (test data)	**Prediction**
Mand får bøde efter råberi	mand får bøde efter dødsulykke
Politiet snuppede bilist med 91 km/t ved skolevej: - Vi kommer snart igen	politiet efterlyser vidner med # kmt - ringsted
46-årig mand fængslet for knivoverfald	#-årig mand sigtet for voldtægt af #-årig
Video: Opskrift på ærtepuré	video opskrift på snaps

Table 9: Manually selected headline predictions using the test data, i.e. the model has not been trained on these specific headlines. Predictions obtained with textual input.

Target (test data)	**Prediction**
Kaos i Bon Bon-Land: Sked på børns slik og bamser	en god dag for at få en dårlig dag for at få en dårlig dag for en dårlig dag for en dårlig dag for en dårlig dag for en dårlig
Øjenvidner i chok i Volgograd: Det er et mareridt	i dag er der en mand i en uge
Trumps køb af Grønland var ikke på bordet	unge børn er ikke nok til at være på en anden
200 år gammelt Austen-manuskript er til sal	# gode råd til #

Table 10: Poor headline predictions using the test data, i.e. the model has not been trained on these specific headlines. Predictions obtained with textual input.

In many cases, the predicted headlines are, in a textual sense, far from the targets, but close in topics. For example, the target *City udstillede svage United i fantomhalvleg* with the prediction *tidligere landsholdsspiller fortsætter i målfest* are far from each other in meaning, but they both revolve around football.

We deemed these results satisfactory, so we decided to use our own encodings as input for the decoder. The final hyperparameters for the `BasicSummarizer` are: an embedding and LSTM size of 1024, maximum prediction length of 30 words. Its training parameters are: batch size of 64, 100 steps per epoch. We left the vocabulary size as default, which is 200K. Our own vocabulary in this test is 103K.

4.8.3 *Using Our Own Encodings*

We needed to have the headline generator use our own encodings as input for the decoder. To do that, we implemented our own vectorizer, i.e. the module converting text to a vector, as the ability to create custom vectorizers was implemented as a feature in the Headliner library. As in their documentation, we will use the term vectorizer and tokenizer interchangeably in this subsection. As the purpose of the tokenizer is not to convert strings to vectors, but rather pass on a vector, we would need to handle that. The tokenizer class is seen in Listing 2.

```
class TrickTokenizer(Tokenizer):
'''This tokenizer is actually just tricking Headliner ;-)
The input are strings, which will be converted to arrays.'''

def encode(self, text):
    '''Encoder is simply just converting string to array'''
    text = text.replace('<start>', '')
    text = text.replace('<end>', '')
    text = text.replace(' ', '')
    text = text.replace('[', '')
    text = text.replace(']', '')
    return np.fromstring(text, dtype=float, sep=',')

def decode(self, sequence):
    return "NOTHING"

@property
def vocab_size(self):
    return 200000
```

Listing 2: The tokenizer used for passing on vectors to the headline generator.

The functions `encode`, `decode` and `vocab_size` were requirements for the class, as stated by Headliner. As the vectorizer will not be used for decoding anything, it simply returns "NOTHING" no matter the input in the decode function. We set up the encode function to take in a stringified version of the actual vector. It contains end and start tokens, which are removed along with whitespaces. Finally, the vector is returned with the Numpy function `fromstring`, which can convert a string to an array. We contemplated using indices instead, but this proved difficult, as the vectorizer will both work with training and test data, hence the actual vector as a string for input was chosen.

Prior to vectorization, we converted the entire training data into strings, which could then be "encoded" by our own tokenizer. The vectorizer used for encoding the actual headlines is a Tensorflow `SubwordTextEncoder`, the same encoder used in the Headliner example used in subsection 4.8.2.

Finally, we do not want to simply pass our encodings as input to the model. The model is an LSTM encoder-decoder, and the encoder part is creating abstract representations of the input, as explained in section 2.1. In our case, we want to completely replace the encoder with our own encoding. Therefore, we set the model parameter `embedding_encoder_trainable` to false, making the model unable to train its encoder, but instead makes it pass our encoding forward to the decoder.

4.8.4 *Drawbacks*

The method described above worked on our manually extracted encodings. However, due to the sheer size of dimensions each article had, the training time was incredibly time consuming, i.e. 165 hours on Cloud for 17 epochs. We experimented with reducing the dimensions using PCA and SVD (Singular Value Decomposition), however, neither worked with the mentioned method. The same went for the automatically extracted encodings. Due to the amount of time it would take to narrow down the issue along with the long training time, we decided to focus on training the model on only our manually extracted features, as this is the area which has not been covered in previous literature. Further discussion will follow in subsection 6.1.4.

4.8.5 *ROUGE Evaluation*

For evaluation of headlines we used ROUGE, an evaluation metric used for automatic evaluation of summaries [56]. ROUGE has different measures of evaluation for comparing predicted summaries against target summaries. We evaluate with the ones stated below.

- ROUGE-N counts overlapping n-grams between prediction and target summaries
- ROUGE-L measures the longest matching sequence between a prediction and target.

We implemented ROUGE with a pre-existing library, `rouge`, that computes ROUGE-1, ROUGE-2, and ROUGE-L.

Summary 4 - Implementation

In this chapter we have gone over our implementation process. For each data subset, we extracted scalar and vector features which make up our manual encoding. To obtain automatic encodings, we used a transformer model, and for headline generation, we used an encoder-decoder model from the library Headliner.

The key points from the implementation are:

- To salvage articles from selected domains, we implemented a web scraper (section 4.2).
- We extracted scalar and vector features for all subsets which make up our manual encoding (section 4.3).
- We tested feature reduction, but results were worse than the original features (subsection 4.6.4)
- We selected a series of metrics for evaluation of the classification tasks, hereunder a ROC AUC which was modified to fit a multi-class classifier (subsection 4.6.2).
- We used random forest and logistic regression for classification (section 4.7).
- For headline generation, we modified an encoder-decoder model to use our manual encodings rather than the built-in encoder (section 4.8).

5

EVALUATION

In this chapter we will present the results obtained in using the encodings described in the previous chapters to our three tasks. Both raw metrics and more in-depth visualizations will be presented to allow for a thorough interpretation and discussion of the findings in chapter 6. As established in section 3.8, all classifications will be compared against a baseline in the form of a dummy classifier. In section 5.1, we first present the results from the classifications obtained by an initial test without a baseline, followed by the final classification results in section 5.2. We then provide insights into feature importance in section 5.3. In section 5.4, we have chosen to only present the domain results, (e.g. section 5.4 and subsection 5.4.1) as the domains have fewer classes and because we have a pre-existing knowledge of the domains' characteristics, making later interpretations less convoluted. We also present confusion matrices in section 5.5. The generated headlines will undergo evaluation using ROUGE, however, we also provide qualitative evaluations in section 5.6.

5.1 INITIAL CLASSIFICATION RESULTS

Based on the domain and author subsets, we extracted the features mentioned in section 4.3 and used a random forest classifier to obtain preliminary results which are shown in Table 11.

Encoding type	Accuracy	F1	AUC	Top 5 accuracy
Author manual	**0.28**	**0.24**	**0.84**	**0.50**
Author auto.	0.16	0.14	0.73	0.33
Newspaper manual	0.82	0.82	0.98	0.97
Newspaper auto.	**0.96**	**0.96**	**0.99**	**1.00**

Table 11: Initial results of authorship and newspaper attribution using a random forest classifier. Best performing in each subset on each metric is highlighted in bold

As seen, the classification of domain when using the manual encodings yielded an F1 score of *0.82* which was suspiciously good. The automatic encodings achieved a near perfect F1 score. Upon seeing the initial results, we inspected the first 50 entries for different domains and found that almost all had junk in the beginning and end of the article. E.g. Berlingske and BT had *Forsæt Fortsæt Luk* in the beginning of most articles and a lot of the domains we inspected (e.g. Berlingske, Information, Ekstra Bladet, Se Og Hør[37], Billed-Bladet) had dates in either the beginning or end of the article. Despite the various pre-processing steps we took (described in section 3.4), data leakage was evidently still an issue. We contemplated using regular expressions to capture patterns specific to each domain but in the interest of time we ultimately decided to strip all articles of the first and last 100 characters with an assumption that the loss of data was expendable. The reason for junk being present mostly in the start and end of the article is due to the fact, that the header and footer of the websites are the same across all articles. To avoid bias due to article length, we only included articles that were more than 200 characters long. This process was also performed on the headline subset. The result of this post cleanup process is shown in Table 12 below.

	Author subset	**Domain subset**	**Headline subset**
Initial size	115,867	124,058	300,8122
Post clean up size	112,445	118,216	291,566
No. removed articles	3,422	5,842	9,256

Table 12: Subset reduction as a result of removing articles shorter than 200 characters

5.2 FINAL CLASSIFICATION

The classifiers random forest and logistic regression were used for classifying authors and newspapers using manually extracted features, automatically extracted features, and a combination of these two. The training times for each run is available in appendix section C.4. Results of authorship and newspaper attribution are shown in appendix Table 13.

37 https://www.seoghoer.dk/

Encoding type	Model	Accuracy	F1	AUC	Top 5 Acc.
Author Manual	RF	**0.26**	**0.23**	0.83	**0.48**
	LR	0.22	**0.23**	**0.85**	0.44
	DUM	0.01	0.00	0.50	0.02
Author Auto.	RF	0.16	0.13	0.72	0.32
	LR	**0.22**	**0.19**	**0.85**	**0.44**
	DUM	0.01	0.00	0.50	0.02
Author Comb	RF	**0.27**	0.23	0.83	**0.48**
	LR	0.24	**0.25**	**0.87**	0.47
	DUM	0.01	0.00	0.50	0.02
Newspaper Manual	RF	0.74	0.73	0.98	0.94
	LR	**0.78**	**0.78**	**0.99**	**0.96**
	DUM	0.05	0.00	0.50	0.08
Newspaper Auto.	RF	0.47	0.44	0.92	0.81
	LR	**0.68**	**0.67**	**0.98**	**0.93**
	DUM	0.05	0.00	0.50	0.08
Newspaper Comb.	RF	0.75	0.75	0.98	0.95
	LR	**0.82**	**0.82**	**0.99**	**0.97**
	DUM	0.05	0.00	0.50	0.08

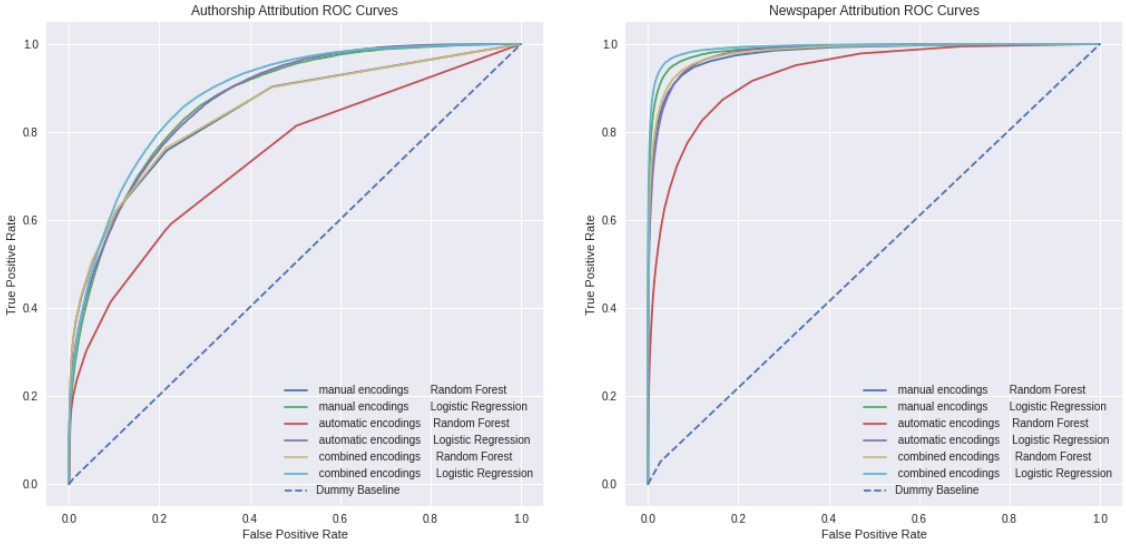
Table 13: RF = Random forest. LR = Logistic regression. DUM = Dummy baseline. Results of the four chosen performance metrics for each classifier shown for each type of encodings. *Manual* denotes our own manually extracted encoding, *Auto.* indicates encodings from `SimpleTransformers`, and *Comb.* is when the two text encodings are combined. The best of each metric for each text encoding is highlighted in bold.

As evident from Table 13, the classification of newspapers was overall better than the classification of authors across all metrics and all models across all tasks and encodings are outperforming the baseline. In comparison to the initial classification results (Table 11) with random forest, accuracy, F1 and top 5 accuracy have all gone down after the additional pre-processing, although AUC has remained the same. For authorship attribution, the combined encodings yielded better or equal results on all metrics which correlates with our expectations stated in subsection 3.1.1. With respect to newspaper attribution, the combined encoding produced better or equal results on all

metrics compared to the manual encodings. Despite the extra pre-processing, the classification for newspaper attribution is still near perfect which we discuss in the next chapter.

In regard to classification with random forest and logistic regression, logistic regression performs better across all metrics for the manual and combined features in newspaper attribution.

In Figure 36a and Figure 36b the ROC curves for authorship and newspaper attribution are plotted, from which it is also evident that newspaper attribution was far better overall. For both authorship and newspaper attribution, the worst performance (excluding the dummy classification) is on the automatic encodings using random forest.



(a) ROC curves for authorship attribution using random forest and logistic regression with each type of encoding. Worst performing is random forest on automatic encodings.

(b) ROC curves for newspaper attribution using random forest and logistic regression with each type of encoding.

Figure 15: The ROC curves for newspaper attribution are generalley better than for authorship attribution. Larger version of the figures are available in section C.1

5.3 FEATURE IMPORTANCES

In order to assert which features perform better in classification, we plotted the feature importances. The feature importances are obtained using the random forest model on manually extracted features. Results may vary if a different model is used. This process allows us to compare if there are differences in the best performing features for authorship and newspaper attribution respectively. (A larger version of the plot below is available in appendix section C.2 or online[38])

38 https://kroglkt.github.io/articlencoding/

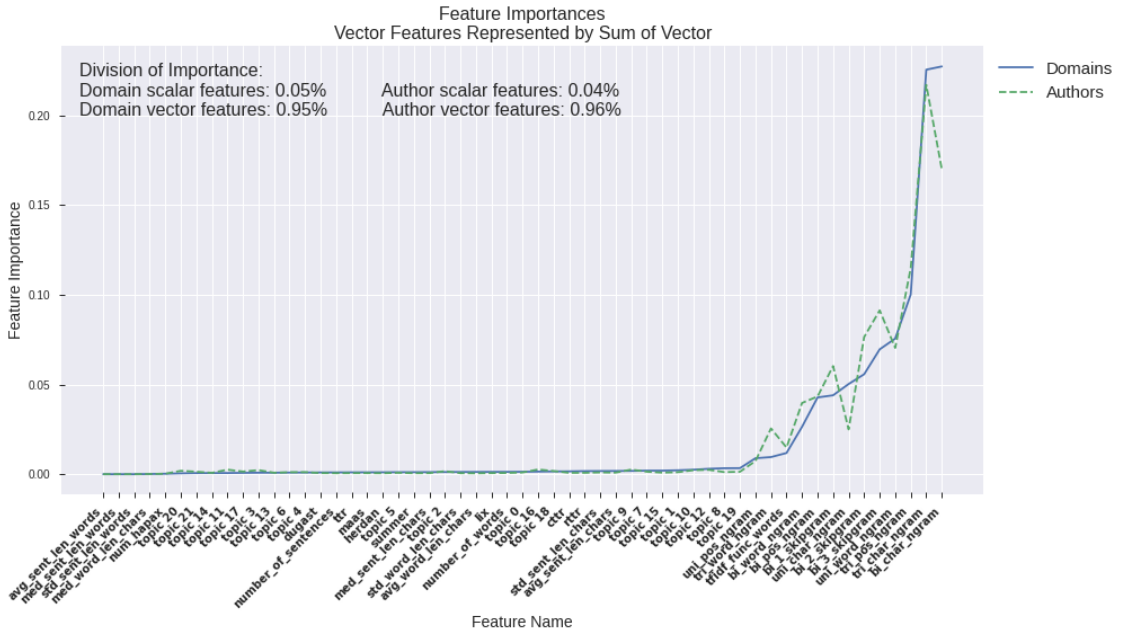
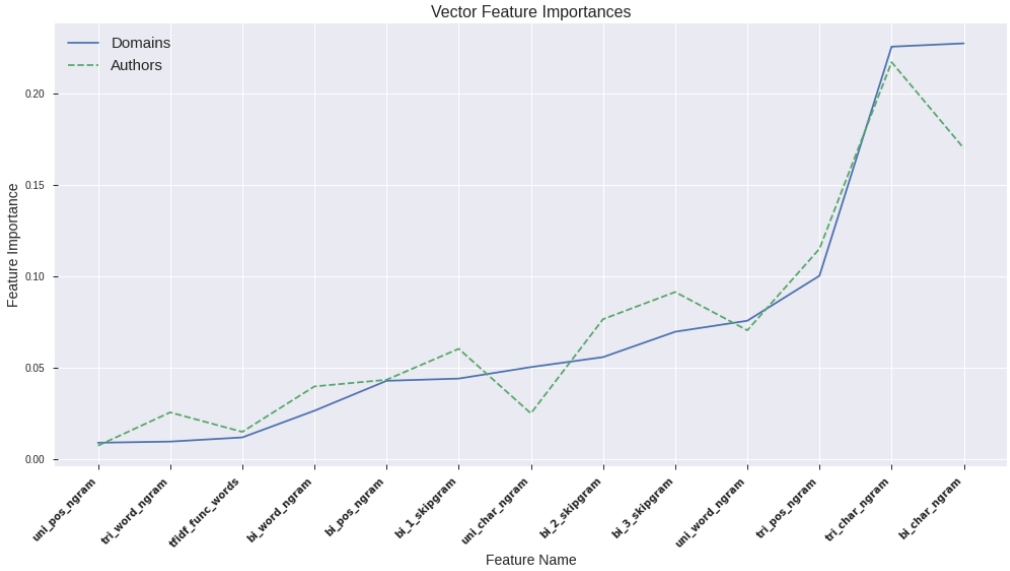
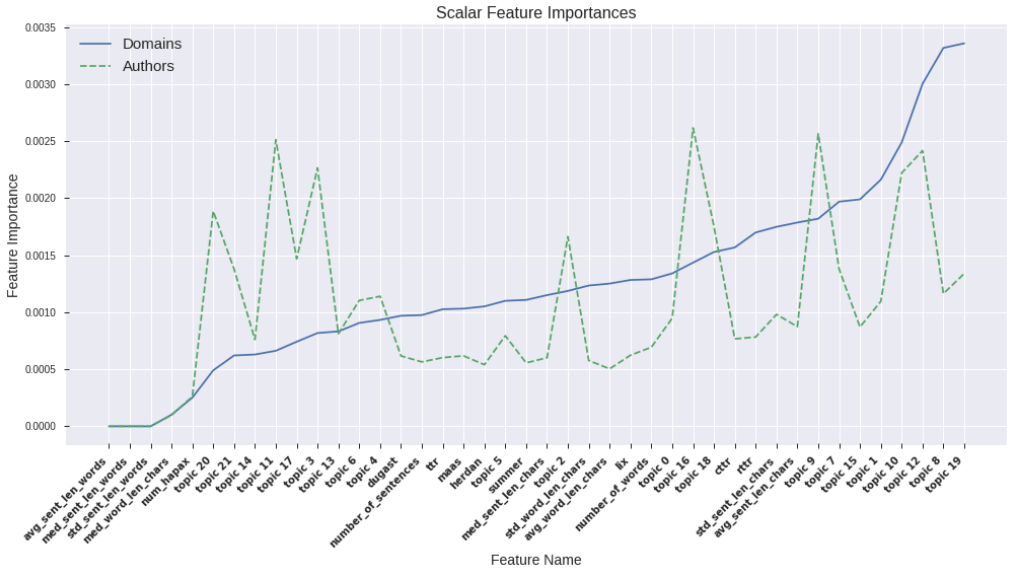


Figure 16: Importance plotted for all features for the author and domain subset respectively

In Figure 16 it is clear that vector features have higher impact on results for authorship and newspaper attribution which conforms with our expectations. In Figure 17a and Figure 17b the importances of vector and scalar features are shown in relation to only vector and scalar features respectively in order to gain better insight into how the features impact the results of each task.



