Linköping University | Department of Computer and Information Science
Master Thesis, 30 ECTS | Statistics and Machine Learning
Spring term, 2020 | LIU-IDA/STAT-A–20/007–SE

Curating news sections in a historical Swedish news corpus

Faton Rekathati

Supervisors: Miriam Hurtado Bodell
and Måns Magnusson

Examiner: Oleg Sysoev

Linköping University
SE-581 83 Linköping
013-28 10 00, www.liu.se

Upphovsrätt

Copyright

Abstract

The National Library of Sweden uses optical character recognition software to digitize their collections of historical newspapers. The purpose of such software is first to automatically segment text and images from scanned newspaper pages, and second to read the contents of the identified text regions. While the raw text is often digitized successfully, important contextual information regarding whether the text constitutes for example a header, a section title or the body text of an article is not captured. These characteristics are easy for a human to distinguish, yet they remain difficult for a machine to recognize.

The main purpose of this thesis is to investigate how well section titles in the newspaper Svenska Dagbladet can be classified by using so called image embeddings as features. A secondary aim is to examine whether section titles become harder to classify in older newspaper data. Lastly, we explore if manual annotation work can be reduced using the predictions of a semi-supervised classifier to help in the labeling process.

Results indicate the use of image embeddings help quite substantially in classifying section titles. Datasets from three different time periods: 1990-1997, 2004-2013, and 2017 and onwards were sampled and annotated. The best performing model (Xgboost) achieved macro F_1 scores of 0.886, 0.936 and 0.980 for the respective time periods. The results also showed classification became more difficult on older newspapers. Furthermore, a semi-supervised classifier managed an average precision of 83% with only single section title examples, showing promise as way to speed up manual annotation of data.

Acknowledgements

I want to thank my supervisors Miriam Hurtado Bodell and Måns Magnusson for being so encouraging in allowing me to try ideas; for their guidance, and for presenting the opportunity to work on such an interesting project.

Contents

1. Introduction

1.1 Background

1.1.1 Digitization and curation of historical documents

Archives, libraries and museums around the world house large collections of audiovisual content, printed books and newspapers. In recent decades, many of these institutions have undertaken efforts to digitize their extensive collections. The National Library of Sweden (Kungliga Biblioteket) started digitizing their materials in the late 1990s. At first digitization was done with a focus on preservation. However, increasingly the library's focus has turned towards improving access to and facilitating research conducted on the digitized content. One especially prioritized area for digitization work has been newspapers, as they constitute "an important research material for many [library] users" (Snickars, 2018).

The digital archives of the National Library of Sweden contain scanned images from four of the country's largest newspaper publications dating back to the 19th century. These images have been digitized using a two step process common to newspapers. In the first step a segmentation algorithm splits the newspaper page in to larger layout component zones. Within each of these larger zones further segmentation is performed to identify sub-zones. Identified sub-zones may either be composed of images or blocks of text. In the second step OCR (optical character recognition) is applied on the segmented text blocks to extract their textual contents.

The current digitization process generates some useful associated metadata connected with each text block. Information regarding the date, page number, font family, font size and the coordinates on a page where the text block was retrieved are logged. However, the mentioned process is unable to produce critical contextual metadata which would help organize digitized contents in the manner humans generally make sense of them. For example: text blocks carry no information as to whether their contents make up a title, body text or ads; neither do they contain any information on which blocks may combine to form cohesive articles.

An often requested – but missing – piece of metadata regards the section (e.g. culture, entertainment, sports) a piece of digitized text belongs to. Field interviews conducted by different researchers help illustrate why such information can be considered of importance. Czarniawska (2014) visited the news agencies Reuters, TT and ANSA to study their organizational cultures. At Reuters detailed coding and classification of news were deemed to be of "central importance" to the organization. The agency produced such vast quantities of news that coding was considered essential to help protect customers from informational overflow. Similarly, at the Italian news agency ANSA, categorizing news items was considered critical to allow editors and clients the ability to browse the company's news archives. Allen, Zhu, and Sieczkiewicz (2010) interviewed historians regarding their needs in interfacing with historical document databases and received similar responses from a user standpoint: the ability to filter content is valued highly when making broad searches returning many results.

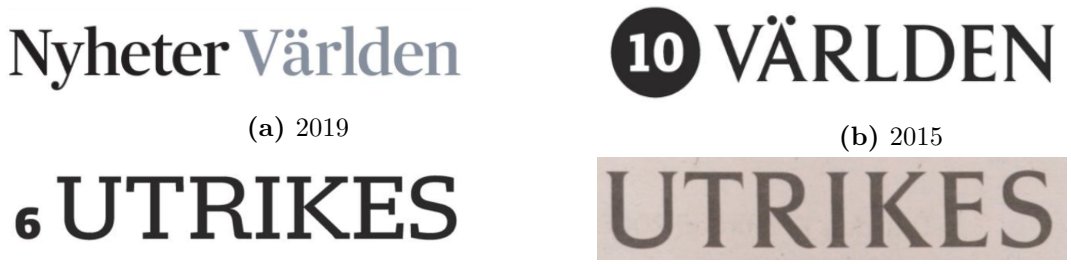
1.1.2 Sections in newspapers

Nerone and Barnhurst (1995) describe a section as a compartment or page(s) of a newspaper where editors have clustered stories sharing topical similarities with each other. However, articles within a section do not necessarily need to be topically similar. An article about an athlete's business ventures may for example be topically similar to articles found in the business section, yet an editor can still choose to categorize it under sports merely because the readers of the sports section are interested in content about the athlete.

In order to more clearly formalize what is meant by a section, we here provide formal definitions of how the terms *section*, *section identifier* and *section title* are to be interpreted in this work.

Definition 1.1 A *section* is a compartment, page, or several pages of a newspaper where newspaper content deemed to be of relevance to the *section identifier* have been grouped together by an editor. Sections are mutually exclusive and exhaustive.
Definition 1.2 A *section identifier* is a title, logo, or an image used to signal newspaper content belonging to a *section*. The section identifier must re-occur in the newspaper at least weekly during a 3 month period to be considered a section identifier.
Subdefinition 1.2.1 A *section title* is a text element with a title of relevance to the contents of a *section*. It is a form of *section identifier*.

The most common type of section identifier found in a newspaper is a section title. However, in certain cases logos and images are used instead of titles. The appearance and naming of titles used for a section can vary across time both within and between different newspapers. Figure 1.1 below provides an example of how section titles signifying the same "international news" section may vary over time in one newspaper.



(a) 2019 **(b)** 2015 **(c)** 2013 **(d)** 2008

Figure 1.1: The changing names and typographic styles of section titles used to signal the "international news" section in Swedish newspaper Svenska Dagbladet during the years 2008 to 2019.

While for a human the task of identifying sections may seem trivial, automated approaches prove challenging for several reasons.

i. The quality of raw images depend upon the state of the physical newspaper copies. Discolored, folded and creased pages are not uncommon.

ii. Names and styles of section titles change over time, as do the pages and the positions of a page in which they appear.

iii. Digitized metadata and text contain errors because segmentation and OCR are not fully accurate. Editorial content and advertisements are also not separated in the process.

iv. Sections are not always clearly marked by section titles. Sometimes a section title may appear just once, but it is implied the section continues until the next title is encountered.

v. Creating gold standard datasets in order to be able to train and effectively evaluate automated approaches is both costly and time consuming.

An approach for identifying sections ideally needs to be flexible and robust enough to handle most of the challenges listed above.

1.2 Related work

1.2.1 Identifying section titles

The vast majority of attempts to connect newspaper content with sections have tended to use text classication approaches. Harbers and Lonij (2017) and Bilgin et al. (2018) classified text data from the National Library of the Netherlands' newspaper archives to eight different news genres using a mix of metadata and TF-IDF (term frequency-inverse document frequency) representations of the textual data. They reported 58% and 70% accuracies respectively. Classifying historical newspaper text data to sections or news genres is however limited by the quality of the OCR procedure. It is therefore more common for text classification studies to make use of data from online publications or newswire services as opposed to historical newspapers.

Lindén, Forsström, and Zhang (2018) retrieved 3600 Swedish language articles from media company Mittmedia's database. The articles were divided in to six categories by the editors of the newspaper before being published. The authors found an LSTM model with continuous bag of words performed the best with a validation accuracy of 71%. In a similar study, García-Mendoza and Gambino Juárez (2018) collected 4031 articles from the online versions of three different Mexican newspapers. They used three different text representations: term frequency counts, binary term occurences and TF-IDF weights as inputs to four different classifiers. They found SVM and logistic regression performed the best, with classification accuracies ranging from 80.3% to 85.5% for the different newspapers. Retrieving texts from online publications or newswire services has the advantage of textual content being organized in cohesive units of articles, which likely impacts model performance positively in comparison to text sourced from OCR.

A second approach towards section classification – and the type of approach used in this thesis – focuses on either using metadata from the digitization process (Hurtado Bodell et al., n.d.), or using the raw images themselves to analyze the design and layout of a newspaper (Wu and Kornprobst, 2019). The image based approaches generally use edge detectors from computer vision literature, either by themselves or in combination with convolutional neural networks (Wu and Kornprobst, 2019). Here, the goal is to classify according to the appearance and layout of the newspaper page as opposed to

directly from the contents of the text.

The National Library of the Netherlands have developed an online tool which allows users to query a database of newspaper advertisements (Lonij and Wevers, 2016). Users of the service supply an image as input. The ten most similar images from the advertisement archives are returned as a result. The tool is based on a method from di Lenardo, Seguin, and Kaplan (2016), where the supplied query image and database images are fed as input to the first few layers of a convolutional neural network pre-trained on a separate dataset. The CNN (convolutional neural network) acts as a feature extractor and creates a condensed vector representation of each image (an "embedding"). The image embeddings can then be used as regular features in classification algorithms such as nearest neighbour or SVMs.

A group of researchers at the Library of Congress (Lee et al., 2020) published a white paper detailing their work in developing a visual content recognition system to recognize for example headlines, illustrations, comics, and advertisements in historical newspapers. They trained an object detection model to perform segmentation and used the cropped images to extract image embeddings to be used for image similarity queries.

1.2.2 Work on the Historical Swedish News Corpus

Early work on curating and adding metadata to the Historical Swedish News Corpus has already been undertaken. Initial efforts have focused on discriminating between editorial and commercial content, classifying whether digitized text contains body text or not, and lastly whether the digitized text block contains a section title or not (Hurtado Bodell et al., n.d.). The authors chose to mainly make use of spatial features instead of textual ones. This was done to avoid potential systematic errors in OCR and text segmentation propagating through to predicted labels, leading to potentially also biasing future research making use of the generated metadata. On the task of binary section title classification an F_1 score of 0.629 was achieved.

1.3 Objective

The main purpose of this thesis is to classify section titles in digitized newspaper pages. More specifically, this thesis investigates whether image embeddings can be used as features in the classification of section titles. While embeddings from pre-trained CNNs have been shown to work well in the context of newspaper advertisements, it is interesting to also study whether they may be effective when applied to images of newspaper text. A further topic of interest is whether the difficulty of classifying section titles changes over time depending on the newspaper's design. Lastly, we explore whether labeling of section titles can be performed more efficiently using only single labeled examples of section titles. The research questions are:

- How well can supervised classification methods classify section titles using image embeddings as input features?

- Is there a difference in how well classifiers perform in periods of changing typographic designs?
- Can the labeling of section titles be done more efficiently using limited training data to guide the process?

1.4 Ethical considerations

The dataset consists of published newspapers which have seen wide circulation. No special considerations need be taken in terms of sensitive or personally identifiable information.

1.5 Delimitations

The National Library of Sweden has digitized materials from the publications Svenska Dagbladet, Dagens Nyheter, Expressen, and Aftonbladet dating all the way back to the 19th century. This thesis only uses data from the newspaper Svenska Dagbladet. The time period studied is also limited to the last 30 years, i.e. 1990 to 2019. Only the first supplement of the newspaper – the part attached to the front page in printing – is considered (see figure 2.1).

2. Data

Svenska Dagbladet has changed designs seven times since 1990. With each change both the layout and the typography of the newspaper were altered. Due to time limitations annotation efforts were focused on three of the seven design periods. The periods were treated as separate datasets. The sampling and annotation is explained in further detail in section 3.1.

Period	Observations	Classes	% non-section
1990-1997	65201	15	99.27%
2004-2013	65830	15	97.88%
2017-	50994	15	95.32%

Table 2.1: The studied time periods span from 1990 to 1997; 2004 to 2013, and 2017 to the month of May in 2019. The number of observations refer to the number of total textboxes in the annotated editions which were used for modeling.

The percentage of section titles in newspapers as a fraction of all text boxes appears to have increased with time (table 2.1). Although this may be related to text blocks becoming bigger over time due to newspaper layout changes. It may partly also be an artifact of how well or how poorly the segmentation performed during the respective periods rather than an indication of an increased prevalence in section titles.

2.1 Annotated section titles

Section titles were manually annotated. All non-section title content in the newspaper was labeled to the category `non-section`. Table 2.2 shows the frequency distribution of labels we are interested in predicting for the annotated datasets of each time period.

Label	n
bridge	10
brännpunkt	36
inrikes	131
kultur	8
marginalen	20
namn_familj	44
non-section	64737
politik	64
samtider	11
sidanfem	10
stockholm	46
stockholmsguiden	14
tv	36
utrikes	83
vädret	44

(a) 1990-1997

Label	n
brännpunkt	53
helg	16
idag	59
kryss_söndag	15
ledare	57
namn_familj	93
nyheter	412
non-section	64437
reportaget	15
sidan2	16
special	10
sport	345
svd_guiden	38
synpunkt	34
utrikes	230

(b) 2004-2013

Label	n
bioprogram	33
debatt	42
familj	38
idag	37
korsord	52
kultur	184
ledare	61
nyheter	41
nyheter_inrikes	263
nyheter_utrikes	207
non-section	49835
sport	34
tvradio	108
understrecket	36
vinmat	23

(c) 2017-

Table 2.2: Freqency distribution of labels in the three different design periods.

2.2 Raw data

Digitized newspaper data at the National Library of Sweden can be accessed through an internal API. The data is stored in a hierarchical structure with increasing granularity as users move through the structure. In the first level we find editions of the printed newspaper. More than one edition of a newspaper can sometimes be printed on the same day. The next step separates the supplements (the separately printed parts) of a newspaper. For each newspaper supplement we find a list of pages. It is on the page level of the hierarchy that the actual scanned images and the coordinates for the larger layout zone boxes are found.

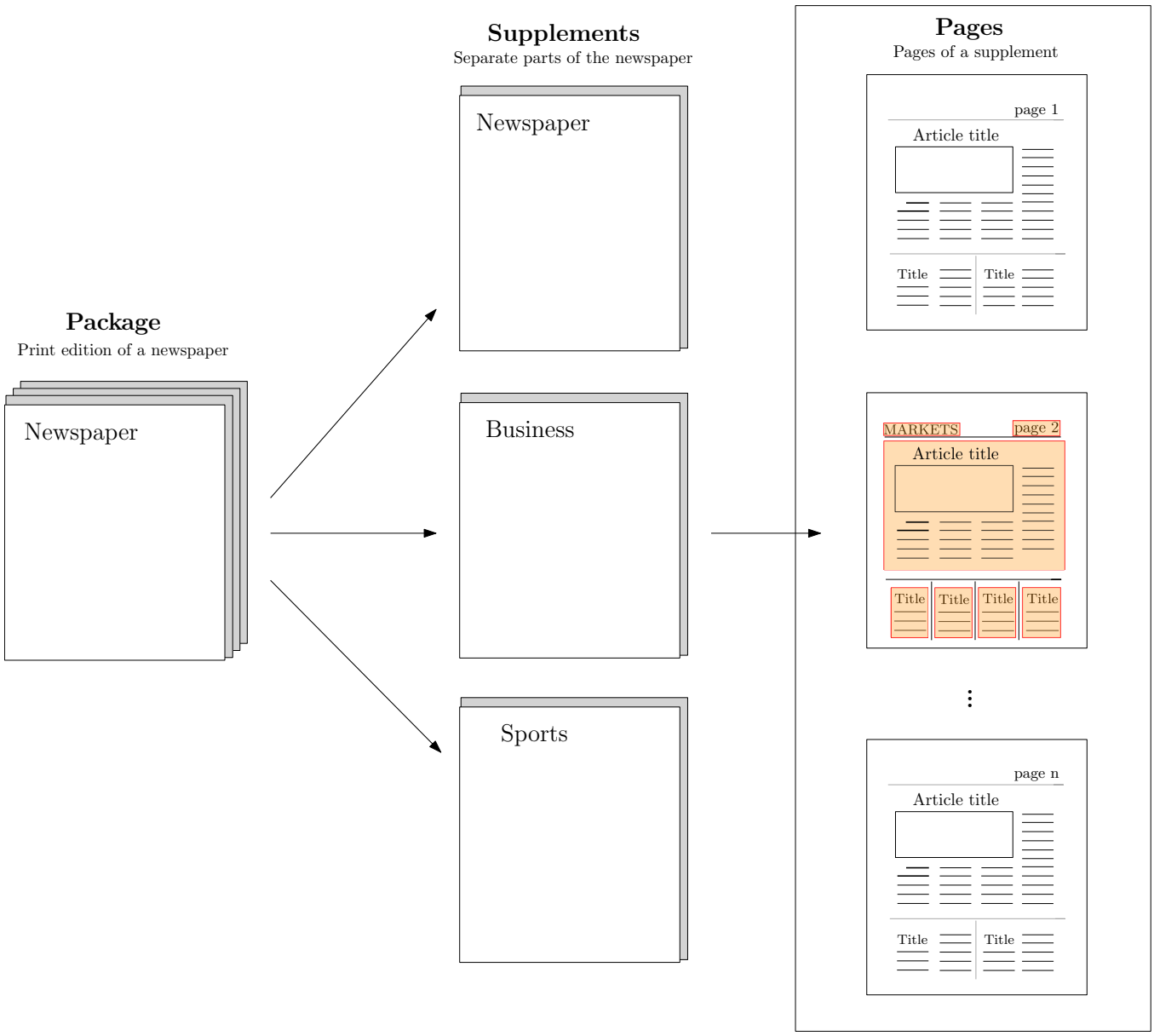


Figure 2.1: The first three levels of the newspaper digitization data structure. In the middle of the page level we have an example of how the segmentation algorithm may divide a page in to greater layout zones (the orange rectangles).

2.3 Data format and structure

The starting point from which the entire digitization procedure proceeds is the scanned image file of a newspaper page. The segmented and OCRed data is stored under the ALTO (Analyzed Layout and Text Object) schema. ALTO is an open XML-standard for storing the layout and text structure of digitized documents. The standard is meant to be able to recreate the layout and appearance of a page even if the original image file is lost. Pixel positions of every layout and text element on a page down to the word and character level are stored. Every textbox is also associated with a font family, font size, font type and style. An ALTO XML-file is connected to each scanned newspaper page.