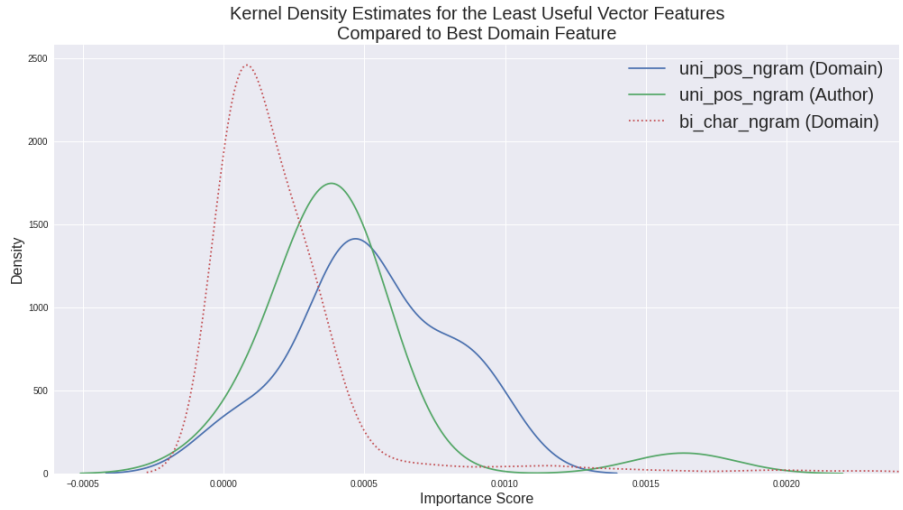
(a) Vector feature importances



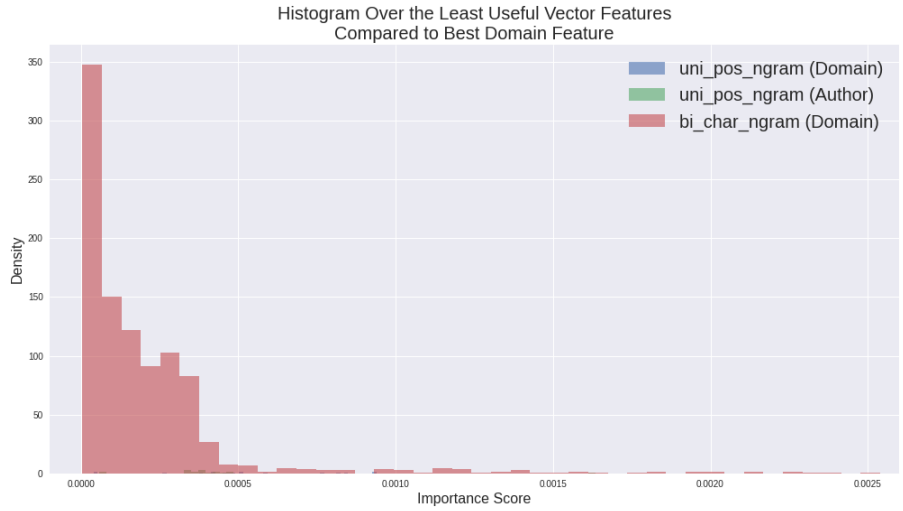
(b) Scalar feature importances

Figure 17b illustrates that certain topics aid the attribution of authors, and that the top 8 performing scalar features for newspaper attribution are also topics. Please note the scale for importances is smaller for scalar features than vector features, however, the scalar importances relative to each other provide insight which will be further discussed in the next chapter.

In Figure 17a we observe that the importance of each vector feature for authorship and newspaper attribution more or less follows the same trend, although skip bigrams where $k=2$ and $k=3$ have a little more impact on authorship attribution results. Additionally, the best performing features for authorship and newspaper attribution are character trigrams and character bigrams respectively. It is also worth noting that the fourth best feature are word unigrams, a feature that captures content.



(a) Estimated probability density functions of the worst performing vector feature in domain and authorship attribution tasks. Y-axis shows probability, making it look misleading in terms of importance.



(b) Histogram plot of the same data as Figure 18a. The worst performing vector features are just barely visible, as compared to the best performing, as the y axis now shows actual measurements.

In Figure 18a, the estimated probability density functions (PDF) for the worst performing vector feature in authorship attribution and newspaper attribution (both POS unigrams) are presented along with the PDF of the best feature for newspaper attribution (character bigrams). The POS unigrams seems to have a higher probability of having higher importance than the character bigrams. However, in Figure 18b, the actual histogram reveals why the POS unigrams are not performing better. While they have a higher importance, there are fewer of them, making them less significant.

5.4 NEWSPAPER TOPICS

The LDA model created 22 topics, outputting a series of keywords for each. One LDA model is created for each subset, i.e. one for domains, authors, and headlines. In Table 14, the topics derived from the model on the newspaper subset are shown below. No obvious junk keywords are present in the topics.

Topic ID	**Keywords**	**Interpretation**
Topic 0	børn, liv, unge, kvinder, far, ja, mor, virkelig, familie, familien	Family
Topic 1	kr, pct, mio, selskabet, kunder, mia, virksomheden, bank, skat, virksomheder	Finances
Topic 2	offentlige, fokus, ansatte, virksomheder, medarbejdere, samarbejde, skabe, sikre, medlemmer, formand	Labour
Topic 3	eu, europa, europæiske, lande, brexit, debat, kommissionen, tyske, parlamentet, bruxelles	Europe/ EU politics
Topic 4	politiet, politi, årig, klokken, sagen, personer, skete, fundet, aften, manden	Police/Crime
Topic 5	aarhus, dr, kim, jesper, jan, skole, pedersen, jacob, elever, skolen	UNK
Topic 6	blot, findes, ofte, væk, ganske, side, steder, små, fleste, form	UNK
Topic 7	film, filmen, serien, oscar, instruktør, erne, serie, sæson, spiller, netflix	Movies/ TV-series
Topic 8	it, data, kina, fn, system, systemet, kinesiske, teknologi, systemer, flygtninge	IT
Topic 9	festival, årets, publikum, scenen, koncert, roskilde, musik, live, navne, koncerter	Music events
Topic 10	kommune, kommunen, området, københavn, byen, odense, lokale, by, helsingør, varde	Local
Topic 11	mad, is, øl, syrien, spise, kurdiske, restaurant, kød, franske, restauranter	Food
Topic 12	prins, google, facebook, microsoft, nettet, apple, brugere, digitale, windows, dronning	Tech Giants
Topic 13	biler, energi, grønne, bil, koster, bruger, bruges, køre, kører, bilen	Vehicles
Topic 14	peter, lars, henrik, thomas, nielsen, søren, jensen, michael, christian, københavn	Names
Topic 15	of, in, and, on, you, all, is, me, it, black	English words
Topic 16	mål, kampen, hold, vm, kamp, spiller, holdet, vandt, point, kampe	Sport
Topic 17	sagen, medier, sag, sager, fejl, oplysninger, spørgsmål, kritik, regler, svar	UNK
Topic 18	regeringen, venstre, overblik, politiske, mette, dagens, folketinget, politisk, frederiksen, politik	Politics
Topic 19	album, sange, sang, musik, albummet, video, sangen, numre, nummer, synger	Music
Topic 20	usa, amerikanske, trump, præsident, new, rusland, york, britiske, john, donald	International politics
Topic 21	antallet, tal, corona, region, uge, danskere, landet, ramt, antal, personer	COVID

Table 14: The 22 predicted topics for the domain subset as produced by the LDA model. Each topic has a collection of keywords from which we created an overall interpretation.

5.4.1 *Visualizing High Dimensions*

Which topics do different newspapers tend to write about? Visualizing high dimensional data is not straight-forward. Humans are capable of understanding three dimensions, though each article is

represented by 10k dimensions. However, it is possible to compile multiple dimensions down to a few, making it more easily understood. In the following subsections we present a series of data insights, where the dimensions have been reduced.

5.4.2 *PCA Reduction*

Though PCA was not used for reducing the dimensions prior to training the models, PCA can be used for visualization. In Figure 19, the first three principal components for the domain features are plotted, showing the variance captured by these three components in the domain dataset.

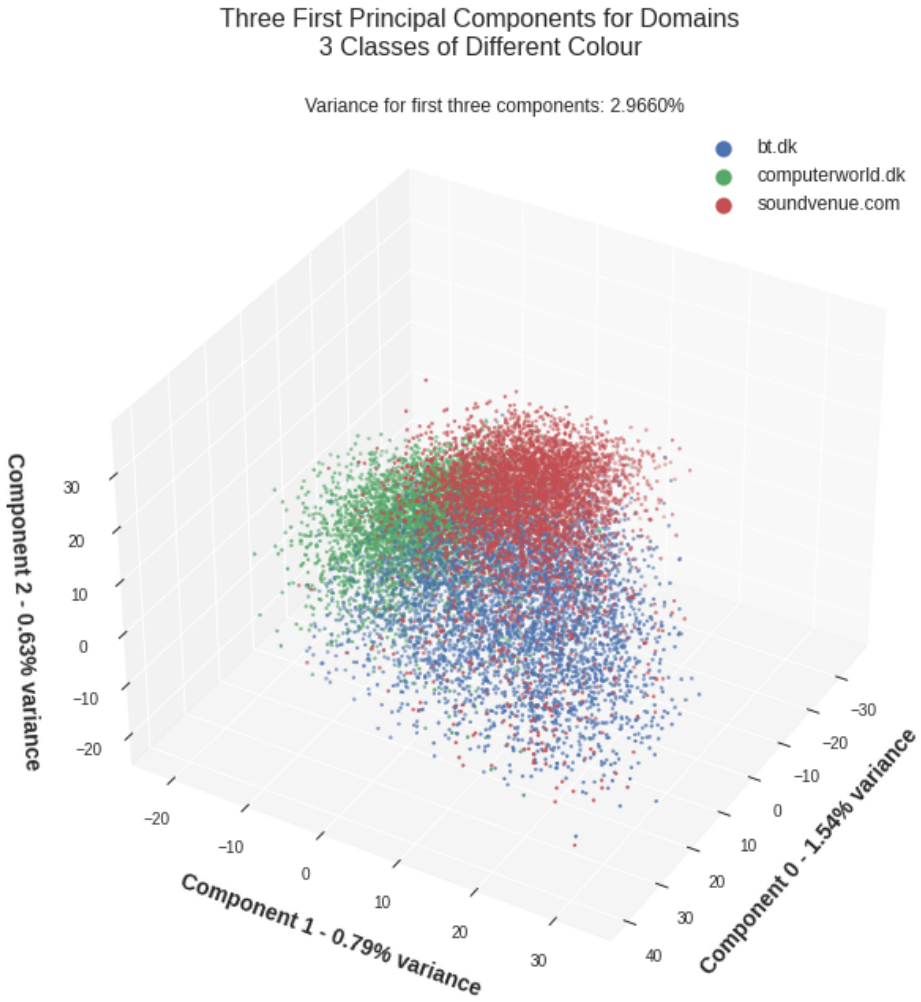


Figure 19: The first three principal components for the extracted features of the domain subset. Each color corresponds to a domain.

While these three classes are separated, they are specifically chosen because they are distinguished from each other in the reduces feature space. With just shy of 3% variance captured by the first components, PCA may not be the best tool of visualizing separation between classes.

5.4.3 *Visualiszing Newspaper Topics*

Each article has 22 topical scores. The difference between domains and their topic coverage has been visualized using the method, t-distributed Stochastic Neighbor Embedding (t-SNE) in Figure 20. This method is based on clustering each point in accordance to its neighbours, assuming they follow a normal distribution. The method is not useful for dimensionality reduction for training, as it will return very different results, depending on the input. However, it is a great tool for visualization. In Figure 20, such a visualization is seen. Note, that no pure mainstream newspapers are included here, as they cover a wide range of topics.

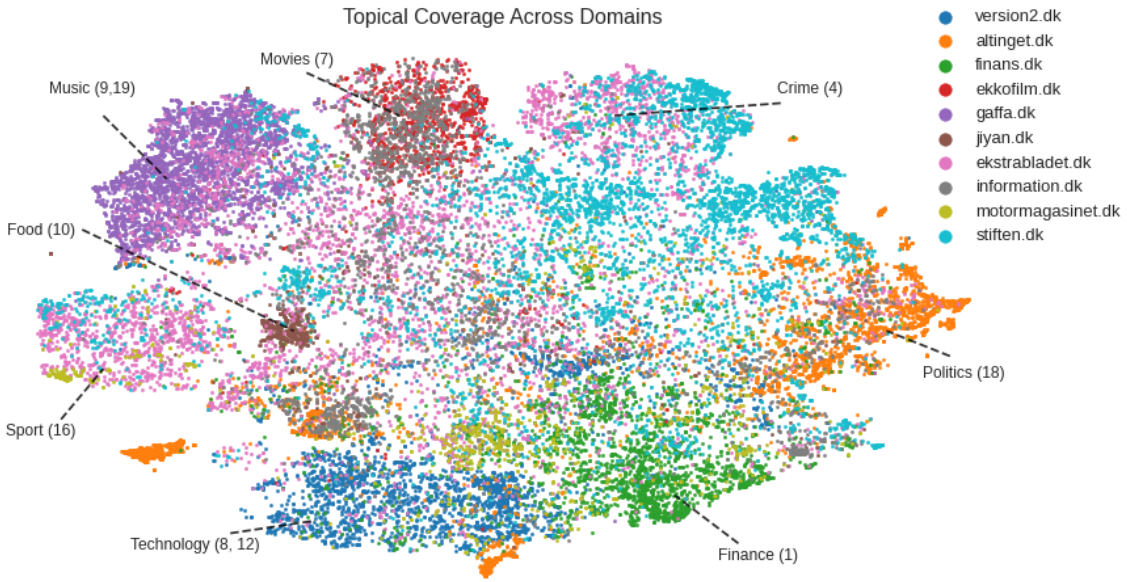
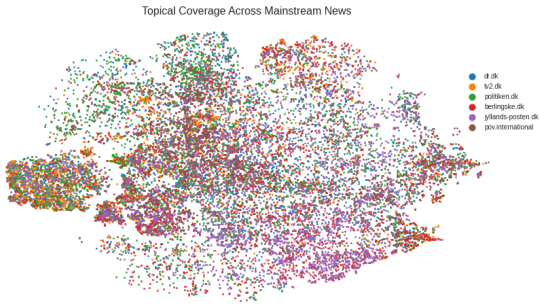


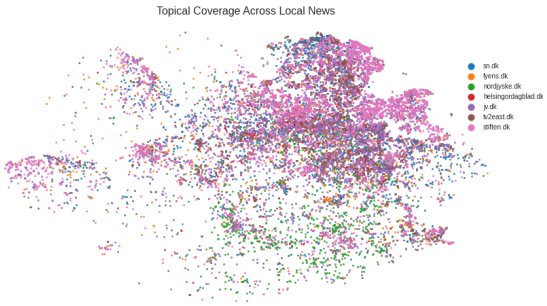
Figure 20: Topic coverage for selected domains. Each dot represents an article colored according to its domain, while the spatial position of each dot is determined by topic of the article. Uni-colored clusters emerge as some domains often write about specific topics.

Because of its varying outputs depending on the inputs, it was not possible to confidently establish the spatial locations of all topics. Otherwise, we could remove one topic and see, which cluster disappeared. However, by knowing which topics certain domains cover (see subsection 3.3.1) and by finding the highest topic score for each domain, some of the most prominent clusters have been specified based on our interpretation, as seen in the plot.

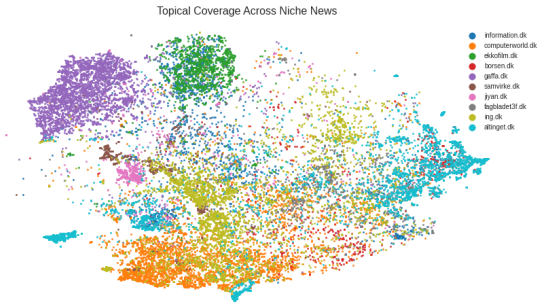
To further gain insight in the topics covered, we made different versions of the plot, containing the four types of newspapers mentioned in subsection 3.3.1 as seen in Figure 21. The local newspapers make up the large chunk of clusters, which were not easily identified in Figure 20. The full-size plots are available in appendix section C.3.



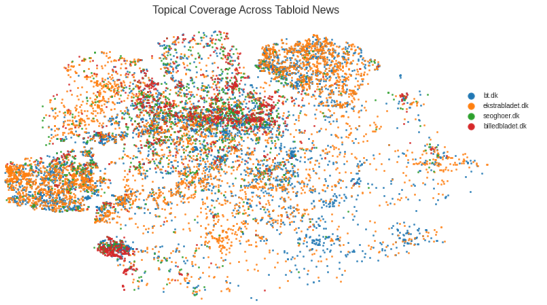
(a) Mainstream News - Broad coverage, difficult to separate. Leftmost chunk is sports.



(b) Local News - Makes up a large part of not easily identifiable clusters.



(c) Niche News - Very clear difference between domains.



(d) Tabloid News - The small red cluster appearing here is topic 0, Family. Especially the magazine, Billed-Bladet writes about this.

Figure 21: Topical coverage of the four different types of newspapers. Local News make up most of the not easily identified clusters.

5.5 CONFUSION MATRICES

A different way to gather insight into the decisions made by a classifier is to create a confusion matrix. It displays to what degree the classifier mixed up two classes. The x axis shows the predicted label and the y axis the actual label. Ideally, the diagonal in the matrix should have high values, as this is where the correct and predicted labels are the same, i.e. true positives. The false positive for each class is indicated by the all additional mis-classifications on the corresponding row. The false negative is indicated by the column. An example of a confusion matrix is seen in Figure 22

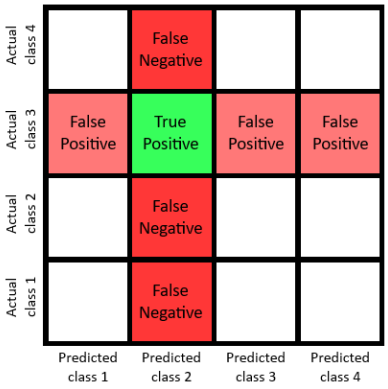


Figure 22: How to read a multiclass confusion matrix. High values at the diagonal is optimal.

5.5.1 *Authorship Attribution*

The confusion matrix in Figure 23 extends the results on authorship attribution presented in the previous sections. Note that all scores that were below 0.05 are presented as 0 as the confusion matrix would otherwise be difficult to read. For some authors, the classification is perfect as indicated by *1*, whereas for other authors, the classification never predicts correctly. The confusion matrix is normalized, evening out the coloring. Otherwise, some authors would naturally appear as better classified than others due to a larger number of articles.

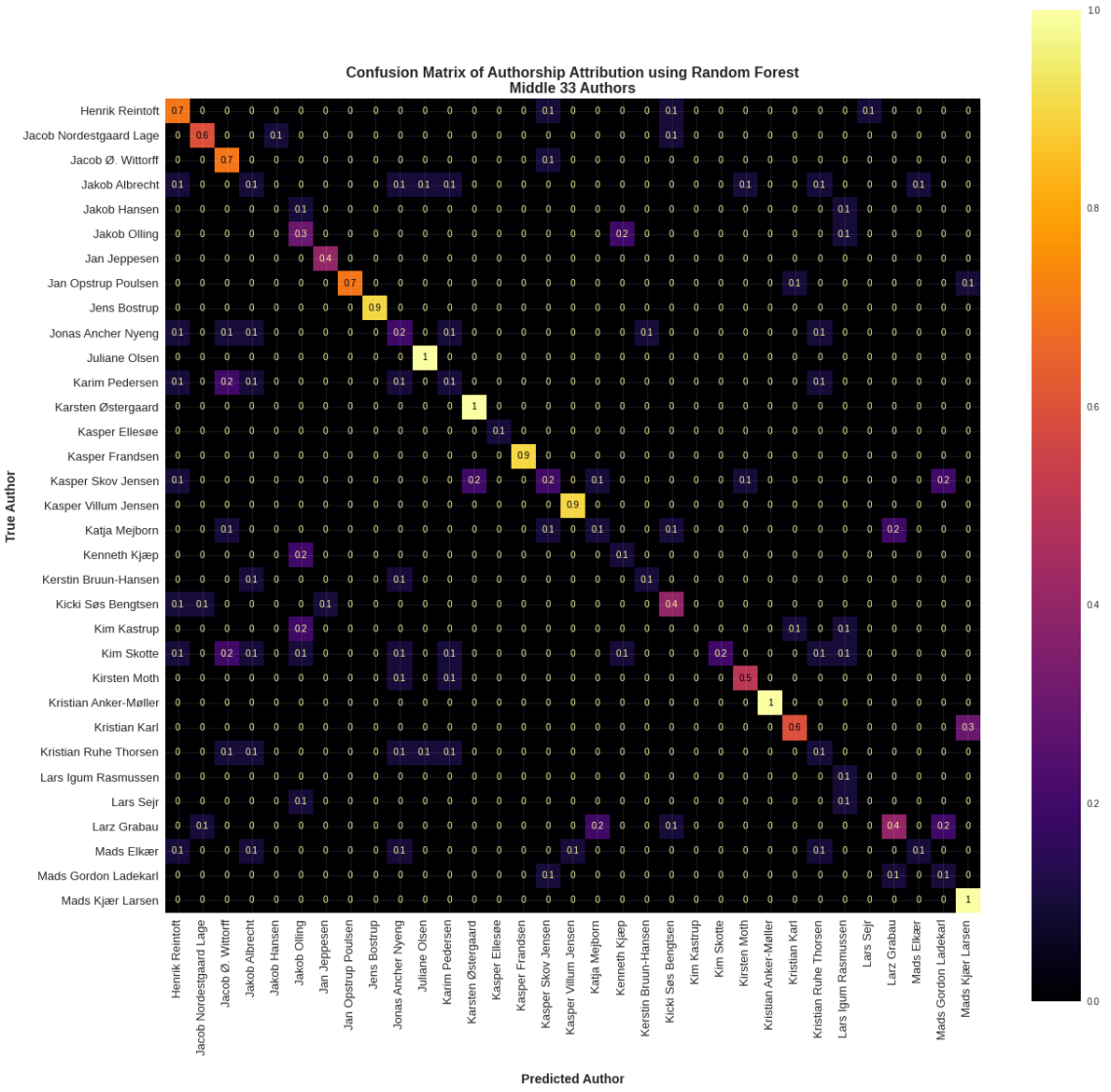


Figure 23: Confusion matrix for authorship attribution on a snippet of the 33rd to 66th authors. The confusion matrix is based on classification from random forest on a normalized scale.

The presented confusion matrix is showing a section of all 100 authors. The confusion matrix using all 100 authors is too big to properly see on paper, so it is available in high resolution at our GitHub pages[39].

39 https://kroglkt.github.io/articlencoding/

5.5.2 *Newspaper Attribution*

Below in Figure 24, the normalized confusion matrix for the newspaper attribution is presented. On each axis, the domain names are labeled and the colored squares are brighter, the more classifications lie in this cell. Available in high resolution online[40].

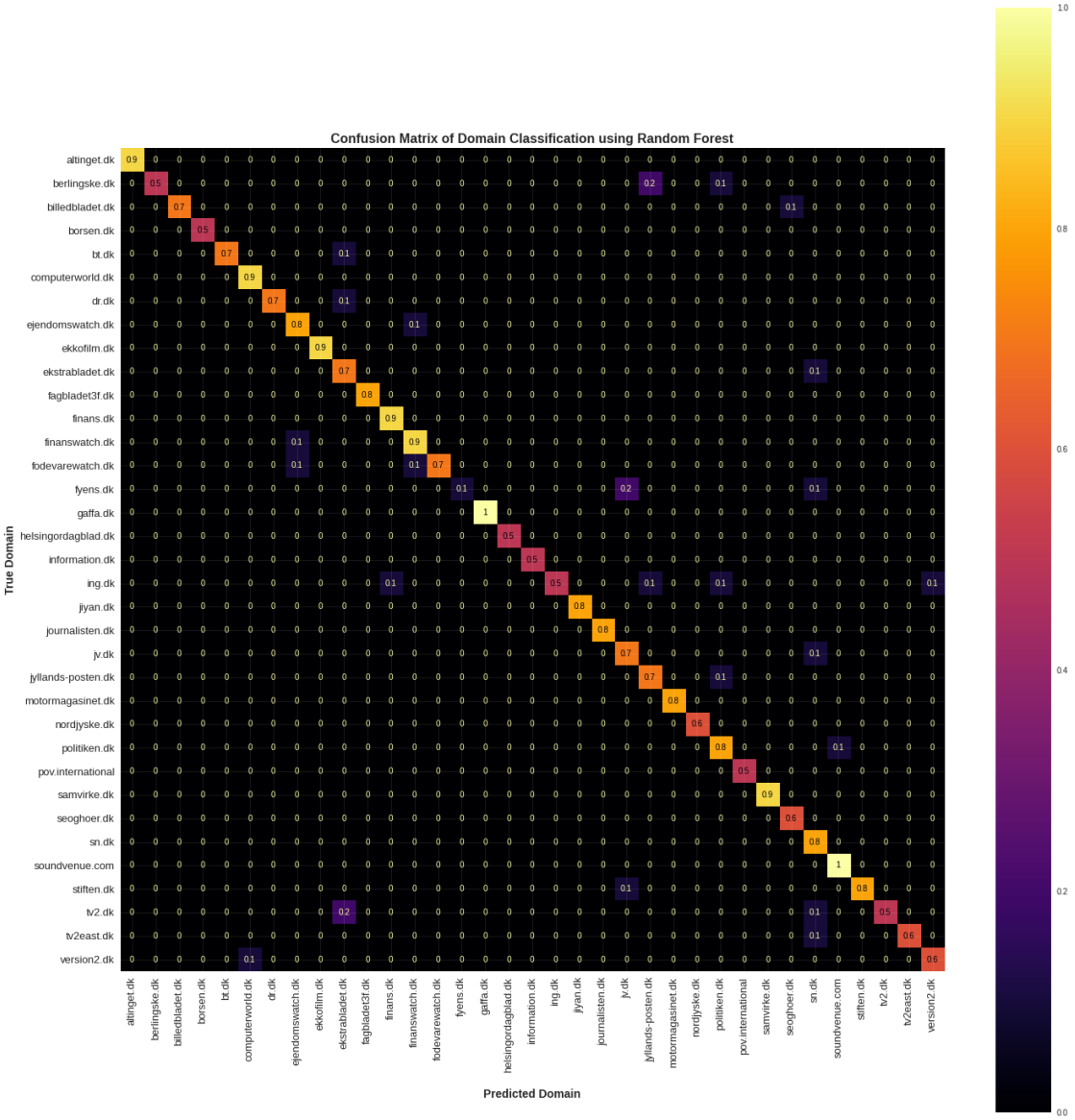


Figure 24: Confusion matrix for newspaper attribution based on classification from random forest on a normalized scale.)

40 https://kroglkt.github.io/articlencoding/

The figure supports the results reported in the previous sections. The classifier has very accurate predictions for most domains, and perfect classification for two: Gaffa and Soundvenue[41]. For Fyens, however, the predictions are rarely accurate. In some cases, it is confused with JydskeVestkysten[42].

5.6 HEADLINE EVALUATION

In subsection 3.1.1, we stated that the headline generation would undergo a in-depth qualitative evaluation. However, this is not possible, as the results provided by the headline generation model using our manual encodings were all the same header: *ny bog om dansk tv-serie er det bedste* ("new book about danish TV-show is the best"). The model trained for over 165 hours (Cloud) but its validation loss never reached a plateau. For good measure, we decided to compute ROUGE1, ROUGE-2, and ROUGE-L anyway and results are presented in Table 15 below.

	ROUGE-1	**ROUGE-2**	**ROUGE-L**
F1	0.036	0.0	0.035
Precision	0.035	0.0	0.034
Recall	0.039	0.0	0.038

Table 15: ROUGE evaluation of predicted headline.

Due to having only one unique string as a predicted headline, we performed the ROUGE evaluation on only 112 headlines. As evident by the score of *0.0* on ROUGE-2, the predicted headline has no sequence above one word that overlaps with any of the 112 headlines in the sample. This would possibly change if we included more headlines but it is unlikely it would make a compelling difference. Further thoughts on this will be presented in subsection 6.1.4.

41 https://soundvenue.com/
42 https://jv.dk/

Summary 5 - Evaluation

In this chapter have presented the results of our evaluation for authorship and newspaper attribution using random forest and logistic regression. Headline generation was evaluated using ROUGE metrics as well as manual evaluation. Initial classification results prompted an additional pre-processing step to avoid bias in the data caused by repeating strings, compromising the newspaper attribution. In the final classification, newspaper attribution still yielded very good results on accuracy, F1, AUC, and top 5 accuracy. The Authorship attribution produced more varied results across the metrics. Though the headline generation trained for an extended period of time, it is unable to produce any sensible headlines.

The key points from the evaluation are:

- Initial classification results for authorship and newspaper attribution prompted a manual inspection of the articles which resulted in an additional pre-processing step and feature extraction anew in an attempt to avoid biased classification (section 5.1)
- Authorship and newspaper attribution were classified with random forest and logistic regression, and evaluated on the metrics accuracy, F1, AUC, and top 5 accuracy. Newspaper attribution achieved good scores across all metrics. The combined encodings for both tasks achieved best results (section 5.2).
- Feature importances were computed with the random forest classifier. Plots show that vector features perform best in both classification tasks, although topics are the most important among scalar features (section 5.3).
- Topics from LDA show promising and relatively coherent topics with respect to the newspaper subset. Visualizations indicate that some newspapers are topic specific whereas others cover a wide range of topics (section 5.4)
- Confusion matrices for authors and newspapers indicate that classifiers confused authors more so than newspapers. For some newspapers, the classification was perfect, i.e. it predicted the newspaper correctly every time (section 5.5). Results are discussed in the next chapter.
- The headline generator using manual encodings was unable to produce any meaningful headlines (section 5.6)

6

DISCUSSION

In this chapter we offer a discussion of the results presented in the previous chapter.
Our discussion is structured according to our research questions posed in chapter 1. We first discuss the implications of our findings in section 6.1, alongside a discussion of potential sources of bias and limitations of our approach. We will reiterate each sub-question, and in subsection 6.1.5 we discuss how the findings regarding the questions each contribute to answering our main research question. Thereafter, we discuss drawbacks and limitations of our methodological approach in section 6.2, as well as limitations of the corpus. Additionally, we discuss feature optimization in section 6.3. Finally, we will suggest improvements on this study in section 6.5.

6.1 IMPLICATIONS OF RESULTS

Due to the focus on interpretability and insights in the results, we decided to deem results which were consistently above baseline as an indication that the methods work. Hence, we did not perform significance tests or any repeated experiments like K-fold. However, for future research on the topic, significance tests could be worth including.

In relation to our first sub-question, *Will manually extracted features, automatically extracted features, or a combination of both generalize best?*, we saw that the combined features resulted in the best or equal performance for both authorship and newspaper attribution, across all four evaluation metrics. This is in accordance with the literature, and may be due to the automatically extracted features capturing elements that our encoding did not and vice versa. As the automatic encodings are based on BERT, they are small in size but likely high in abstraction level. Similarly to multi modality in humans, i.e. having multiple sources of input, like sight and hearing, adding various feature encodings together may prove useful. In similar vein, ensemble classifiers using

different types of models can often outperform single classifiers, as they tend to capture different elements in the data.

Additionally, the logistic regression model outperformed the random forest is many cases, which was surprising. The random forest model often produces convincing results, outperforming other common learning models. Related to random forest, another tree-based model, XGBoost has found great success in various machine learning competitions[43] This indicates, that tree-based ensemble classifiers should be a good choice. However, XGBoost is based on gradient boosting and may provide even better results than what we obtained. While the choice of model was not of focus in this paper, it is still noteworthy, that logistic regression performed this well.

Our second sub-question is *Which of the manually extracted features has the highest impact on the results?*, and in order to answer this we computed feature importances based on the random forest classification.

The 52 extracted feature types are composed of 42 scalar features and 10 vector features. Though there are more scalars, the vector features make up the largest part of the extracted features, due to their lengths. The vectors are responsible for the vast majority of feature importance. This is the case when treating each vector as a single feature, i.e. summing up the feature importances[44]. The best method of obtaining vector feature importances is up for discussion, though using the sum will logically treat each vector as one feature instead of a collection of features. Note also that the vector importances (Figure 17a) for both classification tasks generally follow the same trend, with just a few minor differences. Of the vector features, the character bigrams and character trigrams were the best features for both authorship and newspaper attribution. As mentioned in section 2.5, character n-grams are commonly used to capture lexical patterns because they are resilient to noise such as typos and spelling errors. Furthermore, since they are character-based, they are unlikely to be biased by content. Our findings therefore confirm previous findings [92, 73] in the versatility of character n-grams as text features. In the interest of time, we did not examine the most common character bi- and trigrams, but future studies are encouraged to do so as the authorship attribution benefited most from character bigrams but less from character trigrams compared to newspaper attribution. Insights into why that might be would therefore be valuable in order to better assess the adaptability of different features. The feature importance plots suggest, in line with the findings by Sari et al. [82], that authorship and newspaper attribution are in fact two different tasks, i.e. authorship style is detectable even through potential editing.

43 XGBoost Machine learning challenge (visited May 28th).

44 Had the weights been averaged, the importances would look different as low weights would drag the average (and thereby importance) down a lot.

The third most important feature was POS trigrams which indicates that syntax level features are important to include. POS n-grams also make for scalable features as they merely capture structural information which relates to the stylistics of text rather than the content. On a note of content though, the fourth best feature was word unigrams which can be considered a content feature in that it captures information on a lexical level. As newspapers vary in topical focus, it was not surprising that a content feature would have high importance, and it confirms previous findings [82] of content features being valuable for newspaper corpora. These findings suggest that some of the features previously extracted from English corpora do in fact scale to other languages, Danish in our case. English and Danish, however, belong to the same language family, so investigating character-based and syntactical features in other languages would be worthwhile, especially languages where the orthographical representation (written language) is different. We also wish to note that the fifth and sixth best features were the skip bigrams where k=*2* and k=*3*. This finding is worthwhile as it establishes value to the use of word skipgrams as a feature, and we encourage future studies of similar tasks to include and extend the use of skipgrams to further showcase how they can be utilized.

The plot of feature importances showed that topics were, unexpectedly, not as important when compared to vector features. It is worth noting, however, that the topics were represented as scalars when computing importance and also in order to plot each article according to the assigned topic. In preliminary computations of feature importance, we treated topics as a vector by summing the importance weights for each topic, but the summed importance was still not at level with vector importances and we therefore decided to represent the importance per topic for better interpretability. The reason, their collective importance is lower than the n-grams are, like seen in Figure 5.3, due to the low number of elements in the vector.