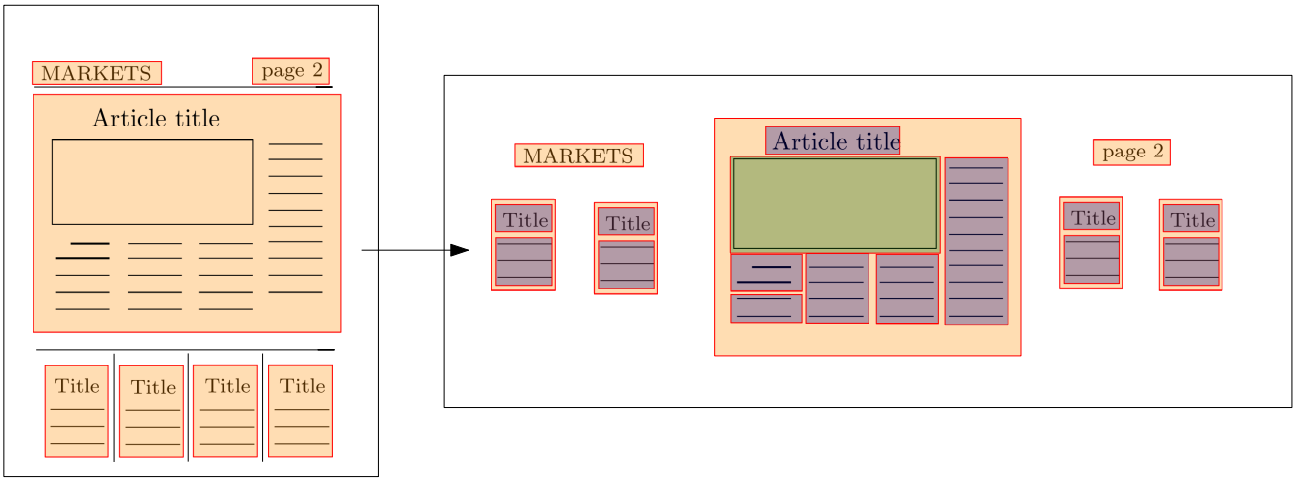


Figure 2.2: ALTO layout zones (to the left) also contain sub-zones of **text boxes** (blue) and **images** (green) within the larger layout zones.

The level of data granularity required for this project's purposes stops at the level of segmented text boxes and images. Every single identified box has a set of (x, y) coordinates denoting the pixel position of the upper left corner of the box within the context of the entire scanned page. The boxes further have `width` and `height` attributes which allow us to reconstruct their position on a page.

2.4 Features

Using the positional attributes of the segmented boxes, an API call which returns cropped images of textboxes can be constructed. From these images embeddings are created for each observation. The embeddings act as our main features. The classification uses the embeddings in combination with seven additional metadata features: `x`, `y`, `width`, `height`, `page`, `weekday` and `font_size`. These are presented in table 2.3 on the next page. In total we have 2055 features, of which 2048 result from the generated embeddings.

Figure 2.3: The position of section titles and other zone "boxes" can be identified using (x, y), `width` and `height` attributes. This allows for easy cropping.

Variable	Type	Description
weekday	Categorical	Day of the week.
page_number	Integer	Page number in supplement for the retrieved textbox.
x	Integer	Upper left corner pixel x-coordinate for textbox in page.
y	Integer	Upper left corner pixel y-coordinate for textbox in page.
width	Integer	Width extending rightwards from (x, y)-coordinate.
height	Integer	Height extending downwards from (x, y)-coordinate.
font_size	Integer	Font size of textual contents within textbox.
image_embedding	Vector (continuous)	2048-dimensional representation of the cropped image of a textbox.

Table 2.3: Variables extracted from OCR metadata and from image embeddings.

2.5 Data quality

The quality of image embeddings, as well as the quality of metadata features and textual contents depend upon the quality of the OCR procedure which generated them. This thesis relies upon the OCR precedure having successfully detected sections titles as textboxes instead of as images. Furthermore, if the procedure fails the section title would consequently be completely missing as a data point – meaning we are neither able to use it in training nor able to predict it.

Figure 2.4: An example of a section title which was not detected at all by the segmentation. Ballpoint pen markings through text tend to confuse the OCR.

A second issue was encountered during the image cropping phase of the project. The positional metadata: `x`, `y`, `width` and `height` of text boxes turned out to be misaligned for significant portions of the data. Further investigation revealed the misalignment

was due to a disagreement between the image dimensions listed in the metadata and the actual dimensions of the raw images. Fortunately this issue could be resolved by applying a scaling factor correction to the metadata features based on the ratio of the raw image dimensions and the incorrectly listed image dimensions in the metadata. Each positional attribute was corrected as follows:

$$\texttt{Positional feature} \cdot \frac{\texttt{Actual image dimension}}{\texttt{Metadata image dimension}}$$

The National Library of Sweden plan on applying the above correction to all positional metadata features in their API.

3. Method

An overview of the project's general work flow is given in figure 3.1. The method chapter is structured to follow the general outline of the chart, providing explanations for the the sampling, data preparation, modeling and evaluation steps in the subsubsections to follow.

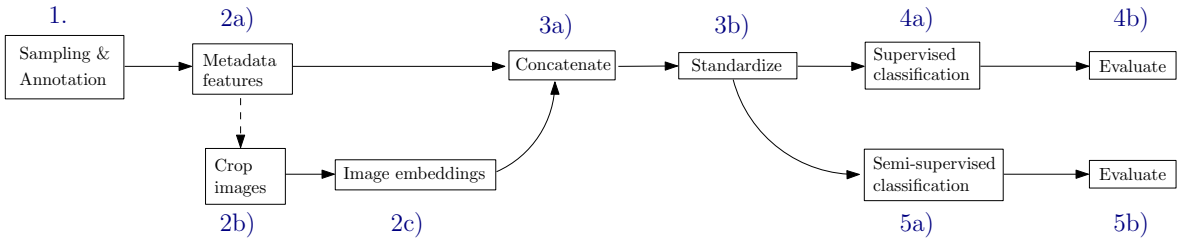


Figure 3.1: Flowchart of the data preparation and modeling processes.

A brief summary of each step is given below.

1. A stratified sampling scheme was implemented to obtain representative samples of newspaper editions from each year. Stratification was done on `year` and `weekday`. The newspaper changed design seven times during the period 1990-2019. Annotation work was focused on three of these time periods. The different design periods were treated as separate datasets in the modeling step.

2. (a) Metadata features from the OCR procedure were collected on the sampled data.

   (b) Using the metadata features, images of text blocks were cropped from each newspaper page.

   (c) The cropped images were passed through a convolutional neural network model with pretrained weights in order to obtain image embeddings.

3. (a) 2048 image embedding features and 7 selected metadata features were combined.

   (b) Each feature was standardized to zero mean and unit variance. This procedure was applied separately to each dataset from the three time periods.

4. (a) Supervised classification was performed with Xgboost and k-nearest neighbours.

   (b) Results were evaluated looking at precision, recall, F_1-measures and variable importance. The models were bootstrapped to provide standard errors.

5. (a) A semi-supervised classifier was developed making use of only a single labeled example.

   (b) The semi-supervised model was evaluated using precision. This model is not compared to the supervised methods, but is rather used to investigate whether labeling can be sped up using only very few training examples.

3.1 Sampling and annotation of data

Stratified sampling was performed on the variables `year` and `weekday`. As a first step three editions were randomly sampled from each weekday within a year (i.e. 21 editions per year). A stratified sampling scheme was chosen because the appearance of section titles were expected to vary based on the weekday a newspaper was published.

Figure 3.2: Three editions were randomly sampled from each weekday within each year. This summed up to 21 editions every year.

Because of time constraints the annotation was further restricted to subsets of the abovementioned dataset. Annotation efforts were focused on three of these design periods (shown in table 3.1). Every design change was announced on the front page of the newspaper by the editors. The start dates for when they were announced and implemented can be found below.

Design period	Start date	End date
1990-1997	**1990-01-19**	**1997-06-17**
1997-2000	1997-06-18	2000-11-15
2000-2001	2000-11-16	2001-12-11
2001-2004	2001-12-12	2004-06-01
2004-2013	**2004-06-02**	**2013-04-10**
2013-2017	2013-04-11	2017-03-28
2017-	**2017-03-29**	**ongoing**

Table 3.1: The start and end dates for each design of Svenska Dagbladet. The periods studied in this thesis are marked in bold (1990-1997, 2004-2013 and 2017-).

To facilitate annotation, the textboxes were filtered based on variables `y` (vertical pixel position), `width`, and `height` before creating spreadsheets used in manual annotation. The following filtering rules were used:

- Only textboxes where `y < 700` were considered when annotating section titles (roughly the top fifth portion of page). The convention in images is to start counting pixels from the top to the bottom of the page (i.e. `y = 0` is at the top).
- Only textboxes where the sum `(width + height)` exceeded 200 pixels were considered when annotating section titles.

All section titles were positioned at the top of a page for the time period `2017-` . In `2004-2013` the vast majority were also found at the top portion of pages. However, during `1990-1997` there was some variation in the placement of section titles. Section

titles under the filtering cut off during this period occured on approximately 1.4% of pages. As a result these were all labelled as `non-section` in our dataset. Each textbox identified as a section title in the filtered spreadsheets was annotated with its name. All other textboxes were assigned to the category `non-section` title. Further details on how the annotated section titles were combined to classes and renamed are available in appendix A.

Figure 3.3 depicts a timeline with the annotated editions for the different studied design periods. The editions were sorted according to their internal API id-numbers and annotated according to this order. Annotation could not be finished for all sampled editions within the given time periods. Thus, some gaps exist in the timeline.

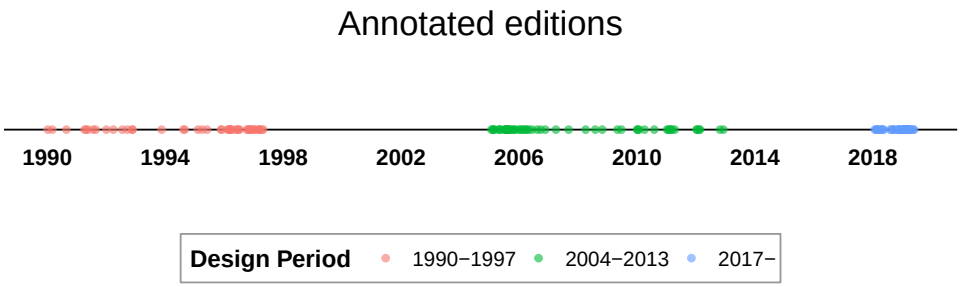


Figure 3.3: 45, 55 and 44 editions were annotated for the time periods `1990-1997`, `2004-2013` and `2017-` respectively.