In the next subsection, we review the use of topics as we discuss our evaluation for newspaper attribution. We do so in order to answer our third sub-question: *How do the newspapers relate to each other? Does our feature representation reflect real world similarities between newspapers?*.

6.1.1 *Newspaper Attribution*

As presented in section 5.2 and section 5.5 in the previous chapter, the results of newspaper attribution were very promising across all four metrics, whereas authorship attribution produced more varied results. It is worth reiterating that the author subset had 100 classes whereas the domain subset had only 35 which will therefore naturally be an easier classification. Considered

naively, the results of newspaper attribution would indicate that our manual encodings worked very well on domains, i.e. each newspaper was easily discernible based on the features we chose to extract. However, for a 35-class classification we were surprised that the AUC score was *0.99*, a near perfect measure of separability. Furthermore, the confusion matrices for domains indicate perfect classification for some domains (Gaffa and Soundvenue) which was surprising as the two domains cover similar topics (music), so we initially expected them to be confused for one another. Upon further inspection of the two domains, we found that, despite the additional pre-processing of removing the first and last 100 characters, Gaffa still had a lot of junk text. The inspected articles had listings of events with dates in the end. These listings were far longer than 100 characters and all were therefore not removed in pre-processing. The event dates were found in 92% of the Gaffa articles which likely contributed to perfect classification for Gaffa. Although, it is also worth remembering that Gaffa is a niche newspaper, as it is highly directed towards topics related to music. In the feature importance figures (section 5.3), we saw that topics did not have high impact on results relative to vector features. However, the vector feature for word unigrams had high impact on classification, perhaps because it would have captured topic specific words. On this note, the same might apply for the domain Soundvenue. We inspected the domain and found that *Læs også* appears in 34% of the articles. However, this phrase is common in web articles, e.g. it also appeared in 52% of DR's articles, and 23% of articles from Version2. Upon further inspection, it was clear that Soundvenue also has a lot of characteristics that could have been caught by the word unigrams. For example, the articles list movies, songs, and albums in quotes in their original language (often English). Therefore articles might contain a high frequency of unigrams containing English words and words related to the musical scene. Additionally, character n-grams with single quotes (the symbols ‘ and ’) would be frequent. Alike Gaffa, Soundvenue is a highly specialized newspaper focusing on music, movies, and TV shows, so it is not unlikely that topic specific words helped the classification, and topic was inadvertently caught by the word unigrams. It is therefore probable that the classification was not based on data leakage only. When we consider the other domains, the ones that scored 0.9 on the normalized scale in the confusion matrix were also topic specific newspapers, e.g. Finans[45] (economy/finance), Samvirke[46] (consumption), Altinget (politics), Computerworld (IT). This gives reason to believe that the niche newspapers benefit from the features that capture content which is supported by the fact that word unigrams is the fourth best feature.

45 https://finans.dk/
46 https://samvirke.dk/

In relation to Gaffa and Soundvenue, the domain Ekko Film[47] (henceforth Ekko), which also covers movies, was correctly predicted in most cases (90%). It has some junk, also in the form of *Læs også* which appeared in 29% of the articles. We also found that *Ekko* appeared in almost 50% of Ekko's articles. Our pre-processing step that masked the domain names in the articles was based on the name of the domain as it appeared before the website suffixes, i.e. *gaffa* for *gaffa.dk*. For some domains, e.g. Ekstra Bladet, we had a step that ensured articles from *ekstrabladet.dk* would catch *Ekstra Bladet* but due to constraints on time, we were not able to manually go through every domain to identify the various ways the domain name appeared. On a related note, as mentioned in subsection 3.4.5 we masked the domain and author names if they appeared in the articles. We used a regular expression for this step, but as we sifted through some articles after classification, we noticed that the regex would also catch, for example, *bl.a.* and mask it due to the punctuation immediately preceded and followed by characters. Again, owing to time constraints we decided to leave this as is rather than pre-processing and extracting anew because this error would affect all authors and domains equally.

Considering the accurate predictions of the niche newspapers, even the ones similar in topic, it seems they are easily seperated in feature space. As we mentioned in section 2.3, one study used the Jason-Shannon Divergence measure to compute topical similarity between newspapers. In light of our findings, this measure would be interesting to explore as it can quantify how similar in topic coverage the niche newspapers are. Our pre-existing knowledge of the newspapers is merely based on surface level observation, not in-depth inspection or analysis of each domain. A quantification of topical similarity would therefore provide deeper insight into this and perhaps lay ground for future expectations.

However, not all domains were predicted as accurately as the niche newspapers. If we again consider the confusion matrix, a single newspaper, Fyens (local newspaper), scored very low on the scale with only being correctly classified 10% of the time. It was more often classified as Jydske Vestkysten (jv.dk), another local newspaper. Furthermore, local newspapers are not restricted by topics covered, but rather a geographical area, i.e. the local newspapers may be alike mainstream news in terms of topics covered.

We also noted that domains such as Berlingske and TV2[48] were only correctly predicted 50% of the time. In light of these findings, it seems newspapers of the same type, i.e. local and mainstream newspapers, are more easily confused which confirms our expectations with respect to local and mainstream newspapers being similarly represented in feature space. The visualization presented in

47 https://ekkofilm.dk/

48 https://tv2.dk

section 5.4 also confirm that newspapers of the same category will resemble each other in feature space as they are a scattered across the plot indicating that their articles all cover the same wide range of topics.
As stated in section 2.3, Albakry [2] controlled for similar popularity of the newspapers of interest. As our corpus was quite varied in type of newspapers, their popularity[49][50], and target readers, we suspect that newspapers with mainstream and local news coverage tend to appeal to the same readers and may thus be more inclined to use similar language.

In summary, these findings suggest that niche newspapers, although revolving around the same topics, are not represented similarly in feature space. In the future, it would therefore be worth augmenting expectations with in-depth analyses of the domains in order to better determine how related newspapers such as Gaffa, Soundvenue, and Ekko actually are beyond the topics they cover. Additionally, we saw that correct prediction of mainstream news scored 0.5 which is in line with our expectations. The local newspaper, Fyens, was very poorly predicted and often mis-classified as Jydske Vestkysten, another local newspaper. It therefore seems that local and mainstream newspaper are represented similarly in feature space, and the text encoding then does reflect real world similarities in these cases. However, there may be other features that would contribute to better separability, which we discuss in section 6.3.
In the upcoming section, we present a validation test we did to support our findings for newspaper attribution.

6.1.1.1 *Validation Test of Domains*

In order to validate our findings and investigate our suspicion of data leakages, we performed a final test for the random forest classifier. We manually copied the article contents from 47 articles across 11 domains, Soundvenue, Gaffa, and Ekko included. With this test we were able to gain better insight into the classification, and better know whether the good results are due to data leaks or if the domains were in fact correctly identified due to differences in writing style and content. The manually retrieved article contents are free from junk text and are never seen by the models before. The domains included and the number of articles for each is seen in Table 16.

49 Weekly readers for daily newspapers, 2021 (visited May 23rd)
50 Weekly readers for weekly newspapers and magazines, 2021 (visited May 23rd)

Domain	# Articles
Ekstra Bladet	5
Gaffa	4
Soundvenue	4
Ekkofilm	3
DR	4
Se og Hør	5
Stiften	4
SN	4
Computerworld	4
Samvirke	6
Altinget	4

Table 16: The 47 articles manually collected across 11 domains.

As the feature extraction process of each article is dependent of the corpus, i.e. TF-IDF scores rely on the entire collection of documents, we used the TF-IDF vectorizers which were fitted on the domain training data to vectorize these new articles. In classification of the new, manually extracted articles, the random forest obtained the accuracy of *0.49*. Its confusion matrix is seen in Figure 25. The values are not normalized as the previous confusion matrices due to the small number of articles. Note, that Stiften[51] is a local newspaper.

51 https://stiften.dk/

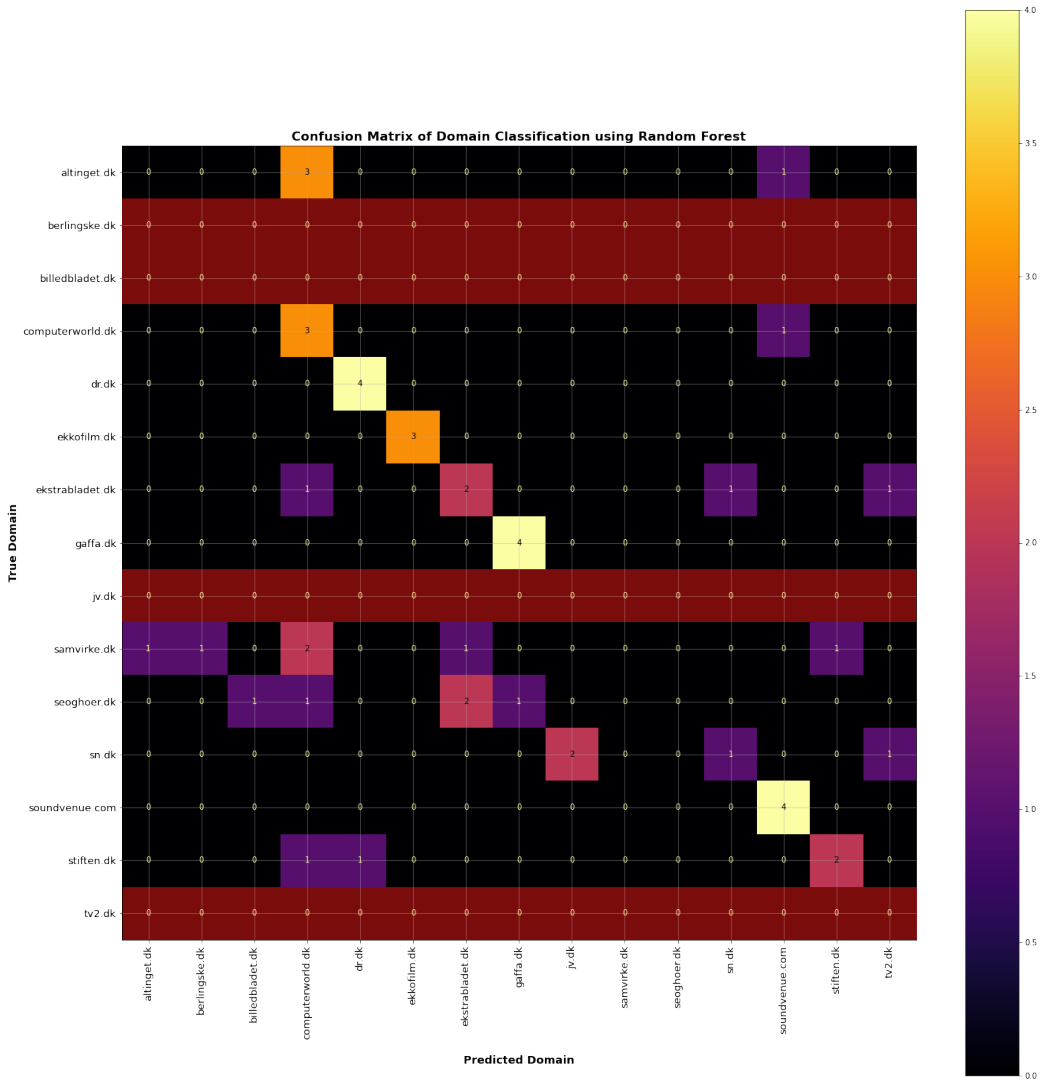


Figure 25: Final test for domain classification using 47 manually extracted articles from various newspapers. The domains Billed-Bladet, Berlingske, JydskeVestkysten and TV2 were predicted by the model, but not present in the sampled articles and are indicated by the red rows.

One of the main reasons for performing this test, was to reveal whether the high accuracies on Soundvenue and Gaffa were due to junk in the articles or if the manual encoding was in fact generalizing well. While this test is limited in scale and nowhere near the previous test in terms of article quantity, it is remarkable, that both Gaffa and Soundvenue had all of their articles correctly attributed. The same applies for Ekko, while none of the articles from Altinget, Samvirke and Se og Hør were correctly identified. The fact, that three of the four articles from Altinget were classified as Computerworld is unexpected.

The new articles from Altinget were revolving around four different subjects: The implications of Turkey's stance on Sweden and Finland joining NATO, western countries preparing for economic

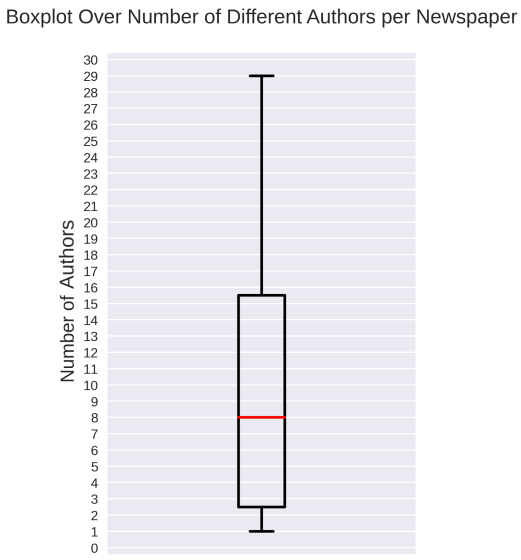
pressure from China, things to consider for an upcoming Danish EU referendum, and a new book about smartphones. The latter could fit in Computerworld, but there is no clear relation to Computerworld in the other articles. However, as mentioned, this test is to back up the results obtained from the previous and the classifications for Altinget may have been a fluke.

6.1.2 *Authorship Attribution*

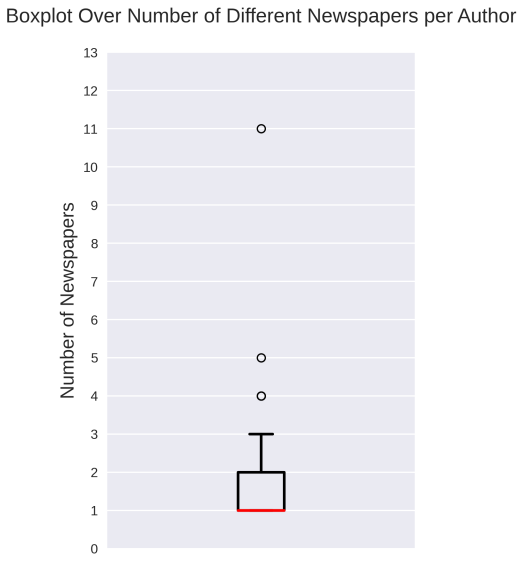
In the section above, we discussed the findings with respect to newspaper attribution, and noted that content features may have played an important role in the classification. For authorship attribution it can be beneficial to avoid features related to topic or content in order to make a model scale across genres (e.g. news, novels, poetry), but in author attribution in a newspaper corpus this may not be the case (as per mentioned in section 2.2) because journalistic authors are more specialized with respect to topics. The attribution of authors was not as accurate as the newspaper attribution. This is likely due to the fact that there were 100 classes compared to the newspapers' 35 classes. As mentioned in section 2.2, our authorship attribution task was under the category, many-candidates problem, as the candidate count was above thousand. However, this is no longer the case with the 100 class authorship subset. The AUC score for authorship attribution was still *0.85* for the manual encodings which is well above chance, showing that some features were in fact beneficial for attributing authors.
However, for some of the authors, as seen in the confusion matrix (Figure 23), the classification was perfect. This gave reason for further inspection.

6.1.3 *Additional Classification Data Insights*

Upon seeing the results, we were curious as to how authors were distributed across domains and vice versa. The reason for dividing the data into subsets was to be able to separate the class values, authors and domains. However, after obtaining the results, it was not explicit how these authors and domains differed from each other. Ultimately, newspapers and their journalists are closely related. In addition to the interpretations and visualizations done so far, an obvious step is to investigate the number of authors for each newspaper and the number of newspapers for each author. These distributions are seen in Figure 6.1.3, with each boxplot being performed on its respective data subset.



(a) Authors per domain, median value at 8 authors. Outliers removed (Politiken has the most authors: 575) Performed on the domain subset.



(b) Domains per author, median value is 1. The author writing for most different newspapers is Sacha Sennov with 11 different newspapers. Performed on the author subset.

Politiken is the domain with the most authors. According to an article from Journalistforbundet[52], freelancers write 29% of all articles in Politken. Such authors would naturally also write for many newspapers, whereas permanently employed journalists would stick to a single newspaper or a few newspapers.

Investigating the number of authors per domain and vice versa may reveal something about how much these two tasks differ. 100% of articles from Gaffa and Soundvenue were correctly attributed and Fyens.dk only $\approx$ 10%. Perhaps Fyens has a larger quantity of authors, making the task more difficult. Below in Table 17 the author count are shown for a collection of notable domains and also domain count for notable authors[53].

52 https://journalistforbundet.dk/freelancegruppen/nyhed/freelancere-skriver-mere-end-en-fjerdedel-af-politiken

53 The scores in the table are seen in the confusion matrices available in high resolution online

Domain	Score	# Authors	Notes
Soundvenue	1.0	14	The top authors has only written for Soundvenue
Gaffa	1.0	16	Only two top authors present in author dataset. Second author has also written for Politiken.
Fyens	0.1	9	Top authors has written for multiple newspapers
Ing	0.5	15	Three top authors has only written for Ing
Børsen	0.5	5	Smallest domain. None of their authors are present in the author subset
Politken	0.8	575	The newspaper with the most unique authors
Fødevarewatch	0.7	1	Only author is Sacha Sennov, which also happens to be the author writing for most newspapers

Author	Score	# Domains	Notes
Juliane Olsen	1.0	1	The only writer for Billedbladet, though this domain has a score of 0.7
Erik Holstein	0.9	1	Only written for Altinget, which has a score of 0.9
Karsten Østergaard	0.9	3	77 of his articles were for Fyens
Allan Bauer	0.0	2	1302 articles for Motormagasinet which has a score of 0.8
Morten Ravn	0.0	1	Has only written for Stiften, which has a score of 0.8
Sacha Sennov	0.1	11	Is the author who have written for the most newspapers.
Nicolai Devantier	0.2	1	Has only written for Computerworld, which has a score of 0.9

Table 17: The number of authors for different domains and number of domains for different authors. It seems, there is no clear trend in how these numbers are distributed.