3.2 Extracting features from images

This section provides a brief introduction to multilayer perceptrons and convolutional neural networks, and motivates the use of CNNs in the context of generating embeddings from images.

3.2.1 Multilayer perceptrons

A fully connected multilayer perceptron (MLP) consists of an input layer, an arbitrary number of hidden layers, and an output layer. The layers are composed of units (also called nodes or neurons) which are connected by weights to all the units in the preceding and proceeding layers. The incoming data to each unit can be expressed as a linear combination of inputs and weights with an added bias term b (Bishop, 2006). An activation function $\sigma(.)$ is then applied to the output of each unit except for the input layer's units. The activation functions are generally chosen to be differentiable and nonlinear. This allows mappings of nonlinear relationships between inputs and outputs and lets the neural network act as a universal function approximator (Hornik, Stinchcombe, and White, 1989).

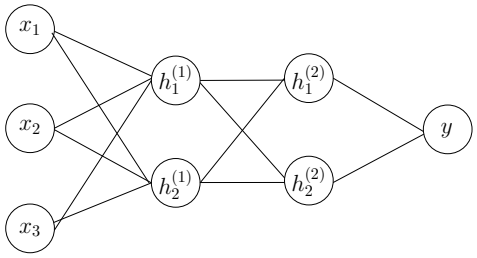


Figure 3.4: An MLP with x denoting the input layer, h the hidden layers, and where y can be either a classification or regression output. The circles are referred to as units, neurons or nodes depending on the source. Weights $w_{i,j}^{(l)}$ connect the units of each layer (the drawn lines). The indices i and j by convention denote which unit a weight is connecting *to* and *from* respectively. Superscript indices denote the layer number.

The output of each layer in figure 3.4 can be represented in a compact manner using matrix multiplications. Below, all the unit outputs of a given layer are collected in matrices, and an activation function $\sigma^{(l)}(.)$ is applied. The superscript of the activation functions denote that a different function may be chosen in each layer. The process of going from inputs to an output is referred to as a feedforward pass of the network.

$$\begin{aligned} \boldsymbol{H}^{(1)} &= \sigma^{(1)}\left(\boldsymbol{X}\boldsymbol{W}^{(1)T} + \mathbf{1}\boldsymbol{b}^{(1)T}\right) \\ \boldsymbol{H}^{(2)} &= \sigma^{(2)}\left(\boldsymbol{H}^{(1)}\boldsymbol{W}^{(2)T} + \mathbf{1}\boldsymbol{b}^{(2)T}\right) \\ \boldsymbol{Y} &= \sigma^{(3)}\left(\boldsymbol{H}^{(2)}\boldsymbol{W}^{(3)T} + \mathbf{1}\boldsymbol{b}^{(3)T}\right) \end{aligned} \tag{3.1}$$

To make things clearer, an example is presented for the calculation of $H^{(1)}$ with matrix dimensions explicitly written out for a feedforward pass with n training examples:

$$\underset{n\times 2}{\boldsymbol{H}}^{(1)} = \sigma^{(1)}\left(\underset{n\times 3}{\boldsymbol{X}} \times \underset{3\times 2}{\boldsymbol{W}}^{(1)T} + \underset{n\times 1}{\mathbf{1}} \times \underset{1\times 2}{\boldsymbol{b}}^{(1)T}\right).$$

3.2.2 Convolutional neural networks

A convolutional neural network is composed of two main parts: convolutional layers followed by an MLP attached at the end. Convolutional layers differ from densely connected ones in that they are sparsely connected by weights (table 3.2). In theory, the feedforward step of data through convolutional layers can be expressed the same way as in equation 3.1 (although in practice it is not implemented in terms of matrix multiplications). The major difference between dense and convolutional layers lie in how the weight matrix $\boldsymbol{W}$ is populated by weights.

$$\begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \vdots & & \ddots & \vdots \\ w_{m,1} & \dots & \dots & w_{m,n} \end{bmatrix} \qquad \begin{bmatrix} w_1 & w_2 & 0 & 0 & \dots & 0 \\ 0 & w_1 & w_2 & 0 & \dots & 0 \\ 0 & 0 & \ddots & \ddots & 0 & \vdots \\ \vdots & \vdots & 0 & w_1 & w_2 & 0 \\ 0 & 0 & \dots & 0 & w_1 & w_2 \end{bmatrix}$$

Table 3.2: Weights of a densely connected layer (left) are unique. Each unique weight is multiplied with an input node only once. In contrast, convolutional layers (right) use weight sharing, where the same set of weights slide over the entire input. Convolutional layer weights are therefore commonly referred to as "filters" or "kernels". Multiple kernels are learned in every convolutional layer – each specializing at detecting some specific aspect of the input.

The convolution operation can be seen as a kernel of weights sliding over the entire input space. This means each weight in the kernel is multiplied at every position of the input (Goodfellow, Bengio, and Courville, 2016). The weight-sharing mechanic of convolutional layers works well on images because images exhibit spatial correlation. A given pixel in an image will be more strongly correlated to nearby pixels than to pixels farther away (Bishop, 2006). Thus, it is not necessary for every unit in one layer to be able to influence every other unit in a subsequent layer.

Weight-sharing provides an important property to CNNs called translational equivariance. Since the same weights are used over the entire input, an object may be placed at different positions within an image and yet the computed activations will be the same (albeit also shifted in their positions). Variation in segmentation quality of section titles resulting in positional shifts within an image should therefore not greatly impact whether or not an activation is present in our extracted features.

The second operation commonly present in every convolutional layer is the *pooling* operation. Pooling is performed after applying convolution and passing the outputs through an activation function. The operation serves to replace a certain output with a computed summary statistic of the nearby outputs (Goodfellow, Bengio, and Courville, 2016). Often the procedure serves as a down-sampling step. However, more importantly its purpose is to make the network invariant to minor translations in the input. In contrast to the equivariance property, translational invariance ensures the activations will approximately be the same without being shifted in their positions. This property is especially important when features are to be used by external classifiers (as they are in this thesis). It ensures the presence of a specific object or shape in an image causes a similar activation in one and the same extracted feature irrespective of whether the object is shifted slightly out of position. This is illustrated in figure 3.5

below.

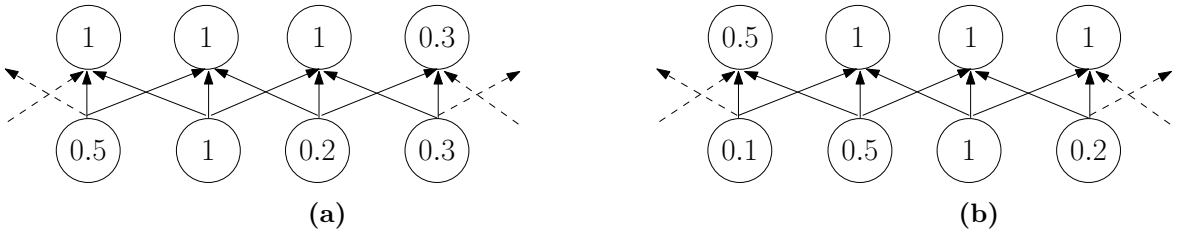


Figure 3.5: Two cases of maximum pooling. At the left in figure 3.5a we have the inputs on the bottom, with the results of the maximum pooling in the top units. The top nodes' values are set to the maximum of the three input values pointing towards them. At the right in figure 3.5b all the inputs have been shifted one step rightwards. Despite all of the inputs being "off position", half of the max pooling outputs still retain the same activations. The above illustration of local translational invariance draws inspiration from an example in Goodfellow, Bengio, and Courville (2016).

3.2.3 Image embeddings

Suppose we have an image $\mathbf{I}$ – a 3D-array with dimensions $456 \times 456 \times 3$ (width, height and RGB color channels). We can extract a set of features $\vec{x}$ (vector) from this image to be used in a classifer by passing the image as an input to a feature generating convolutional neural network $f_\theta(\mathbf{I})$ whose parameters are θ. Such a CNN has typically been trained on a dataset similar to $\mathbf{I}$ with the goal of learning a set of weights θ to minimize the loss of a classifier. The purpose of the convolutional layers can be viewed as one of finding the best mapping of the input to a lower dimensional "embedding space" where the classes are separable. The word "embedding" is commonly used to describe the output of any given layer in such a network (i.e. $f_\theta(\mathbf{I}) = \vec{x}$).

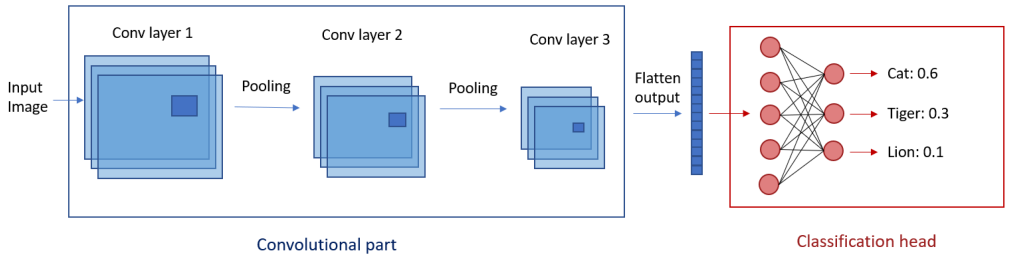


Figure 3.6: CNNs are composed of two main parts: convolutional layers and a densely connected multilayer perceptron used for classification. The classification head can be cut off after training, allowing the use of the convolutional part as a feature generator whose features can be used by any classifier.

3.2.4 Task transfer

Ideally a network is trained on images and classes similar to one's own dataset. However, a CNN tends to require large number of labelled training examples to learn a good mapping from input space to embedding space. When labelled data is not available, one may instead use CNNs pretrained on large natural image datasets such as ImageNet (Russakovsky et al., 2015).