From the table, we can see that some domains are made up entirely from one author and some authors are made up entirely of one domain. However, this does not seem to directly map between the authorship and newspaper attribution scores. Some authors have only one domain and a bad score, but the domain in question has a good score. Additionally, the scores for the domains do not seem directly related to the number of authors, which is good. Were that the case, one could interpret it as newspaper attribution just being a collection of authorship attributions. The purpose is, of course to have a general idea of how the general writing style and contents are for the domains. On a side note, when splitting the domain data into training and test data, we stratified it after domain. While this is fine, it does not take into account, that some authors may be present in the test set but not in the training set. Perhaps the poorly scored domains have had a bad split of authors.

With more time, we would experiment with the data subsets to ensure that the number of classes across tasks (i.e. 35 classes against 100) and distribution of authors per domain and vice versa is controlled for in order to make the classifications as fairly comparable as possible. However, the entire focus of the project was to explore the possibility of a scalable text encoding, so ideally the number of classes should not matter.

6.1.4 *Headline Generation*

In the sections above, we discussed the evaluation of our classifications to answer our first three sub-questions. Here, we discuss the evaluation of headline generation when using manual text encodings.

As explained in subsection 4.8.4, we did not succeed in generating headlines from automatically extracted features or a combination of manual and automatic. While the manually extracted features were the essential and novel part for headline generation, combining the two for classification provided better results. This might also have been the case for headline generations, as suggested by the literature review.

We evaluated the headline predictions from our manual encoding using ROUGE which, unsurprisingly, revealed very low F1 scores of X and Y for ROUGE-1 and ROUGE-L respectively. Considering that only a single unique headline was predicted for all targets, we make no attempts at interpreting the quality of the headline prediction.

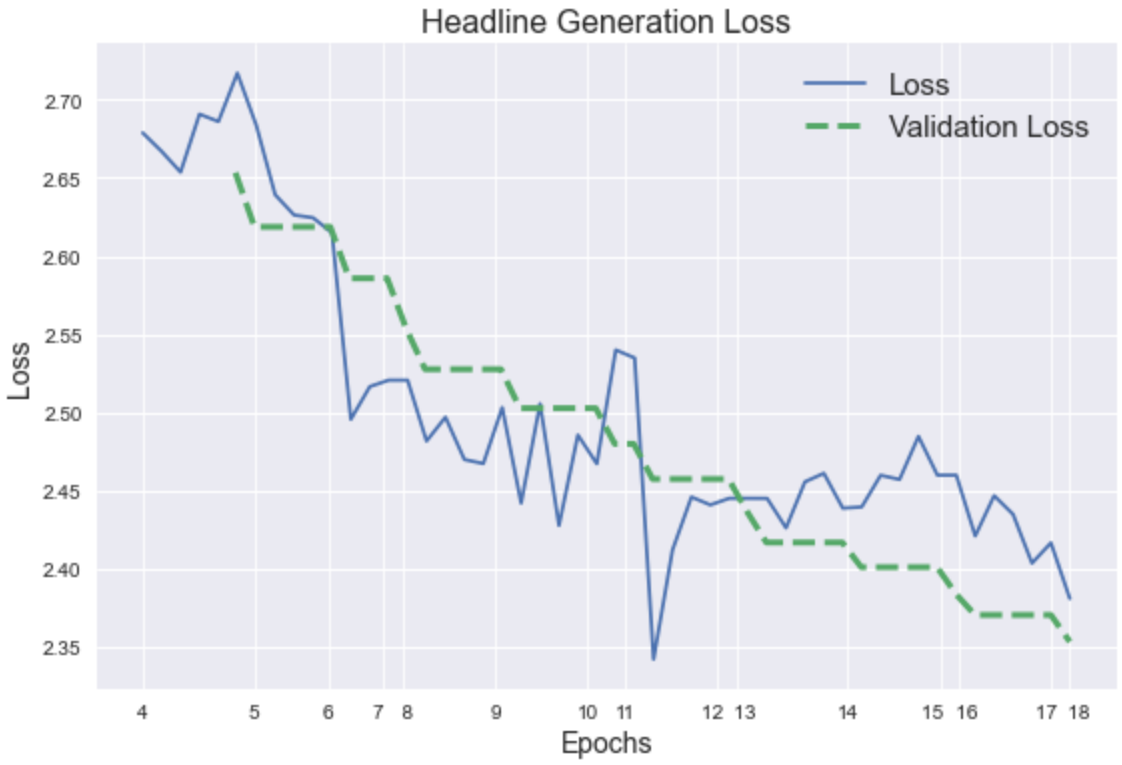


Figure 27: The loss falling over time along with validation loss. The epochs in the plot are uneven due to logging error.

However, as seen in Figure 27, the validation loss during training never went to a standstill. This gives reason for continuing training the model. From prior training sessions using the same model (section 4.8), the same behavior was seen where the model would produce the same headline for all articles before beginning to change them. Based on these observations, we intend to continue training the model beyond the thesis deadline with hopes that with enough training time, the model will be able to produce headlines of comparable quality as the examples showcased in section 4.8.

This unfortunately means, that our fourth and last sub-question, *How will a manual encoding perform for headline generation?*, goes unanswered as we deem the results obtained non-final. Once the validation loss will no longer fall, we can once again attempt answering this question.

6.1.5 *Answering the Research Question*

In the sections above, we have discussed the implications of our findings with respect to our four sub-questions which all contribute to answering our main research question: ***How can we create a general purpose feature encoding for authorship attribution, newspaper attribution, and headline generation?***

In answering our first sub-question[54], we found that an encoding combined of manually and automatically extracted features generalizes best. Answering our second sub-question[55] led us establish that use of character-based features generalize best, likely due to their resilience to noise and lack of content bias. Content features, however, also performed well for our specific dataset. Additionally, we found that syntax-based features were favorable. Our third sub-question[56] concerned the different newspapers' relation to each other in feature space, and we found that niche newspapers are separated in feature space, despite sometimes being similar in topical coverage. Local and mainstream news on the other hand seemed more similar in feature space which lines up with their similarity in news coverage. The results of manual encodings for headline generation was inconclusive with the given training time, and we intend continue training the model. We therefore address our main research question primarily in light of our classification findings.

In answer to our main research question, we can therefore suggest that in order to create a general purpose encoding, it is beneficial to use manual and automatic encodings combined, with some consideration for the task. Manual features should include features that are character-based and some that capture syntactical elements of the text. However, more research is needed to better assert if manual encodings can scale across the type of tasks, e.g. classification and generation, as our findings for headline generation are inconclusive. The model did not finish its training in time, and we will therefore continue training it, hopefully achieving usable headline predictions.

54 *Will manually extracted features, automatically extracted features, or a combination of both generalize best?*

55 *Which of the manually extracted features has the highest impact on the results?*

56 *How do the newspapers relate to each other? Does our feature representation reflect real world similarities between newspapers?*

With the above in mind, it would be worth exploring the limitations to scalability, i.e. whether encodings can be scalable across the different number of classes, across genres, across languages, etc. We acknowledge that more research is needed in order to determine how well the general encoding scales to other genres and languages, but we consider these findings promising as they give insight into which features perform successfully across tasks. It has further provided us with knowledge of features on which newspapers tend to differ. This lays ground for a lot of further research, discussion, and experimenting within scalability across tasks, corpora, and languages.

The preceding sections of this chapter have revolved around discussing the implications of our evaluation, as well as augmenting the findings with further validation tests and data insights. We now move on to discussing possible limitations of our approach and things we have reconsidered since implementation.

6.2 METHODOLOGICAL LIMITATIONS AND RE-CONSIDERATIONS

Though all decisions taken in the methodology were carefully taken, some considerations emerged after the results were obtained.

6.2.1 *Challenging the Decision of 22 Topics*

When deciding on a number of topics the LDA model should find, we settled on 22 topics as a result of a single run of a coherence model. However, we later attempted this process again, as the results may vary between each run, because the process is stochastic. We computed coherence from 8-50 topics four times to see how well 22 topics performed.

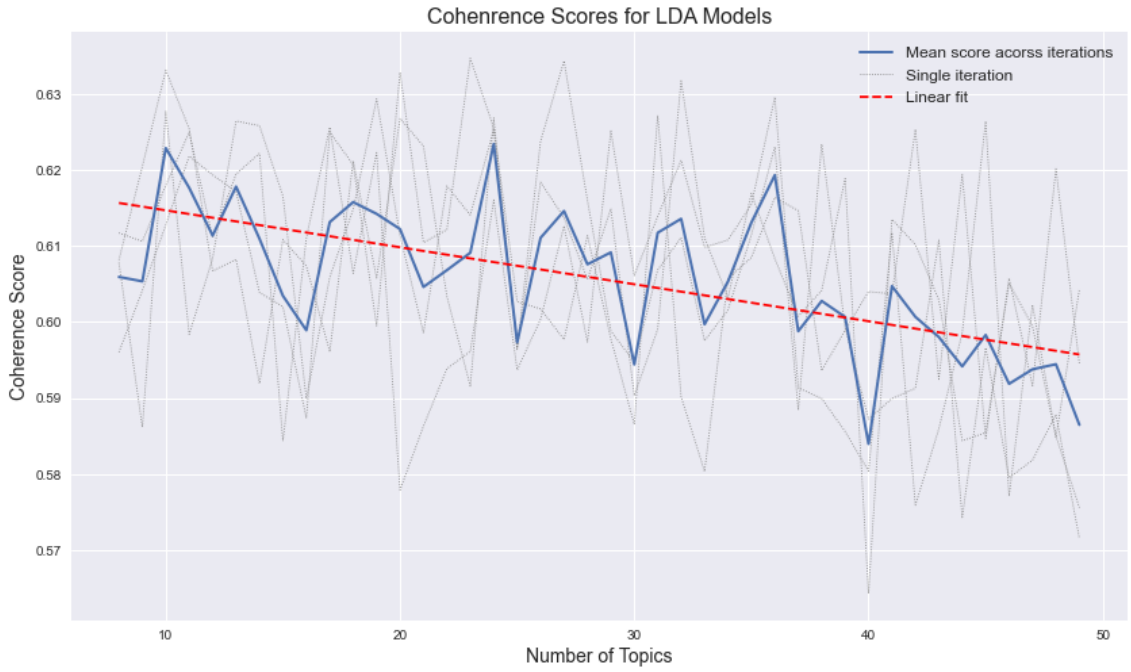


Figure 28: The coherence scores for 8-50 topics, repeating four times.

The coherences from this process are seen in Figure 28 and 22 topics are no longer the dominant topic count in terms of coherence. In fact, it seems 24 topics are the best choice with a coherence of 0.62, compared with 0.60 coherence for 22 topics. The blue line in the figure is the mean value across all runs and the red dotted line is a linear regression fitted on the mean (downward slope of -0.004). Generally, the coherence falls as number of topics increase and whether the consistently good score at 24 topics is a coincidence would require the process to be repeated several more times, which is not feasible with a training time for each 8-50 iteration counting at 71 hours (PC). Of course, coherence is a method of quantitatively assess the number of topics needed and from our qualitative evaluation of the topics found using 22 topics, this finding may not be critical.

6.2.2 *Subset Class Thresholds*

The upper limit for article counts in the overrepresented classes in the domain and author subsets was set to the third quartile of the distribution. This limit was somewhat arbitrary, and the mean and median was also contesting. However, we went with the chosen value of third quartile and decided to redo the limit, were the results biased towards article count.

Testing for a linear correlation using Pearson's R on the confusion matrix scores and article counts per domain yields insignificant results: *0.19 correlation with 0.27 p-value*. The same process was applied to authors and yielded a correlation of *0.04 with 0.67 p-value*. Neither values have high

correlation nor a p-value below 0.05, meaning that there is no linear correlation between article count and classification score.

6.2.3 *Dataset Drawbacks*

As the data we worked with is real world data, it is imperfect. It went through a thorough data cleaning phase in some areas perhaps too thorough. After obtaining the results, a few complications with the dataset came apparent.
As mentioned in chapter 3 we took several steps to pre-process and clean the dataset to avoid biases. We used various measures of article lengths using sentences and words as a feature for the encoding, however, we note that some articles were shortened due to a paywall. We decided to overlook this as we were in part bound by the data that we had available as we did not scrape the articles ourselves (except for three domains) and thus did not have control of the quality of the article bodies. As a few paragraphs were still included before the article was cut off, we tentatively included it under the assumption that stylometric features would still be present and could help discern the author or domain. However, future works would benefit from either excluding incomplete articles or excluding features related to article length. Excluding the articles entirely is arguably to be preferred as more or less all features are inadvertently affected by the article being cut short. On the other hand, the worst performing features were sentence and word lengths so we do not deem it a critical drawback.

6.2.4 *Cleaning Up Headlines*

We spent a lot of time making sure, that no revealing information was included in the headlines, i.e. no author and domain names. Many headlines were formatted such that their domain name was included at the end of the headline *(Ny forskning viser at danske hajer ikke er glade | BT.dk)*. This decision was later challenged when we noticed that some of the domain names were not properly removed, as they have different ways of spelling, i.e. Billedbladet, Billed Bladet, Billed-Bladet. In hindsight, this information could have been left as is, as the headline is not used for classification of neither domains or authors and can in fact be seen as a characteristic of how headlines are written. Were the headline generator to write the correct domain name in the headline, it would make for interesting discussion, as it would have created a domain classifier on its own, though this would not be its intended purpose.

6.2.5 *Removing Unique Entries*

The decision of removing unique domains and authors from the data was made prior to the decision of splitting the data into subsets. The initial approach was to perform all three tasks using the exact same entries, hence they would all need the same pre-processing steps. However, with the recent approach, the removal of unique entries should have been tailored to each subset, which it was not. Having unique or multiple authors would not be an issue to include in the domain dataset, but rather create a better representation of the domain, as it would introduce more varied articles. Similarly, removing unique domains in the authorship subset should not have been performed, as the focus lies on the author, rather than which newspaper they write for. In the headline subset neither of the processes were necessary. While the removal of these articles was not necessary, it was still an affordable process in terms of data quantity. Therefore, this decision is not of critical importance, though it ideally should have been different.

As we have discussed, there are several aspects of our methodological approach that could have been different, and that should be considered in future studies. In the next section, we will discuss which features could have improved our manual encoding so that future studies might benefit from it.

6.3 FEATURE OPTIMIZATION

Although our focus has been on interpretability of results rather than optimization of features, there are several ways in which the manual encoding could be improved.

As mentioned in chapter 3 we excluded function phrases to limit the time spent on pre-processing, however, future studies would benefit from including it as a way to optimize the encoding. As function words have been shown to be a strong indicator for stylistic preferences it is relevant to compare the use of, e.g. *mht.* and *med hensyn til* as abbreviations or lack thereof tend to be used consistently and can thus be a very good indicator of the author or domain.

In chapter 2 we mentioned one study that considered differences between press release and news reports, in which it was noted that symbols and numbers often were converted to written words. Frequency of symbols and numbers was ultimately not included in our manually extracted feature, however, considering the high frequency of quotation marks we observed in some domains, this could potentially have improved the classification further. Although it is worth noting that these

observations are from a news corpus, and we are therefore unsure of how well such a feature would scale to other datasets.

Our feature importances showed that syntactical features might be useful to include. They are good for scalability as they do not get biased from content. We used features such as POS n-grams and skipgrams to include syntax level features. These two features capture surface level structures. One way to optimize syntactical knowledge of the texts is to make use of syntactic n-grams which allow for more extensive insight in structures beyond the surface. Syntactic n-grams, unlike n-grams of words, characters, and POS tags, are n-grams constructed through the path of a syntax tree rather than how they appear on the surface [87].

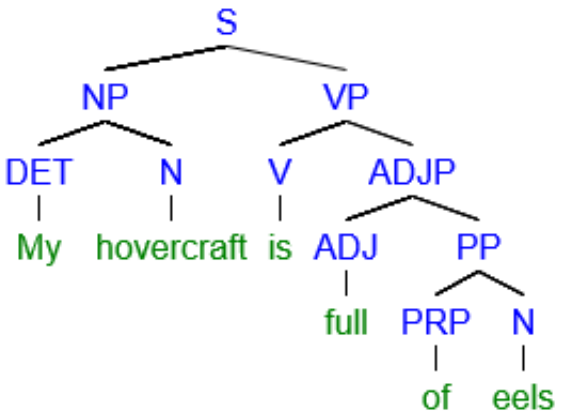


Figure 29: An example of a constituency-based syntax tree.

As seen in Figure 29 above, sentences can be analyzed and divided into constituents or phrases, i.e. collocations of words that collectively form an element in the sentence.

The feature importance plots also showed that word skipgrams are beneficial to include. The encoding could be expanded by including skipgrams of characters and POS tags. It is probable that an inclusion of skipgrams of characters and POS tags would enrich the encoding substantially. In similar vein, using word2vec embeddings may have been a strong feature to include.

As indicated by our newspaper attribution, the niche newspapers were among the ones that were easiest for the random forest classifier to predict correctly, likely due to their specialized content as discussed previously. Another way to benefit from content would be to use named entities as a feature. Some of the niche newspapers and tabloids are likely more prone to discuss entities, e.g. specific companies in relation to the stock market, and celebrities or royalty. The frequency of

named entities could then be a strong indicator of domain.

Upon seeing the topics from LDA and manually inspecting some domains, we saw that some newspaper, e.g. Soundvenue or Gaffa, use English a lot in their mention of movies, albums, and songs. With this in mind, it would be interesting to include a feature that, for example, counted English words as a high frequency of such could be very domain specific. However, this of course would not be of use, were the corpus to be English.

6.4 ADDITIONAL DATA ANALYSIS

Every time a result has been obtained, questions arise. Why did this the model classify this domain so well? Is there a source of bias? Both in chapter 5 and this chapter we have performed an in-depth analysis of the results as practicable, but as the processes are laborious, some interesting analyses were not performed. Below are examples of additional insights, which were not performed:

We made a t-SNE plot of the topics based on domains. The plot effectively reveals relations between classes, and it would be interesting to perform the same operations on authors as well. Additionally, performing the plot on all 10k dimensions for either authors or domains would be interesting to investigate.

As discussed in subsection 6.1.3, it is unknown, whether the authors were split well between training and test data in the domain dataset and which effects it had on the domain results. The same applies to the authorship dataset. Investigating this could reveal why some domains and authors are harder to classify than others.

We attempted plotting the average vector values for authors and domains, though this did not provide any useful insight. However, as TF-IDF measures the importance of a term, it would be interesting to investigate some of the most important terms in various vectors, such as which word class is most revealing when identifying authors?

Some articles had sponsored content which by law has to be stated as such. It is likely that this has been captured as a feature for domain classification, but how and which features represent it is not clear. It could be interesting to see, which domains tend to publish sponsored content and to what degree the sponsors rely on certain newspapers.

6.5 FUTURE WORKS

As mentioned throughout this discussion, the work in this study can be extended in several ways. We have focused our project around feature representations of text but we also wish to note that the implications of our results can be applied practically and in an ethical discussion as well.
Socio-political analyses and in-depth discussion is beyond the scope of this study but future studies in political and social sciences are encouraged to dive into the linguistic differences between the aforementioned domains to illuminate how political differences might be reflected in rhetorics. In 2020, Chen et al. [16] investigated methods of measuring political bias in a news corpus in different granularity, even down to specific words. Such a feature would be interesting to include, especially for comparing domains, as some tend to assume a political standpoint. Furthermore, in subsection 3.5.2 we found that headline lengths increase along with publication year, contrary to our expectations. As noted by previous studies [61, 79], the shortness of headline improves readability, and we assumed the headlines would get shorter with clickbait becoming a more dominant concept of the web-based news scene. With more and more articles available online and with a multitude of them relying on advertising revenue, one would assume the sensational clickbait headlines to be frequent. Perhaps clickbait, in fact, appears as unnecessarily long headers, e.g. *Top 10 cutest capybaras chilling in hot springs: number 8 will make your heart melt!*, but from inspections of the headlines in our dataset, the longest headline tend be due two different trends: briefing news, where multiple short headlines are collected in one, or headlines containing the name of the newspaper itself. The latter is likely due to the headline appearing in social media when it is linked, making people know its origin. Further investigation of the linguistic differences can help inform readers about bias and contribute to their ability to make informed decisions about the articles they consume and domains they support. An exploration of clickbait would also aid the transparency of the methods online media make use of to attract readers.

In relation to domains, we worked with a monolingual news corpus. The features extracted should in theory be language agnostic, except for a few components, i.e. the POS tagger and list of function words. As the features extracted are commonly used NLP features, it would be interesting to see how well our encodings scale to other languages and corpora. There are a multitude of news article corpora available online, some with the goal of identifying fake news, which could be an interesting task to challenge our encoding with. With minimal modifications, how wide could we expand the range of tasks for the encodings?

Additionally, it could be worthwhile to work with data of only freelance journalists, as they are not tied to a single newspaper, which multiple of the authors in our dataset are. Isolating freelance journalists would make a dataset, where the authors and newspapers are less connected, making the distinction between authorship attribution and newspaper attribution even more clear. On a similar note, investigating the use of ghost writers or automated articles would be valuable to gain insight into the current state of public news. As explained in subsubsection 3.4.4.1 we performed web scraping in order to update incorrectly credited authors of some articles. It is safe to say that there are still some mis-credited authors in the dataset, though not as big an amount as prior to updating the authors. On note of ghost writing, it is unknown to what degree the actual credited authors are in fact the real authors. How big of a trend is ghost writers in the journalistic world? Ghost writing is the concept of attributing one writer's work to another author, i.e. a politician having a writer for their speeches. The concept is arguably a larger trend in other fields of literature than articles, such as biographies. However, in recent years the debate on fake news and automated journalism has flourished and to what degree this is a problem for our dataset is not known. A method for distinguishing fake news is with an autoencoder tuned for detecting anomalies. Training it on real news by real people, a fake news article written by a bot would protrude from the real articles. Additionally, (legitimate) automated articles would likely be more common in articles following a template such as reporting results from sport events or changes in the stock market. Investigating such articles would be a project on its own.

The corpus we used is not publicly available but we hope that future works will make use of similar corpora to dive further into the findings we have presented.

Summary 6 - Discussion

In this chapter we have discussed and interpreted the results obtained by our models. We have investigated the strengths of our model and its drawbacks along with a thorough analysis of the reasons behind the obtained results. We discussed which mistakes we made during the process and how they should be fixed and how our project could be expanded in future works.

The key points from the discussion are:

- Newspaper attribution initially seemed overly optimistic, however investigation of the results suggest it is not completely due to data leaks. (subsection 6.1.1, subsubsection 6.1.1.1)
- Adding more syntactic features to the manual encodings is worth exploring. (section 6.3)
- The headlines produced were not of use. We will leave the headline generation model to continue training until validation loss no longer decreases. (subsection 6.1.4)
- With more time, additional data insights such as the split of authors in the newspaper subset, and the most important terms in the vector features would be valuable to explore to better interpret the results (subsection 6.1.3).
- Future works would benefit from expanding this study to corpora of other genres and other languages to asses how well the approach scales. A corpus of articles from freelance journalists would be especially interesting to investigate (section 6.5).

7

CONCLUSION

In this project we worked with a large novel dataset consisting of Danish online news articles, with the goal of creating a general-purpose textual encoding applicable to a range of tasks. We derived the research question: ***How can we create a general-purpose feature encoding for authorship attribution, newspaper attribution, and headline generation?*** along with a series of sub-questions to answer this question. We compared manually extracted features, automatically extracted features, and a combination of these two to see which would perform the best on the mentioned tasks. The novelty aspects of the project were, besides working with a novel Danish dataset, the task of attributing newspapers and generating headlines from commonly used text features.

The manually extracted features were composed of scalar and vector features, such as lexical diversity, word and sentence statistics, n-grams, and skipgrams. The automatically extracted features were obtained using a BERT transformer model for textual representation.
Two of the tasks are classification and the third is text generation. The classification metrics consistently outperformed the baseline on all metrics. Both classification tasks, i.e. authorship attribution and newspaper attribution, performed the best when combining manually and automatically extracted features.
Results obtained for newspaper attribution prompted further investigation of bias in the data after which we concluded that data leakage was not the main contributor of the good performance but the encoding does, in fact, seem to represent domains well.

Insights into feature importance revealed that the best features for both authorship attribution and newspaper attribution are character n-grams which is in line with findings from previous studies. In analyzing topics and confusion matrices for different domains, we observed that the niche newspapers were clustered in a cohesive manner, grouping similar newspapers together.
Considering the high accuracy in the prediction of newspapers, the results suggest that newspapers can be separated in feature space. However, contradictory to our initial expectations, domains

of similar coverage were easily separated. This may be due to the use of content features, and although some domains cover the same topics, e.g. music, movies, and TV-shows, they may still use language differently.

The training time for the headline generation model was excessively long and the training process had to be terminated prematurely, resulting in all generated headlines being the same. As the validation loss for the model steadily decreased with each epoch, we have decided to continue the training the model in hopes that sensible headlines will be produced eventually.

From the findings, we suggest that a manual feature encoding utilizing both scalar and vector features can scale across text classification tasks, and character n-grams are especially worth including in such an encoding. Furthermore, we encourage an encoding combined of manually and automatically extracted features.

BIBLIOGRAPHY

[1] Ahmed Abbasi and Hsinchun Chen. “Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace”. In: *ACM Transactions on Information Systems (TOIS)* 26.2 (2008), pp. 1–29.

[2] Mohammed Albakry. “Usage prescriptive rules in newspaper language”. In: *Southern Journal of Linguistics* 31.2 (2007), pp. 28–56.

[3] Fahad Alotaiby, Salah Foda, and Ibrahim Alkharashi. “New approaches to automatic headline generation for Arabic documents”. In: *Journal of Engineering and Computer Innovations* 3.1 (2012), pp. 11–25.

[4] Shlomo Argamon and Shlomo Levitan. “Measuring the usefulness of function words for authorship attribution”. In: *Proceedings of the Joint Conference of the Association for Computers and the Humanities and the Association for Literary and Linguistic Computing*. 2005, pp. 1–3.

[5] Shlomo Argamon-Engelson, Moshe Koppel, and Galit Avneri. “Style-based text categorization: What newspaper am I reading”. In: *Proc. of the AAAI Workshop on Text Categorization*. 1998, pp. 1–4.

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

[7] Alexandra Balahur et al. “Sentiment Analysis in the News”. In: *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC’10)*. Valletta, Malta: European Language Resources Association (ELRA), May 2010.

[8] Subhabrata Banerjee. “Newspaper Identification in Hindi”. In: *Intelligent Computing and Applications*. Springer, 2021, pp. 741–747.

[9] Dor Bank, Noam Koenigstein, and Raja Giryes. “Autoencoders”. In: *arXiv preprint arXiv:2003.05991* (2020).

[10] Angelo Basile. “An Open-Vocabulary Approach to Authorship Attribution”. In: *CLEF 2019 Labs and Workshops, Notebook Papers*. Ed. by Linda Cappellato et al. CEUR-WS.org, Sept. 2019.

[11] Mikhail Bautin, Lohit Vijayarenu, and Steven Skiena. “International sentiment analysis for news and blogs”. In: *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 2. 1. 2008, pp. 19–26.

[12] Ilker Bozkurt, O. Baghoglu, and Erkan Uyar. “Authorship attribution”. In: vol. 1. Dec. 2007, pp. 1–5. ISBN: 978-1-4244-1363-8.

[13] Marcelo Luiz Brocardo et al. “Authorship verification for short messages using stylometry”. In: *2013 International Conference on Computer, Information and Telecommunication Systems (CITS)*. IEEE. 2013, pp. 1–6.

[14] John Frederick Burrows et al. *Computation into criticism: A study of Jane Austen’s novels and an experiment in method*. Oxford University Press, USA, 1987.

[15] Wei-Fan Chen et al. “Analyzing Political Bias and Unfairness in News Articles at Different Levels of Granularity”. In: *Proceedings of the Fourth Workshop on Natural Language Processing and Computational Social Science*. Online: Association for Computational Linguistics, Nov. 2020, pp. 149–154.

[16] Wei-Fan Chen et al. “Analyzing Political Bias and Unfairness in News Articles at Different Levels of Granularity”. In: *Proceedings of the Fourth Workshop on Natural Language Processing and Computational Social Science*. Online: Association for Computational Linguistics, Nov. 2020, pp. 149–154.

[17] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[18] Sumit Chopra, Michael Auli, and Alexander M Rush. “Abstractive sentence summarization with attentive recurrent neural networks”. In: *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*. 2016, pp. 93–98.

[19] Carlos A Colmenares et al. “Heads: Headline generation as sequence prediction using an abstract feature-rich space”. In: *Proceedings of the 2015 conference of the North American chapter of the Association for Computational Linguistics: human language technologies*. 2015, pp. 133–142.

[20] Olivier De Vel et al. “Mining e-mail content for author identification forensics”. In: *ACM Sigmod Record* 30.4 (2001), pp. 55–64.

[21] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[22] Joachim Diederich et al. “Authorship attribution with support vector machines”. In: *Applied intelligence* 19.1 (2003), pp. 109–123.

[23] Bonnie Dorr, David Zajic, and Richard Schwartz. *Hedge trimmer: A parse-and-trim approach to headline generation*. Tech. rep. MARYLAND UNIV COLLEGE PARK INST FOR ADVANCED COMPUTER STUDIES, 2003.

[24] Yong Fang, Yue Yang, and Cheng Huang. "EmailDetective: An Email Authorship Identification And Verification Model". In: *The Computer Journal* 63.11 (2020), pp. 1775–1787.

[25] Rama Rohit Reddy Gangula, Suma Reddy Duggenpudi, and Radhika Mamidi. "Detecting political bias in news articles using headline attention". In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. 2019, pp. 77–84.

[26] Daniil Gavrilov, Pavel Kalaidin, and Valentin Malykh. "Self-attentive model for headline generation". In: *European Conference on Information Retrieval*. Springer. 2019, pp. 87–93.

[27] Namrata Godbole, Manja Srinivasaiah, and Steven Skiena. "Large-Scale Sentiment Analysis for News and Blogs." In: *Icwsm* 7.21 (2007), pp. 219–222.

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

[29] Dmitry Grigorev and Vladimir Ivanov. "Inno at SemEval-2020 Task 11: Leveraging Pure Transfomer for Multi-Class Propaganda Detection". In: *Proceedings of the Fourteenth Workshop on Semantic Evaluation*. 2020, pp. 1481–1487.

[30] Vishal Gupta and Gurpreet Singh Lehal. "A survey of text summarization extractive techniques". In: *Journal of emerging technologies in web intelligence* 2.3 (2010), pp. 258–268.

[31] Oren Halvani and Lukas Graner. "Cross-Domain Authorship Attribution Based on Compression". In: *CEUR Workshop Proceedings* 2125.1 (2018), pp. 1–19.

[32] James Hammerton. "Named entity recognition with long short-term memory". In: *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*. 2003, pp. 172–175.

[33] Niels Dalum Hansen et al. "Temporal context for authorship attribution". In: *Information Retrieval Facility Conference*. Springer. 2014, pp. 22–40.

[34] Sepp Hichreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[35] John Houvardas and Efstathios Stamatatos. "N-gram feature selection for authorship identification". In: *International conference on artificial intelligence: Methodology, systems, and applications*. Springer. 2006, pp. 77–86.

[36] Laurent Itti and Christof Koch. “Computational modelling of visual attention”. In: *Nature reviews neuroscience* 2.3 (2001), pp. 194–203.

[37] Carina Jacobi, Wouter Van Atteveldt, and Kasper Welbers. “Quantitative analysis of large amounts of journalistic texts using topic modelling”. In: *Digital journalism* 4.1 (2016), pp. 89–106.

[38] Di Jin et al. “Hooks in the Headline: Learning to Generate Headlines with Controlled Styles”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 5082–5093.

[39] Hongyan Jing and Kathleen McKeown. “Cut and paste based text summarization”. In: *1st Meeting of the North American Chapter of the Association for Computational Linguistics*. 2000.

[40] Anders Johannsen, Héctor Martınez Alonso, and Barbara Plank. “Universal dependencies for danish”. In: *International Workshop on Treebanks and Linguistic Theories (TLT14)*. 2015, p. 157.

[41] Shinya Kawata and Yoshi Fujiwara. “Constructing of network from topics and their temporal change in the Nikkei newspaper articles”. In: *Evolutionary and Institutional Economics Review* 13.2 (2016), pp. 423–436.

[42] Mike Kestemont, Folgert Karsdorp, and Marten Düring. “Mining the twentieth century’s history from the time magazine corpus”. In: *Proceedings of the 8th workshop on language technology for cultural heritage, social sciences, and humanities (LaTeCH)*. 2014, pp. 62–70.

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

[44] Yakunin Kirill et al. “Propaganda Identification Using Topic Modelling”. In: *Procedia Computer Science* 178 (2020), pp. 205–212.

[45] Nikita Kitaev and Dan Klein. “Constituency Parsing with a Self-Attentive Encoder”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 2676–2686.

[46] Bradley Kjell, W Addison Woods, and Ophir Frieder. “Information retrieval using letter tuples with neural network and nearest neighbor classifiers”. In: *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*. Vol. 2. IEEE. 1995, pp. 1222–1226.

[47] Moshe Koppel, Navot Akiva, and Ido Dagan. “Feature instability as a criterion for selecting potential style markers”. In: *Journal of the American Society for Information Science and Technology* 57.11 (2006), pp. 1519–1525.

[48] Moshe Koppel and Jonathan Schler. “Exploiting stylistic idiosyncrasies for authorship attribution”. In: *Proceedings of IJCAI’03 Workshop on Computational Approaches to Style Analysis and Synthesis*. Vol. 69. 2003, pp. 72–80.

[49] Moshe Koppel, Jonathan Schler, and Shlomo Argamon. “Authorship Attribution: What’s Easy and What’s Hard?” In: *Journal of Law and Policy* 21.2 (2013), pp. 317–331.

[50] Moshe Koppel, Jonathan Schler, and Shlomo Argamon. “Computational Methods in Authorship Attribution”. In: *Journal of the American Society for Information Science and Technology* 60.1 (2009), pp. 9–26.

[51] Lun-Wei Ku, Yu-Ting Liang, and Hsin-Hsi Chen. “Opinion extraction, summarization and tracking in news and blog corpora”. In: *Proceedings of AAAI*. 2006, pp. 100–107.

[52] Kirti Kumari et al. “Bilingual Cyber-aggression detection on social media using LSTM autoencoder”. In: *Soft Computing* 25 (2021), pp. 8999–9012.

[53] Konstantina Lazaridou and Ralf Krestel. “Identifying political bias in news articles”. In: *Bulletin of the IEEE TCDL* 12 (2016).

[54] Yong-Bae Lee and Sung Hyon Myaeng. “Text genre classification with genre-revealing and subject-revealing features”. In: *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. 2002, pp. 145–150.

[55] Chen Li et al. “Document summarization via guided sentence compression”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 2013, pp. 490–500.

[56] Chin-Yew Lin. “Rouge: A package for automatic evaluation of summaries”. In: *Text summarization branches out*. 2004, pp. 74–81.

[57] Grace W. Lindsay. “Attention in Psychology, Neuroscience, and Machine Learning”. In: *Frontiers in Computational Neuroscience* 14 (2020). ISSN: 1662-5188.

[58] Peter J Liu et al. “Generating Wikipedia by Summarizing Long Sequences”. In: *International Conference on Learning Representations*. 2018.

[59] Konstantin Lopyrev. “Generating news headlines with recurrent neural networks”. In: *arXiv preprint arXiv:1512.01712* (2015).

[60] Henk Pander Maat. “Editing and genre conflict: How newspaper journalists clarify and neutralize press release copy”. In: *Pragmatics* 18.1 (2008), pp. 87–113.

[61] Ingrid Mårdh. *Headlinese: On the grammar of English front page headlines*. Vol. 58. Liberläromedel/Gleerup, 1980.

[62] Philip M McCarthy and Scott Jarvis. "MTLD, vocd-D, and HD-D: A validation study of sophisticated approaches to lexical diversity assessment". In: *Behavior research methods* 42.2 (2010), pp. 381–392.

[63] Florian Meier, Birger Larsen, and Frederik Stjernfelt. "Exploring the Potential of Bootstrap Consensus Networks for Large-scale Authorship Attribution in Luxdorph's Freedom of the Press Writings". In: *CEUR Workshop Proceedings*. Vol. 2612. CEUR Workshop Proceedings. 2020, pp. 110–124.

[64] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems* 26 (2013).

[65] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[66] Shervin Minaee et al. "Image Segmentation Using Deep Learning: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[67] Frederick Mosteller and David L Wallace. "Inference in an authorship problem: A comparative study of discrimination methods applied to the authorship of the disputed Federalist Papers". In: *Journal of the American Statistical Association* 58.302 (1963), pp. 275–309.

[68] Lukas Muttenthaler and Gordon Lucas. "Authorship Attribution in Fan-Fictional Texts given variable length Character and Word N-Grams Notebook for PAN at CLEF 2019". In: Jan. 2019.