The first layers of a convolutional neural network trained on natural image datasets have been shown to learn generalizable features. In the majority of cases these learned features resemble edge detectors, simple color blobs or linear filters for texture analysis (Yosinski et al., 2014). The first layers can thus be said to recognize simple shapes and textures in an input image. Each subsequent layer in the network proceeds to learn features of increasing levels of abstraction (Olah, Mordvintsev, and Schubert, 2017).

3.2.5 ImageNet

ImageNet is a large scale natural image dataset consisting of millions of manually labelled images. The full dataset contains roughly 14 million images with thousands of output classes (Russakovsky et al., 2015). During the past decade it has been used both as a competition benchmark and as a common dataset to pre-train the weights of CNNs.

It is generally common practice to use ImageNet-initialized weights when training a network on new datasets as opposed to starting out with randomly initialized weights. This practice is called "fine-tuning" a network and goes under the broader name of transfer learning. Training a network in such a manner leads to faster learning and in some cases also higher performance as useful knowledge from the previous dataset may be partially retained even after fine-tuning (Yosinski et al., 2014).

A subset of the full ImageNet dataset is used to pre-train weights. This subset contains 1000 output classes and over a million images. In this thesis, we did not proceed to fine-tune the weights of our chosen network, but rather used the ImageNet-initialized weights directly to generate embeddings. This choice was made because only a limited amount of labeled data was available at the onset of the project. Successful fine-tuning of a network generally requires sufficient training data in order not to overfit.

3.2.6 Efficientnet

We used the network architecture Efficientnet to generate image embeddings (Tan and Le, 2019). The Efficientnet creators designed a baseline network called `b0` through automatic neural architecture search. The baseline design was then scaled up, creating a further seven networks (named `b1` to `b7`). Each subsequent network was also trained with progressively larger input size images (another factor leading to increased performance). In this thesis the Efficientnet `b5` model was used. The model expects input images of size 456×456 pixels. This model was chosen because the scanned newspaper pages were of high resolution.

The network was downloaded with a set of weights obtained through pretraining it on ImageNet. The output of the convolutional layers is used as our feature vector. Counting layers is not straightforward in modern convolutional nets. The output of the final convolutional layer corresponds to the output of the sixth so called MBConv-block. Efficientnet `b5` feature vectors with ImageNet weights are freely and publicly available via Tensorflow Hub [1].

When the network was trained, the color values of input images were transformed to the range of 0 to 1. The same transformation is expected when passing new images to it. Thus each RGB-channel's values were divided by 255 before feeding images to the network. The performed normalization is common to ensure numerical stability in training. The output of the model is an embedding with 2048 features.

3.2.7 Combining image embeddings with other features

Positional data variables `x`, `y`, `width` and `height`; as well as other variables `weekday`, `page_number` and `font_size` were concatenated to the image embedding vector as additional features. All features – both the image embedding and the extra ones – were standardized to have mean 0 and unit variance 1.

$$z_i = \frac{x_i - \bar{x}_i}{s_i} \tag{3.2}$$

A standardized feature transformation z_i is computed by centering the original feature x_i and dividing it with its standard deviation. Recall that our dataset was divided in to three parts depending on the time period and typographic design. The standardization was performed separately across all observations for each specific dataset. The features were transformed so that no feature would have disproportionate influence based on their scale in models depending on distance metrics.

[1] https://tfhub.dev/tensorflow/efficientnet/b5/feature-vector/1

3.3 Supervised classification

3.3.1 K-Nearest Neighbour

K-nearest neighbour (KNN) is a simple yet strong baseline classifier when working with image embeddings.

KNN computes distances between a test observation x_0 and training data. The K points which are closest to the test observation compose its neighbourhood $N_k(x_0)$. Distances for a given training observation x and test observation x_0 are computed

$$d(x, x_0) = \sqrt{\sum_{j=1}^{m}(x_j - x_{0j})^2}, \tag{3.3}$$

where m denotes the number of features in our data set and the index j iterates through the features. A class is assigned to the test observation x_0 by majority voting among the K nearest neighbours according to

$$\hat{Y}(x_0) = \underset{j}{argmax} \left(\frac{1}{K} \sum_{(x_i, y_i) \in N_K(x_0)} I(y_i = j) \right), \tag{3.4}$$

where j is a class label and y_i are the class labels of the observations in the neighbourhood of $N_K(x_0)$. The number of observations where j and y_i match are counted by applying the indicator function $I(.)$ (Hastie, Tibshirani, and Friedman, 2001). We use only the single nearest neighbour in classification (i.e. $K = 1$). This convention of using small K or a centroid point is common in the field of few-shot and metalearning applied on embeddings (Wang et al., 2019).

3.3.2 Gradient boosted trees

Xgboost is an algorithm from the family of classification and regression trees (CART). CART models at their core are decision trees where splits are made on some decision criteria based on the values of predictors. In the case of regression the data points assigned to a leaf are given a continuous prediction value, whereas in classification each leaf corresponds either to a class or a set of probabilities for the classes. The difference compared to decision trees, where the leaves only contain the decision values, is that CART models assign a score to each leaf.

Decision trees

Assume we have data (x_i, y_i) with J input variables, N data points and K response classes, where $i = 1, 2, ..., N$, $x_i = (x_{i1}, x_{i2}, ..., x_{iJ})$ and where y_i are a series of one hot encoded vectors $y_i = (y_{i1}, y_{i2}, ..., y_{iK})$ taking the value 1 whenever the observation corresponds to the true class and 0 otherwise. We perform binary partitions of our data that splits it into M regions $R_1, R_2, ..., R_M$ (corresponding to leaf nodes of the tree). In a decision tree the response in a particular region R_m can be modelled as

(Hastie, Tibshirani, and Friedman, 2001)

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m), \tag{3.5}$$

where c_m is a constant. The best estimate $\hat{c}_m$ of this constant in a region depends on the type of criterion we are minimizing when performing binary partitions. In multiclass classification this generally is the cross-entropy objective $\sum_{i=1}^{N}\sum_{k=1}^{K} I(y_i = k)\log(\hat{p}_k(x_i))$, where $\hat{p_k}$ is the predicted probability. Classification trees may either output probabilities or assign observations in a node m to the class with highest relative frequency $k(m) = \arg\ \max_k \hat{p}_{mk}$, where the relative frequencies $\hat{p}_{mk}$ are computed as

$$\hat{p}_{mk} = \frac{1}{N} \sum_{x_i \in R_m} I(y_i = k).$$

The automatic split finding algorithm of decision trees starts with a root node containing all observations. Its goal is to identify the splitting variable j and split point s of a predictor variable which minimizes the chosen objective. The data is then split into two regions (Hastie, Tibshirani, and Friedman, 2001)

$$R_1(j, s) = \{X|X_j \leq s\} \quad \text{and} \quad R_2(j, s) = \{X|X_j > s\}. \tag{3.6}$$

This step is repeated until either all predictions in a node are correct (the node is pure), or a predefined stopping criterion is met where for example the objective no longer improves enough from further splits.

Xgboost

Xgboost is a tree ensemble method, where prediction scores from multiple different trees are added together to form a prediction $\hat{y}_i$. We follow the paper of the authors of the algorithm (Chen and Guestrin, 2016) in describing it.

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \ \ f_k \in \mathcal{F}. \tag{3.7}$$

Here, $\mathcal{F}$ is the set of regression trees. The prediction $\hat{y}_i$ is the result of summing K additive functions. f_k is an independent tree structure whose leaves contain leaf weights w which can be seen as the scores of the leaves. The goal is to minimize the objective function

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k), \tag{3.8}$$

where, $\Omega(f) = \gamma T + \frac{1}{2}\lambda||w||^2$, is a regularization term penalizing model complexity T (number of leaves). The second term in the regularization involving w penalizes the magnitude of the weights to prevent overfitting on training data. The λ and γ are regularization hyperparameters of the model which can be chosen by the user. The loss function is denoted by l.

In practice all possible trees cannot be tried. The optimization objective in equation 3.9 needs to be reframed in a way where the model is additively trained. What was already learned up until the previous tree $t - 1$ is treated as fixed. In the current

iteration t, a new tree is trained and the previous prediction $\hat{y}_i^{t-1}$ is adjusted by the output of the new tree $f_t(x_i)$ with the goal of minimizing the loss

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(\hat{y}_i^{(t-1)} + f_t(x_i), y_i) + \sum_k \Omega(f_t). \tag{3.9}$$

Xgboost learns a variety of tree structures because the purpose of each new tree is to adjust the predictions of a previous tree rather than train a new one from the same initial conditions. In practice a second order Taylor expansion of the loss in equation 3.9 is used, expressed in terms of the first and second order gradients of the loss function

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_i^2(x_i)] + \sum_k \Omega(f_t), \tag{3.10}$$

$$\begin{aligned} \text{where } g_i &= \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \\ h_i &= \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}). \end{aligned}$$

In this thesis we have a multiclass response, meaning a softmax objective is used. Softmax is a function with multiple inputs and outputs. Its purpose is to normalize the vector of outputs (denoted by $\vec{z}$ in eq. 3.11) by the model for K classes into a probability distribution where the elements sum up to 1.

$$Softmax(\vec{z}) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{3.11}$$

Chen, Singh, et al. (2015) provide a diagonal upper bound of the function's hessian matrix which is used to calculate scores in Xgboost. The official documentation does not clearly state what loss function is used (the vague term "softmax objective" is used). From online discussions it appears the upper bounds of these first and second order gradients of the softmax function constitute the loss.

Splits and variable importance

CART models allow us to compute variable importances and assess whether the addition of metadata variables helped improve predictive performance. The optimal weight (i.e. the score) of a leaf can be calculated in terms of only the first and second order derivatives g_i and h_i. Here we sum those derivatives over all observations in the leaf j in a given fixed tree t

$$w_j^{*(t)} = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}. \tag{3.12}$$

Each tree learns unique weights because the gradients depend on the output predictions of the previous model. Important variables used to perform splits from previous iterations provide less of a gain in the objective when they are re-used multiple times, encouraging the algorithm to learn a variety of tree structures. The objective function can also be written to express the *gain* in score from performing a split of a node in the current tree

$$\mathcal{L}_{split} = \frac{1}{2} \left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma \tag{3.13}$$

where the brown part of the equation represents the node score before a split was performed (I denoting the node), and the blue parts denote the score after a split to left (I_L) and right (I_R) leaves from the parent node I. Subtracting the score of the unsplit node from the split nodes gives us a measure of the gain in the objective from performing the split on a given variable. The variables leading to the highest average gain across all tree structures are considered more important since they lead to the greatest reduction in the objective function.

Hyperparameters and training rounds

The model was trained for 61 rounds with the exact split finding algorithm (`tree_method = 'exact'`), a learning rate `eta` of 0.2, max tree depth of 11 and subsampling of rows fraction set to 0.7. Subsampling of observations was done as a way to combat overfitting on training data. The training rounds and learning rate were tuned to achieve a good trade off between training speed and model performance. This meant choosing the highest possible learning rate which achieved similar performance on validation set as lower learning rates. The number of training rounds were then chosen according to where validation performance leveled off and plateaued. Maximum tree depth was tuned to a depth where validation performance no longer improved by increasing it.

3.4 Semi-supervised classification

Semi-supervised classifiers generally use a single or a few labeled examples as a starting point. The training then proceeds to incorporate unlabeled data whose labels the algorithm is confident in being able to predict. In particular, the proposed semi-supervised algorithm below is similar to algorithms belonging to the self-training or self-labeling paradigm (Triguero, Garcıa, and Herrera, 2015). It aims to use a single training example to classify other confident examples. The most confident unlabeled examples are successively incorporated in to the computations as the single ground truth reference is expanded to a "weighted average embedding" of similar observations.

3.4.1 Rolling temporal classifier

A rolling temporal classifier is proposed in order to classify section titles starting from a single ground truth example. Section titles must re-occur at least weekly during a 3 month period to be considered section titles. Their regular appearance in newspapers should allow one to start with an example of a section title, and look at which images in the following pages of a newspaper are most similar.

A possible application of the algorithm is to speed up the labeling of section titles. The annotator in this scenario would be supplied with a folder of predictions in the form of images, and may be tasked with deleting the incorrect predictions from the folder. The correctness of image predictions are relatively easy for a human to verify – either the image class predictions contain the section title of interest or they do not.

Table 3.3 shows how the data input to the proposed algorithm is organized. The algorithm assumes a dataset ordered by publication date. The dataset contains $1 : T$ scanned newspaper pages. Each page t in turn contains $1 : J_t$ observations (textboxes). Every textbox has an associated embedding vector $\vec{x}_{t,j}$. Using these embeddings, a normalized similarity measure $\hat{d}_{t,j}$ is calculated between a ground truth example $\vec{q}$ to each textbox embedding vector $\vec{x}_{t,j}$ extending k pages forward in time. All textboxes within a page t are then internally ranked according to their similarity to the ground truth example. Among the highest ranked images of each page in the first k pages, the n_w examples most similar to the ground truth are chosen to be weighted together as a new "weighted average embedding" $\hat{q}$.

To compute a set of weights which sum to 1, the softmax function is applied to the vector of similarities among the n_w most similar observations (eq. 3.14). The softmax also serves to give an increased weight to those observations which are more similar to the ground truth.

$$w_i = \frac{e^{d_i}}{\sum_{j=1}^{n_w} e^{d_j}} \tag{3.14}$$

The algorithm then proceeds towards pages beyond k, comparing textboxes in these pages against the weighted average embedding. Similarities are again computed and if the most similar example on a page has a greater similarity measure than the mean of the previous n_w most similar examples, it is added to the weighted average. The output of the algorithm is a ranking of the textboxes most similar to the weighted

reference embeddings and their corresponding similarities. The top ranked predictions may optionally be classified to the ground truth image class (binary classification). The rest of the predictions are left as "unknown". The classifier is re-run and separately trained for each class, using a ground truth example from the relevant class.

The classification step is performed by only considering the observations ranked number 1 on each page. These are subsequently sorted by their similarities. If the ground truth label occurs n times in the test set, then the top n observations are predicted as the ground truth label.

As a similarity measure we suggest using cosine similarity because it is bounded between -1 and 1. If other similarity measures are used, they first need to be normalized to a suitable range to avoid overflow in the softmax function.

t	j	$\vec{x}_{t,j}$	y
1	1	$\vec{x}_{1,1}$	`non-section`
1	2	$\vec{x}_{1,2}$	`kultur`
⋮	⋮	⋮	⋮
1	J_1	$\vec{x}_{1,J_1}$	`non-section`
2	1	$\vec{x}_{2,1}$	`non-section`
2	2	$\vec{x}_{2,2}$	`non-section`
⋮	⋮	⋮	⋮
2	J_2	$\vec{x}_{2,J_2}$	`non-section`
⋮	⋮	⋮	⋮
T	1	$\vec{x}_{T,1}$	`sport`
T	2	$\vec{x}_{T,2}$	`non-section`
⋮	⋮	⋮	⋮
T	J_T	$\vec{x}_{T,J_T}$	`non-section`

(a) Data before algorithm is applied is ordered after publication date (first page of the oldest newspaper being page 1, and last page of most recent newspaper page being page T).

t	j	$\vec{x}_{t,j}$	$\hat{d}_{t,j}$	$r_{t,j}$	y
1	2	$\vec{x}_{1,2}$	0.95	(1)	`kultur`
1	43	$\vec{x}_{1,43}$	0.67	(2)	`non-section`
⋮	⋮	⋮	⋮	⋮	⋮
1	18	$\vec{x}_{1,18}$	0.12	(J_1)	`non-section`
2	11	$\vec{x}_{2,11}$	0.23	(1)	`non-section`
2	36	$\vec{x}_{2,36}$	0.21	(2)	`non-section`
⋮	⋮	⋮	⋮	⋮	⋮
2	18	$\vec{x}_{2,18}$	0.09	(J_2)	`non-section`
⋮	⋮	⋮	⋮	⋮	⋮
T	1	$\vec{x}_{T,1}$	0.86	(1)	`sport`
T	22	$\vec{x}_{T,22}$	0.54	(2)	`non-section`
⋮	⋮	⋮	⋮	⋮	⋮
T	8	$\vec{x}_{T,8}$	0.22	(J_T)	`non-section`

(b) Table after algorithm has calculated similarities and ranks of every observation to the (weighted average) reference embedding. Ordered by similarity or rank within each page t.

Table 3.3: In table (3.3a) the observations are ordered by page number and textbox index. Algorithm 1 adds similarity measures (table 3.3b) of all the observations against a reference embedding where the ground truth in this example is `kultur`. These similarities are ranked within each page. If an optional classification is performed, only the highest ranked textboxes within every page are considered (i.e. $r_{t,j} = 1$).

Pseudo-code for the algorithm is provided below (algorithm 1).

Algorithm 1: Rolling temporal classifier

Inputs:

k : start window length in pages

n_w : number of most similar obs to weight together in start window

$(\vec{q}, d)$: A ground truth example where $\vec{q} \in \mathbb{R}^m$ is the embedding vector of m features, and d its normalized similarity to itself (i.e. 1).

Data: $\{\vec{x}_{t,j} : t = 1, ..., T; j = 1, ..., J_t\}$. A set of embedding vectors where each vector $\vec{x}_{t,j} \in \mathbb{R}^m$ is indexed by page number t and textbox index j (m is number of features). Data contains T total pages. Textboxes per page, J_t, can vary.

Output: $\{(\hat{d}_{t,j}, r_{t,j}) : t = 1, ..., T; j = 1, ..., J_t\}$. The similarity metrics $\hat{d}_{t,j}$ of each textbox's embedding to the reference embedding ($\vec{q}$ or $\hat{q}$), and the similarity ranks $r_{t,j}$ of the textboxes ranked according to $\hat{d}_{t,j}$ within each page t.

Procedures: $similarityRanking(\hat{d}_t, q)$: takes a set of cosine similarities $\hat{d}_t$ of textboxes within a page (i.e $\hat{d}_{t,1}, ..., \hat{d}_{t,J_t}$) and returns ranks $r_{t,1}, ..., r_{t,J_t}$ where the most similar textbox to a reference embedding $\vec{q}$ is ranked (1) and the least similar is ranked (J_t) for all textboxes within page t.
$softMax(S)$: Applies softmax function (see eq. 3.14) to input.

$r \leftarrow \varnothing$

/* S and $\boldsymbol{X}$ stores the ground truth, and later on also similarities and corresponding embedding vectors of the most similar observations to the ground truth. Used to incorporate confident unlabeled data. */

$S \leftarrow d$ // Initialize with ground truth cosine similarity to itself, 1×1.

$\boldsymbol{X} \leftarrow \vec{q}$ // Ground truth embedding vector, $1 \times m$

for $t \leftarrow 1$ **to** k **do**

  for $j \leftarrow 1$ **to** J_t **do**

    $\hat{d}_{t,j} \leftarrow \frac{\vec{q} \cdot \vec{x}_{t,j}}{||\vec{q}|| \, ||\vec{x}_{t,j}||}$ // Cosine similarity

  $r \leftarrow r \cup similarityRanking(\hat{d}_t, \vec{q})$

  for $j \leftarrow 1$ **to** J_t **do**

    if $r_{t,j} = 1$ **then**

      $S \leftarrow S \cup \hat{d}_{t,j}$ // Row vector $1 \times k$ ($\cup$ appends columnwise)

      $\boldsymbol{X} \leftarrow \boldsymbol{X} \cup \vec{x}_{t,j}$ // Matrix $k \times m$ ($\cup$ appends rowwise)

Jointly sort $\boldsymbol{X}$ (by rows) and S according to similarities in S (descending order). Subset the top n_w rows from $\boldsymbol{X}$ and the top n_w similarity elements from S.