[69] Ani Nenkova and Kathleen McKeown. "A survey of text summarization techniques". In: *Mining text data*. Springer, 2012, pp. 43–76.

[70] Andrew Ng et al. "Sparse autoencoder". In: *CS294A Lecture notes* 72.2011 (2011), pp. 1–19.

[71] Joakim Nivre et al. "Universal dependencies v1: A multilingual treebank collection". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. 2016, pp. 1659–1666.

[72] Kishore Papineni et al. "Bleu: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, pp. 311–318.

[73] Fuchim Peng et al. "Automated authorship attribution with character level language models". In: *10th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003)*. Vol. 10. 1067807.1067843. 2003.

[74] Yao Jean Marc Pokou, Philippe Fournier-Viger, and Chadia Moghrabi. "Using Frequent Fixed or Variable-Length POS Ngrams or Skip-Grams for Blog Authorship Attribution". In: *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer. 2016, pp. 63–74.

[75] Kevin M Quinn et al. "How to analyze political attention with minimal assumptions and costs". In: *American Journal of Political Science* 54.1 (2010), pp. 209–228.

[76] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).

[77] Mostafa Rahgouy et al. "Cross-domain Authorship Attribution: Author Identification using a Multi-Aspect Ensemble Approach". In: *CLEF 2019 Labs and Workshops, Notebook Papers*. Ed. by Linda Cappellato et al. CEUR-WS.org, Sept. 2019.

[78] Suman Ravuri and Andreas Stolcke. "Recurrent neural network and LSTM models for lexical utterance classification". In: *Sixteenth Annual Conference of the International Speech Communication Association*. 2015.

[79] Danuta Reah. *The language of newspapers*. Psychology Press, 2002.

[80] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986, pp. 318–362. ISBN: 026268053X.

[81] Alexander M Rush, Sumit Chopra, and Jason Weston. "A neural attention model for abstractive sentence summarization". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (2015), pp. 379–389.

[82] Yunita Sari, Mark Stevenson, and Andreas Vlachos. "Topic or Style? Exploring the Most Useful Features for Authorship Attribution". In: *Proceedings of the 27th International Conference on Computational Linguistics* 27 (2018), pp. 343–353.

[83] Sunil Saumya and Jyoti Prakash Singh. "Spam review detection using LSTM autoencoder: an unsupervised approach". In: *Electronic Commerce Research* 20 (2020).

[84] Jacques Savoy. "Authorship attribution based on a probabilistic topic model". In: *Information Processing & Management* 49.1 (2013), pp. 341–354.

[85] Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. "Authorship attribution with topic models". In: *Computational Linguistics* 40.2 (2014), pp. 269–310.

[86] Shi-Qi Shen et al. "Recent advances on neural headline generation". In: *Journal of computer science and technology* 32.4 (2017), pp. 768–784.

[87] Grigori Sidorov et al. “Syntactic n-grams as machine learning features for natural language processing”. In: *Expert Systems with Applications* 41.3 (2014), pp. 853–860.

[88] Sandro Skansi. “Autoencoders”. In: *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Cham: Springer International Publishing, 2018, pp. 153–163.

[89] Jette Drachmann Søllinge. “omnibusavis i Den Store Danske på lex.dk.” In: Oct. 2020.

[90] Jette Drachmann Søllinge. “tabloidpresse i Den Store Danske på lex.dk.” In: Jan. 2011.

[91] Efstathios Stamatatos. “A survey of modern authorship attribution models”. In: *Journal of the American Society for Information Science and Technology* 60.3 (2008), pp. 538–556.

[92] Efstathios Stamatatos et al. “Ensemble-based author identification using character n-grams”. In: *Proceedings of the 3rd International Workshop on Text-based Information Retrieval*. Vol. 36. 2006, pp. 41–46.

[93] Efstathios Stamatatos, Nikos Fakotakis, and George Kokkinakis. “Text genre detection using common word frequencies”. In: *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*. 2000.

[94] Josef Steinberger et al. “Multilingual entity-centered sentiment analysis evaluated by parallel corpora”. In: *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*. 2011, pp. 770–775.

[95] Rui Sun et al. “Event-driven headline generation”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2015, pp. 462–472.

[96] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to Sequence Learning with Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014.

[97] Quintus Van Galen and Bob Nicholson. “In search of America: Topic modelling nineteenth-century newspaper archives”. In: *Digital Journalism* 6.9 (2018), pp. 1165–1185.

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

[99] Ike Vayansky and Sathish AP Kumar. “A review of topic modeling methods”. In: *Information Systems* 94 (2020), p. 101582.

[100] Haiyan Wu, Zhiqiang Zhang, and Qingfeng Wu. “Exploring syntactic and semantic features for authorship attribution”. In: *Applied Soft Computing* 111.1 (2021), p. 107815.

[101] Peng Xu et al. “Clickbait? sensational headline generation with auto-tuned reinforcement learning”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing* (2019), pp. 3065–3075.

[102] David Zajic, Bonnie Dorr, and Richard Schwartz. “Automatic headline generation for newspaper stories”. In: *Workshop on Automatic Summarization*. Citeseer. 2002, pp. 78–85.

[103] David Zajic, Bonnie Dorr, and Richard Schwartz. “Bbn/umd at duc-2004: Topiary”. In: *Proceedings of the HLT-NAACL 2004 Document Understanding Workshop, Boston*. 2004, pp. 112–119.

[104] Eva Zangerle et al. “Overview of the Style Change Detection Task at PAN 2020.” In: *CLEF (Working Notes)*. 2020.

[105] Ying Zhao and Justin Zobel. “Effective and scalable authorship attribution using function words”. In: *Asia Information Retrieval Symposium*. Springer. 2005, pp. 174–189.

[106] Rong Zheng et al. “A framework for authorship identification of online messages: Writing-style features and classification techniques”. In: *Journal of the American society for information science and technology* 57.3 (2006), pp. 378–393.

[107] Chengqing Zong, Rui Xia, and Jiajun Zhang. *Text Data Mining*. Vol. 711. Springer, 2021.

A

PREPROCESSING

Below are some figures made when investigating the correlation between average article length and article count for each author.

A.1 ARTICLE LENGTHS PER AUTHOR

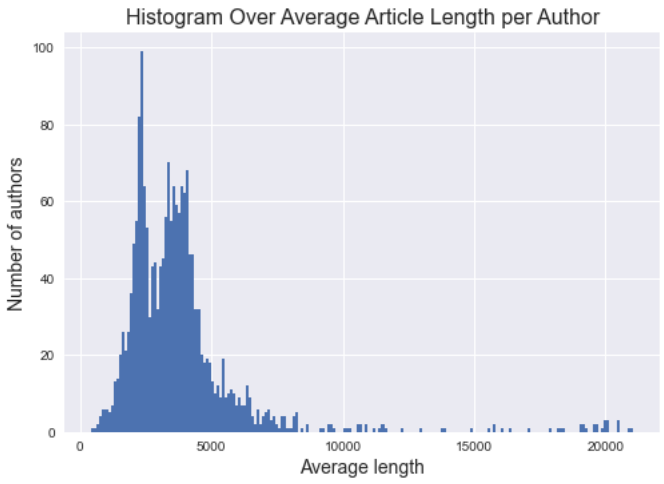


Figure 30: A histogram of the distribution of average article body lengths for each author.

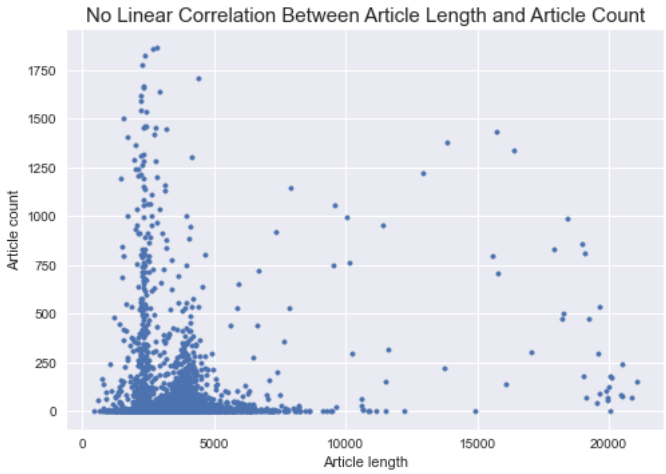


Figure 31: A scatter plot showing possible (lack of) correlation between average article lengths and article counts for authors. Each dot represents an author.

Figure 30 is quite similar to Figure 6 in terms of distribution shape. Figure 31 have no linear correlation, mostly independent horizontal and vertical spread, i.e. no linear correlation between the two variables.

A.2 ARTICLE LENGTHS PER DOMAIN

Simmilarly to author lengths, we investigated lenghts of articles per domain. Not surprisingly, it is skewed. A line plot was chosen here, due to fewer unique classes than authors - the histogram would be unevenly distributed.

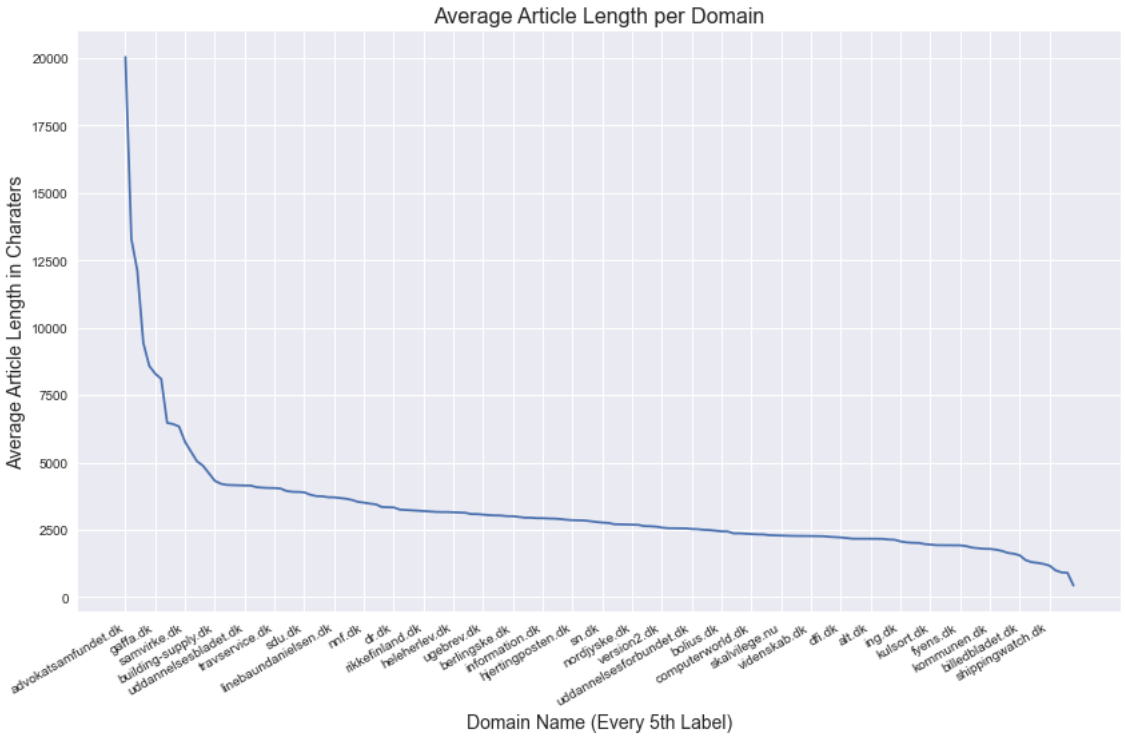


Figure 32: Average article lengths in characters per domain. A sample of domain names is seen on the x axis.

A.3 HEADLINE LENGTHS ON PUBLICATION DATE

The headline lenghts actually increases with newer publications. One could have thought it decreased as clickbait is more common in the age of the internet.

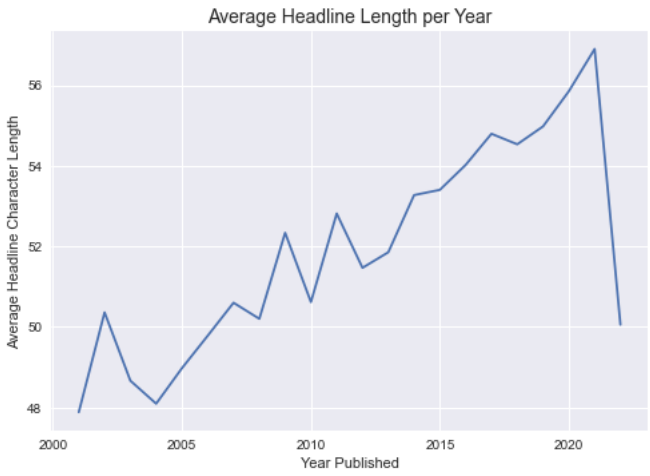


Figure 33: Average article lengths in characters per domain. A sample of domain names is seen on the x axis.

B

IMPLEMENTATION

Below are some figures made when implementing various parts of the project.

B.1 PYTHON LIBRARIES

Below is a table with the libraries used for implementation and their versions.

Package Name	Version
beautifulsoup4	4.10.0
gensim	4.2.0
headliner	1.0.2
lexicalrichness	0.1.4
matplotlib	3.5.1
nltk	3.7
numpy	1.22.3
pandas	1.4.2
scipy	1.8.0
seaborn	0.11.2
selenium	4.1.5
simpletransformers	0.63.6
sklearn	1.1.0
spacy	3.3.0
tensorflow	2.8.0
tensorflow_datasets	4.5.2
tqdm	4.63.1
wordcloud	1.8.1
rouge	1.0.1

Table 18: Python packages used.

B.2 LEXICAL DIVERSITY

As seen in Figure 34 and Figure 35, some measures follow each other, especially RTTR and CTTR. However other measurements do not necessarily agree on the complexity of the texts, which may result in them being a good collection of features.

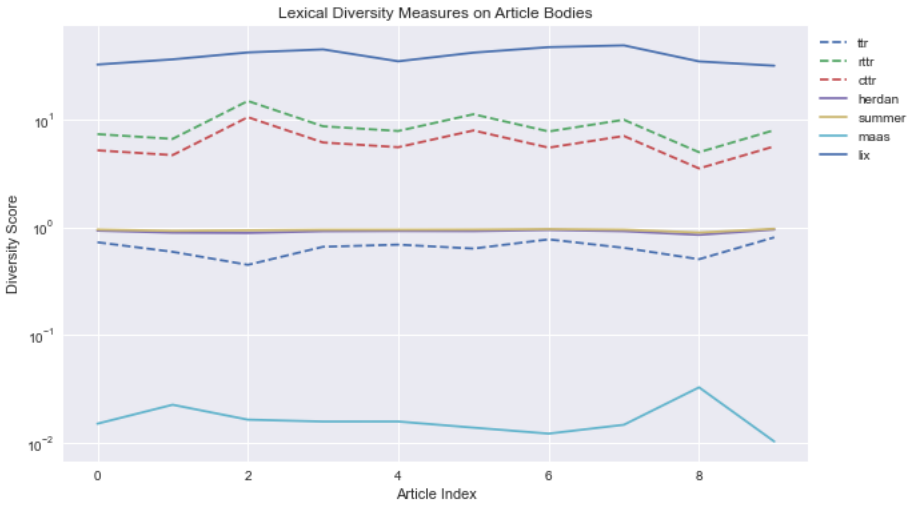


Figure 34: The various lexical diversity measures on 10 different article bodies. Y scale is logarithmic.

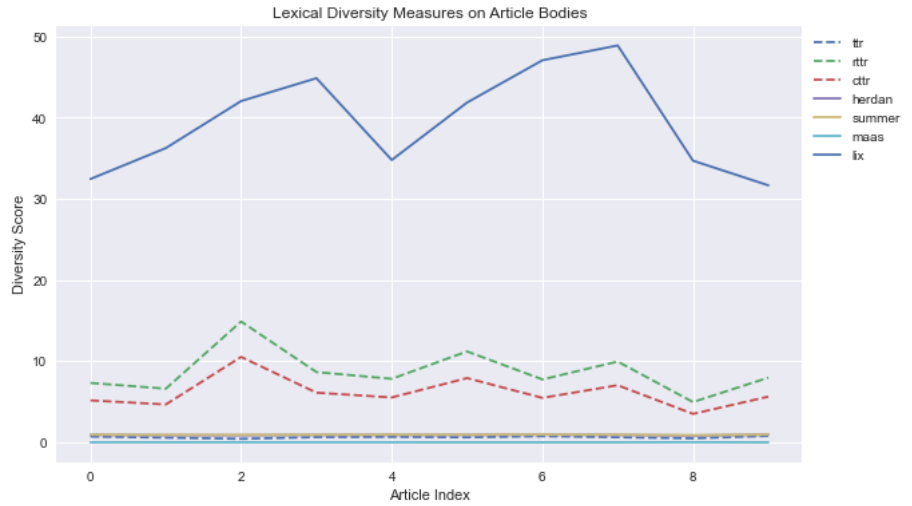


Figure 35: The various lexical diversity measures on 10 different article bodies. Y scale is linear.

B.3 METRICS

All the metrics used for comparison of classification models in the unit test are the following:

- Number of correct classifications
- Accuracy
- Balanced accuracy
- F1
- Top K accuracy
- Cohen's Kappa

- ROC AUC

However, in this report we use a select few to make tables and graphs less cluttered.

B.4 HUMEROUS HEADLINES

Some generated headlines made us laugh. And as they serve no real purpose in the project, but will spread joy, we decided to include them in the appendix for the bored reader. Below are headlines generated from test data using textual input:

Target (test data)	**Prediction**
Skakstjernen går glip af hæder: Det er udueligt!	jeg er ikke så glad for at være en mand i en spand
Ung mand lagt i kunstigt koma efter trafik-ulykke	ung mand i retten for at voldtage sin egen far

Table 19: Innocent and vulgar headline predictions that have little to nothing to do with the target headline.

Another humerous, or rather ominous, prediction was found when using the initial attempt at automatic feature extraction using an LSTM autoencoder. We tested the model on article bodies, training it to reconstruct text. The command line interface had a function for testing the model's ability to reconstruct sentences. When prompted with *Lad os skrive noget længere*, it reconstructed the sentence as *Las savner skævt. savner Copyfighter savner Copyfighter savner Copyfighter savner*, with our data originating from the company, Copyfighter, the model suddenly felt homesick. Besides the ominous feeling of having a self-aware model, we were afraid that Copyfighter would have tagged their data in a lot of articles. However, it was found, that an article from Politiken had written about the company.