// S now has dimension $1 \times n_w$ and matrix $\boldsymbol{X}$ $n_w \times m$

$\boldsymbol{w} \leftarrow softMax(S)$

$\hat{q} \leftarrow \boldsymbol{w}\boldsymbol{X}$ // Weighted average embedding of labeled and confident unlabeled data

for $t \leftarrow k + 1$ **to** T **do**

  for $j \leftarrow 1$ **to** J_t **do**

    $\hat{d}_{t,j} \leftarrow \frac{\hat{q} \cdot x_{t,j}}{||\hat{q}|| \, ||x_{t,j}||}$

  $r \leftarrow r \cup similarityRanking(\hat{d}_t, \hat{q})$

  for $j \leftarrow 1$ **to** J_t **do**

    /* $\hat{d}_{t,(1)}$ notation refers to similarity with highest rank on page t */

    if $r_{t,j} = 1$ ***and*** $\hat{d}_{t,(1)} \geq mean(S)$ **then**

      $S \leftarrow S \cup \hat{d}_{t,(1)}$

      $\boldsymbol{X} \leftarrow \boldsymbol{X} \cup \vec{x}_{t,j}$

      $\boldsymbol{w} \leftarrow softMax(S)$

      $\hat{q} = \boldsymbol{w}\boldsymbol{X}$

3.5 Evaluation metrics

A majority of observations in the data are made up of the category `non-section title`. A naïve classifier predicting all observations to that class would yield a very high accuracy. Therefore we instead report precision and recall scores for each class separately when evaluating our supervised classifiers. The F_1 score is also reported and used to provide a balanced overall measure of precision and recall. The evaluation of the semi-supervised classifier is evaluated using precision.

3.5.1 Precision

Precision can be viewed as the proportion of correctly predicted examples of a given class out of all predicted examples of the same class.

$$Precision = \frac{TP}{TP + FP}. \tag{3.15}$$

In a multiclass setting, using a confusion matrix where the rows represent the predicted class and columns the true class, this corresponds to dividing the diagonal element with the sum of all elements in the row.

	Class A	Class B	Class C
Class A	5	1	2
Class B	0	3	0
Class C	0	1	4

Table 3.4: Divide the diagonal element (blue) with the sum of all elements in the row to compute precision for class B.

Precision can reveal the proportion of predictions of a given label which we expect to match the true label when applying a classifier on test data.

3.5.2 Recall

Recall provides a measure for the proportion of correctly predicted examples of a class out of all existing true labels of the class. It answers the question of what proportion of true labels were detected (i.e. classified correctly as that label).

$$Recall = \frac{TP}{TP + FN} \tag{3.16}$$

Assuming columns represent true labels, recall can be calculated for a given class by dividing the diagonal element with the sum of all elements in the same column.

	Class A	Class B	Class C
Class A	5	1	2
Class B	0	3	0
Class C	0	1	4

Table 3.5: Divide the diagonal element (blue) with the sum of all elements in the column to compute precision for class B.

3.5.3 Macro F_1 score

The F_1 score is an evaluation metric that weights together the true positive rate (Recall) with the rate of correctly predicted examples among all predicted examples of a class (Precision) in a single score.

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3.17}$$

In a multiclass setting we calculate F_1 scores for each class using our already computed precision and recall values. The class specific F_1 scores are averaged to obtain a macro F_1 score.

3.6 Bootstrapping

Bootstrapping was used to estimate standard errors of evaluation measures to better understand the uncertainty in model performance. 50 bootstrap samples were generated. KNN and XGBoost models were trained on each of the bootstrap samples and used to predict on both the in-sample and the extra-sample out of bag observations.

Suppose we have data $y_1, ..., y_n$ from a c.d.f. F. We are interested in estimating the performance θ of a classifier through the metric P (precision or recall). In bootstrap, sampling is performed from the empirical distribution function $\hat{F}$ (i.e. the data sample). We sample B times with replacement from $\hat{F}$. Each sample b is of the same size n as the data set. A test set is constructed out of the set of observations which do not get included in the bootstrap sample b (Davison and Hinkley, 1997).

For each bootstrap sample b the performance measures $P^*_{b\,train}$ and $P^*_{b\,test}$ are computed based on the predictions of a model trained on b. $P^*_{b\,train}$ is computed from predictions made on the same training data set b, whereas $P^*_{b\,test}$ is computed on the out-of-bag test samples. Using either of these measures to estimate θ is problematic. $P^*_{b\,train}$ is calculated from a model trained on the same data it is predicting on. It also contains duplicates of observations since samples were taken with replacement. These factors combined bias $P^*_{b\,train}$ towards overestimating model performance. Conversely, $P^*_{b\,test}$ is biased towards underestimating model performance – as our training set b contains duplicate observations rather than all unique observations.

Hastie, Tibshirani, and Friedman (2001) suggest performing the .632-correction to alleviate this bias. The correction proposes weighting together training and test performance according to

$$P^*_b = 0.368 \cdot P^*_{b\,train} + 0.632 \cdot P^*_{b\,test}. \tag{3.18}$$

The constants in the correction arise because only a proportion of approximately 0.632 of the elements in the data set are expected to be used in a given bootstrap sample b.

Our repeated estimates of P_b^* constitute a sampling distribution of the evaluation metric P. The standard error is calculated as $SE(\hat{P}) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(P_b^* - \bar{P})^2}$, where $\bar{P}$ is the mean of the evaluation measure estimates across all B bootstrap samples.

4. Results

The results are presented separately for each studied typographic design period. We compare the classification performance of the supervised classifiers Xgboost and K-nearest neighbour using precision and recall. These measures are presented for each section title and the category `non-section`. We also present a macro F_1 score as a measure of overall performance.

4.1 Time period 1990-1997

The models in general performed worse on older newspapers. The older editions showed more signs of discoloration and wearing. They also tended to have notes and markings scribbled in ballpoint pen which occasionally interfered with OCR. Furthermore, the typographic design in 1990-1997 was less standardized and clean as compared to the more modern newspaper designs.

4.1.1 Precision

Precision gives us a measure of the proportion of correctly predicted labels of a given class among all predictions of that class. The vast majority of observations belong to the category `non-section`. In a real world setting where these classifiers are applied to non-labelled data we expect some actual true section titles to be predicted as `non-section`. Such misclassifications will tend to be more difficult to spot for humans verifying the results compared to misclassications on a minority class, because a small number of section titles will be hidden among a large number of `non-section` predictions.

	Xgboost		**KNN**		
Section	Mean	SE	Mean	SE	**n**
non-section	0.9986	0.0002	0.9993	0.0002	64737
inrikes	0.9662	0.0160	0.8961	0.0337	131
utrikes	0.9714	0.0220	0.8839	0.0428	83
marginalen	0.9945	0.0229	0.8663	0.0864	20
vädret	0.9774	0.0308	0.9010	0.0540	44
stockholm	0.9569	0.0517	0.7678	0.0657	46
politik	0.9646	0.0285	0.9229	0.0403	64
tv	0.9819	0.0445	0.8207	0.0777	36
namn_familj	0.9703	0.0434	0.8002	0.0679	44
brännpunkt	0.9511	0.0506	0.7783	0.0830	36
stockholmsguiden	0.9427	0.1101	0.7298	0.1067	14
bridge			0.6235	0.1295	10
sidanfem					10
samtider			0.7606	0.1463	11
kultur					8

Table 4.1: Precision with associated standard errors (SE) for Xgboost and K-nearest neighbours. Cells in the table were left empty when one ore more bootstrap samples had undefined precision values (division with zero).

Table 4.1 above reveals KNN has higher precision in the `non-section` category, whereas Xgboost has higher precision for all other section titles. While Xgboost may appear better, one should again consider the difficulty of spotting the misclassified observations. If a human were to check and correct the results – it would take less time to manually correct the KNN results. Nonetheless, Xgboost displays an impressive precision, with its predictions being correct 95% of the time or more for all of the classes.

Precision seems to decrease for section titles which have fewer labeled examples. The uncertainty is generally higher for the classes with fewer examples. Labeling more examples could likely help improve model performance and stabilize the results.

4.1.2 Recall

Recall provides a measure for the proportion of true labels of a class which were detected by the algorithms. Xgboost is better at correctly identifying and classifying instances of the `non-secion` category, whereas KNN is better at ensuring a higher proportion of the true section titles are among its predictions of section titles. In general KNN is more aggressive at making a higher number of section title predictions which benefits it in terms of recall (however, its precision in the predictions are lacking).

	Xgboost		**KNN**		
Section	Mean	SE	Mean	SE	**n**
non-section	0.9999	0.0000	0.9986	0.0002	64737
inrikes	0.9453	0.0194	0.9599	0.0199	131
utrikes	0.9197	0.0359	0.9426	0.0311	83
marginalen	0.9077	0.0876	0.9662	0.0476	20
vädret	0.8347	0.0793	0.9026	0.0579	44
stockholm	0.7777	0.0671	0.8613	0.0634	46
politik	0.9013	0.0478	0.9497	0.0383	64
tv	0.7178	0.0810	0.8462	0.0746	36
namn_familj	0.7447	0.0712	0.8330	0.0555	44
brännpunkt	0.8280	0.0710	0.8333	0.0905	36
stockholmsguiden	0.6799	0.1468	0.8938	0.1059	14
bridge	0.3849	0.0685	0.7755	0.1488	10
sidanfem					10
samtider			0.8694	0.1331	11
kultur			0.6526	0.1923	8

Table 4.2: Recall.

Again, the uncertainty in the estimates increase for section titles with fewer labeled examples. Model performance likely varies depending on which examples were sampled to the training set in a given bootstrap sample. Adding more labeled examples would help stabilizing model performance on unseen data.