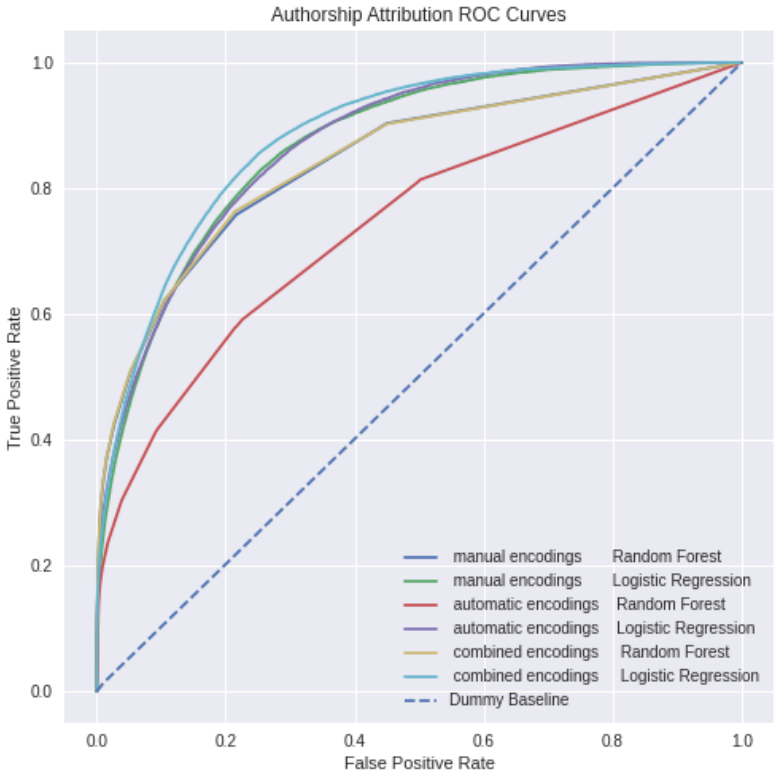
C

EVALUATION

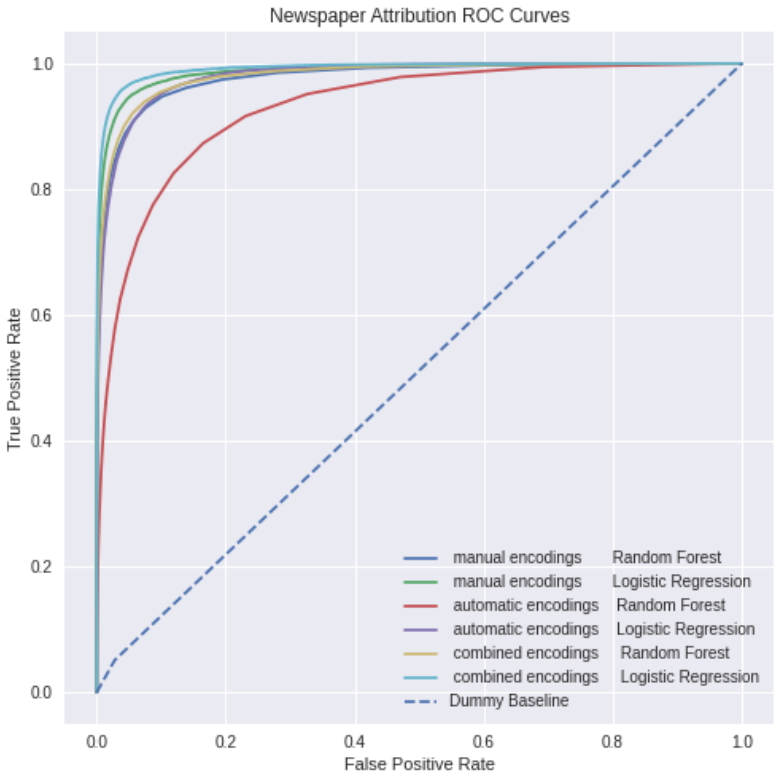
Below are additional information from the evaluation chapter, that did not make the cut in terms of importance. However, they should not feel left out, so they are included here.

C.1 ROC CURVES

Below are larger versions of the ROC curves presented in the evaluation.



(a) ROC curves for authorship attribution using random forest and logistic regression with each type of encoding. Worst performing is random forest on automatic encodings.



(b) ROC curves for newspaper attribution using random forest and logistic regression with each type of encoding.

C.2 FEATURE IMPORTANCES

The plot Figure 16 contains a lot of small text, but it would be a shame to remove it. So here is a larger version of the same plot:

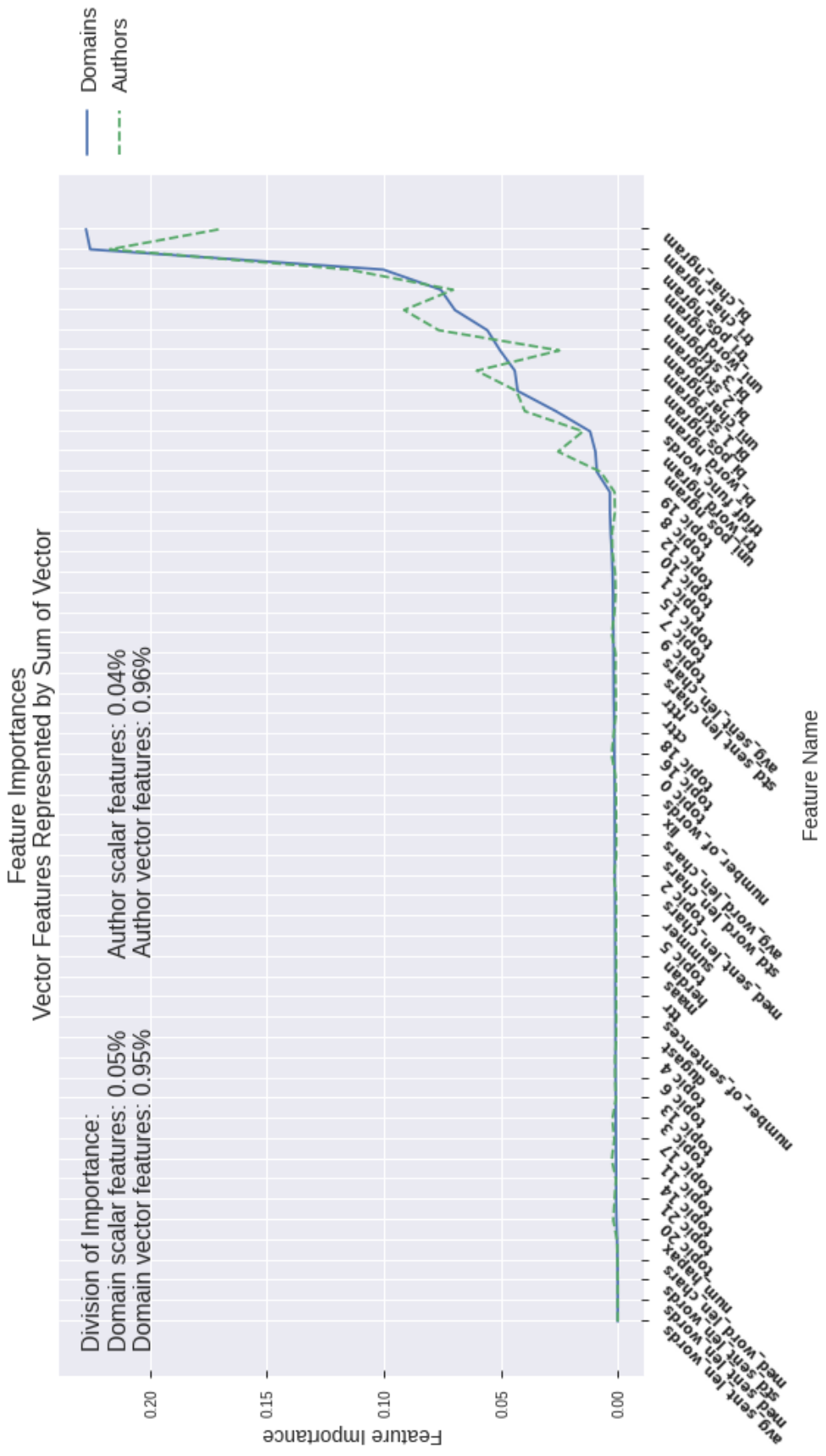


Figure 37: Rotated plot of all feature importances for authorship and newspaper attribution. Vector features are clearly more important.

C.3 TOPICAL COVERAGE OF NEWSPAPER TYPES

Below are the full size versions of the subplots in Figure 21.

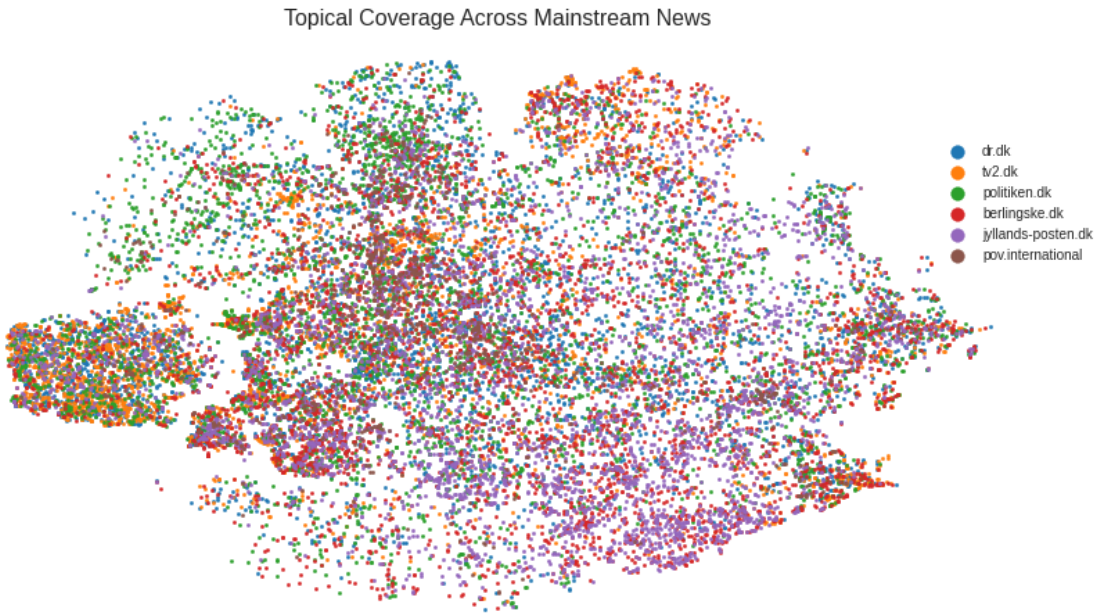


Figure 38: Mainstream news topic coverage. Broad coverage, with a predominant collection of sports (leftmost cluster)

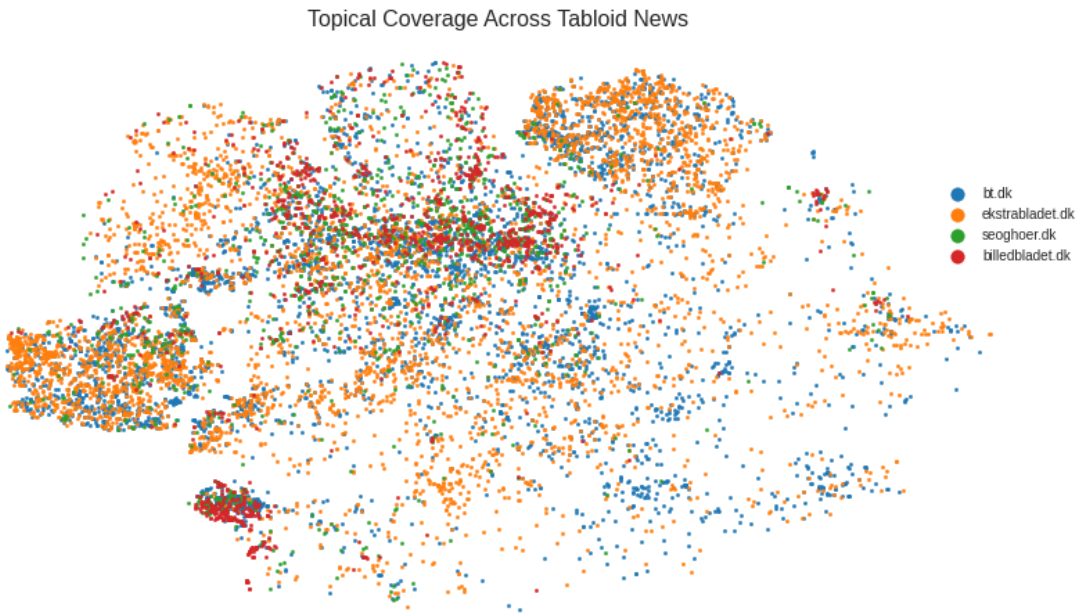


Figure 39: Tabloid newspaper topic coverage. Also a broad coverage, with a new cluster not seen previously, being Family (topic 0), mainly written about by Billed Bladet.

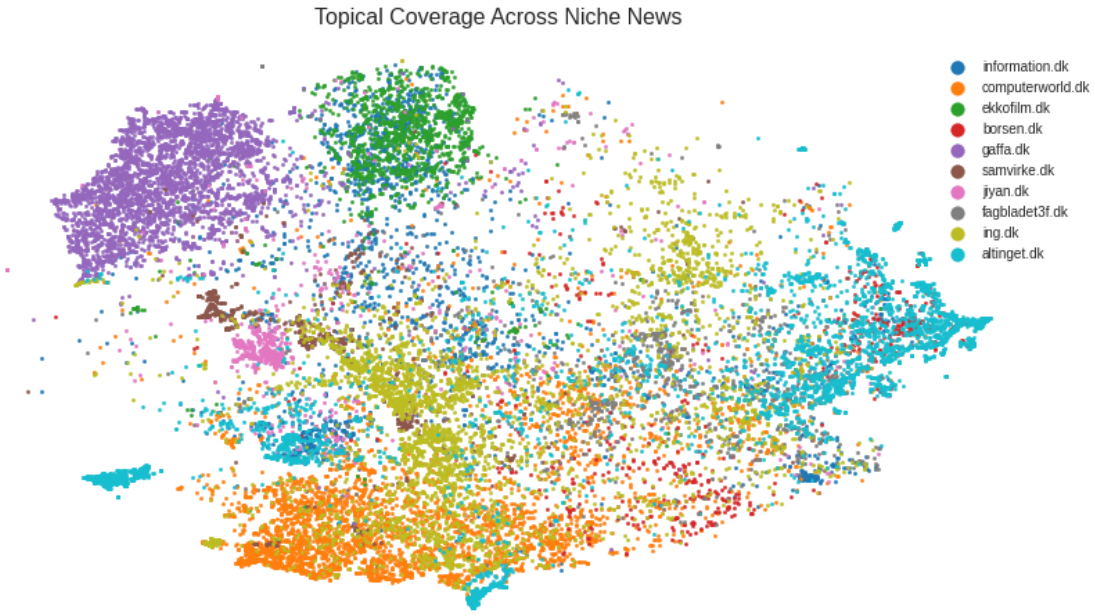


Figure 40: Niche domain topic coverage. Jiyan is a domain for Kurdish news in Danish language, making up most of topic 11, Food. Whether this is indeed food or different cultures is up for debate.

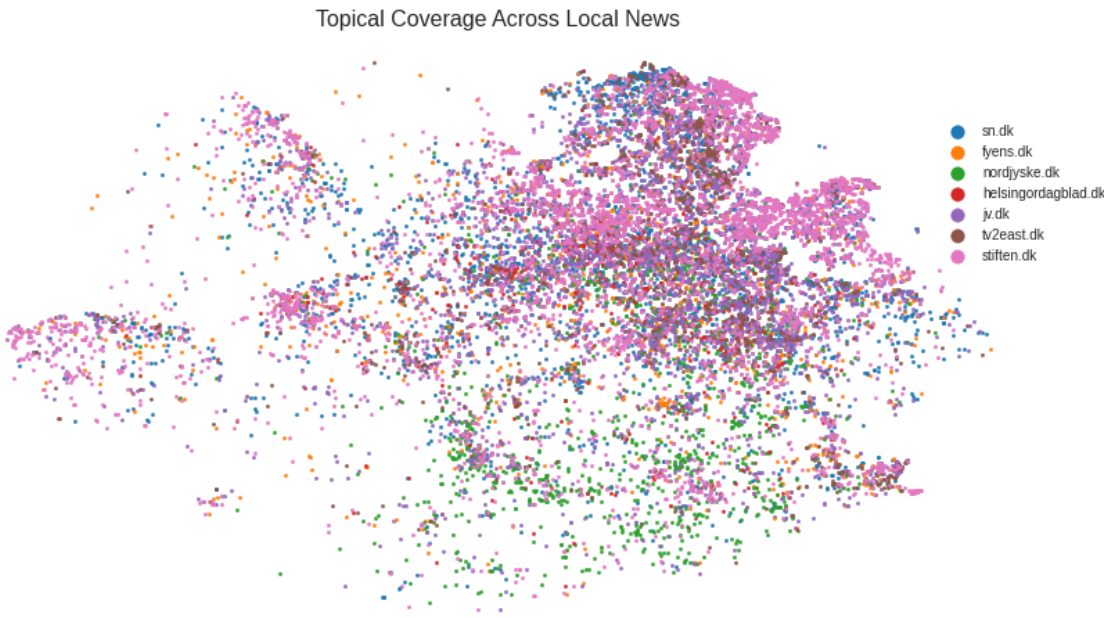


Figure 41: Local news topic coverage. The topics covered by the local news are broad, bu make up a large amount of clusters not easily identified previously. Perhaps because local news can be about a lot of things.

C.4 TRAINING TIMES

The training times for the classifiers across tasks and subsets are available in the table below. All models were trained using cloud system with 16 concurrent processes.

Encoding type	Model	Training Time h:m:s
Author Manual	RF	00:05:31
	LR	00:22:24
Author Auto.	RF	00:00:47
	LR	00:01:57
Author Comb	RF	00:53:38
	LR	00:28:54
Newspaper Manual	RF	00:02:40
	LR	00:20:12
Newspaper Auto.	RF	00:00:15
	LR	00:01:26
Newspaper Comb.	RF	00:35:00
	LR	00:18:02

Table 20: Training times for Logistic Regression (LR) and Random Forest (RF) across tasks (Author = Authorship Attribution, Newspaper = Newspaper Attribution) and feature types (Manual = Manually extracted features, Auto. = Automatically extracted features, Comb. = Combined manual and automatic features)