4.1.3 Macro F_1 score

A balanced overall measure of recall and precision is the macro F_1 score. Table 4.3 shows Xgboost overall has a higher mean performance compared to KNN.

Model	Mean	SE
Xgboost	0.8861	0.0203
KNN	0.8620	0.0198

Table 4.3: Macro F_1-score with standard errors during the period 1990-1997.

4.1.4 Feature importance

Xgboost also supplies measures of variable importance. Most of the important variables turn out to be embeddings (all variables beginning with V). However, vertical positional information `y` proves to be the most important variable for the time period 1990-1997. Since we only considered the topmost parts of the page in annotation the importance of `y` is somewhat expected. The rather wide uncertainty intervals of many embedding variables suggests there may be correlation among the variables. The algorithm can choose to perform the splits on different variables while still gaining predictive power.

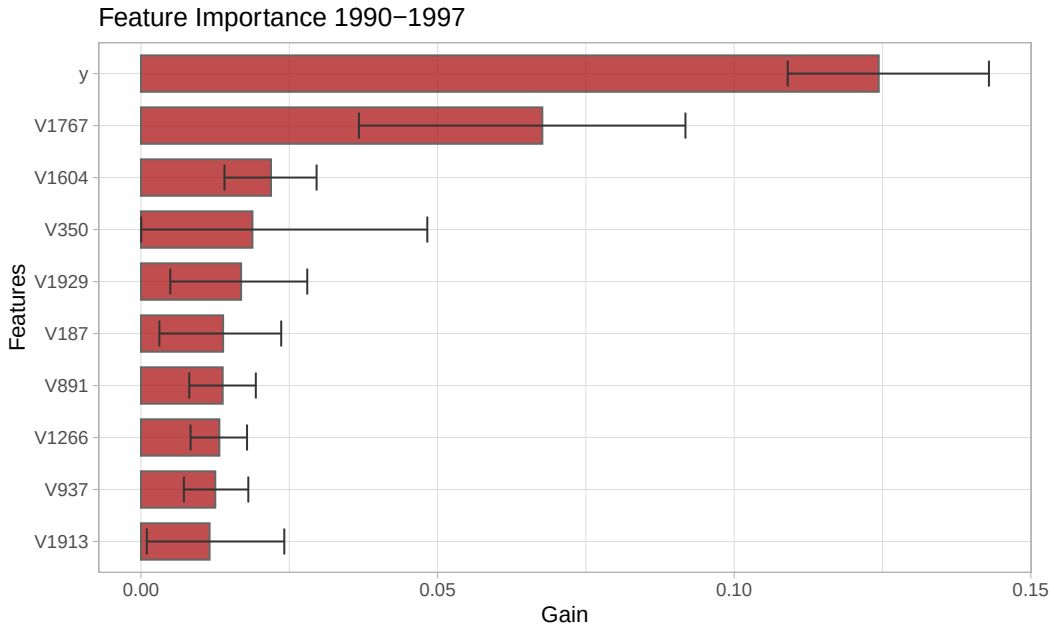


Figure 4.1: Feature importances from Xgboost for the period 1990-1997. Bars denote 10th and 90th percentile uncertainty intervals.

If the dimensionality of the data is of concern for computational reasons, a feature reduction or selection could likely be performed without large drops in model performance.

4.2 Time period 2004-2013

The period 2004 to 2013 saw the supervised models perform better than 1990 to 1997. The physical newspaper copies which were scanned were of higher quality. The design was also more streamlined with less `non-section` textual content in the vicinity of section titles. A number of factors still complicated segmentation quality however. One being that section titles often had a short catchy header right beside them describing the contents of larger articles or reports found below. Subsequently, both the header and the section title were often – but not always – segmented in one and the same textbox. These headers also interfered with segmentation quality, occasionally leading to partially cropped section titles.

4.2.1 Precision

The differences in terms of precision between the two algorithms remain even when evaluated on a different time period. KNN has better precision for `non-section` content, whereas Xgboost is more precise at predicting the section titles. For section titles such as `synpunkt`, `ledare`, `svd_guiden`, `idag` and `sidan2` its predictions are correct 99% of the time or more.

	Xgboost		**KNN**		
Section	Mean	SE	Mean	SE	**n**
non-section	0.9988	0.0002	0.9994	0.0001	64437
sport	0.9480	0.0128	0.8773	0.0204	345
svd_guiden	0.9952	0.0172	0.9580	0.0305	38
ledare	0.9926	0.0161	0.9602	0.0300	57
nyheter	0.9687	0.0118	0.9338	0.0158	412
utrikes	0.9429	0.0200	0.8784	0.0190	230
idag	0.9916	0.0173	0.8893	0.0470	59
synpunkt	0.9974	0.0127	0.9618	0.0385	34
brännpunkt	0.9591	0.0336	0.9153	0.0466	53
namn_familj	0.9707	0.0175	0.8994	0.0377	93
kryss_söndag			0.8844	0.1041	15
reportaget	0.9617	0.0655	0.8882	0.0988	15
sidan2	1.0000	0.0000	0.8609	0.1080	16
helg			0.7630	0.0978	16
special					10

Table 4.4: Precision.

The standard errors of the precision estimates do not vary as much compared to the `1990-1997` period. Fewer training examples seem sufficient to achieve good performance for many of the categories. This is likely a result of segmentation quality being better, as well as the physical newspaper copies being of higher quality during this period.

4.2.2 Recall

Both the classifiers perform well in terms of recall. Xgboost has higher sensitivity in the proportion of true `non-section` textboxes it detects, but also outperforms KNN in detecting some section titles such as `brännpunkt` and `nyheter`. KNN again is better at detecting most other section titles.

	Xgboost		**KNN**		
Section	Mean	SE	Mean	SE	**n**
non-section	0.9996	0.0001	0.9981	0.0002	64437
sport	0.9465	0.0166	0.9621	0.0111	345
svd_guiden	0.9665	0.0500	1.0000	0.0000	38
ledare	0.9432	0.0361	1.0000	0.0000	57
nyheter	0.9657	0.0094	0.9594	0.0114	412
utrikes	0.9298	0.0254	0.9403	0.0175	230
idag	0.8841	0.0550	0.9273	0.0295	59
synpunkt	0.8729	0.0699	0.9577	0.0523	34
brännpunkt	0.9567	0.0389	0.9256	0.0536	53
namn_familj	0.8975	0.0409	0.9459	0.0319	93
kryss_söndag	0.5934	0.1559	0.8292	0.1366	15
reportaget	0.7880	0.1502	0.9514	0.0691	15
sidan2	0.8443	0.1494	1.0000	0.0000	16
helg	0.4915	0.1275	0.8803	0.1011	16
special					10

Table 4.5: Recall.

A larger degree of uncertainty is still present in the recall estimates of the section titles that have fewer labeled examples.

4.2.3 Macro F_1 score

XGboost again has a slightly higher average macro F_1 score. Compared to the 1990-1997 period, both models perform better.

Model	**Mean**	**SE**
Xgboost	0.9359	0.0182
KNN	0.9246	0.0152

Table 4.6: Macro F_1 score.

4.2.4 Variable Importance

Three of the variables added in addition to the embeddings show up in the top ten feature importance plot. The vertical pixel position `y` is number one again. Splitting on `font_size` and `height` also seems to occasionally lead to large gains in the optimized objective function. However, both variables display large uncertainty in the gain from one bootstrap sample to another. This indicates certain feature embeddings may pick up on the mentioned characteristics in the image. The gain of `font_size` and `height` then fluctuate in different model runs depending on whether or not those correlated variables have already been included in the tree when the variable in question is used in splits.

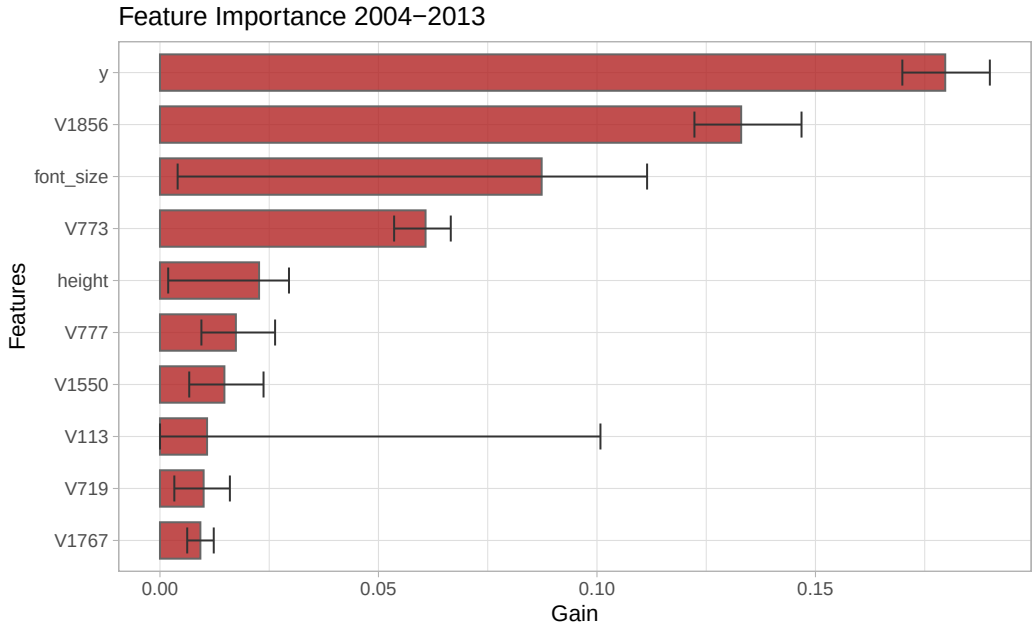


Figure 4.2: Feature importances with 10th and 90th percentile interval bars.

4.3 Time period 2017 and onward

In 2017-2019 the newspaper got a design with large and clear section titles. They were always placed at the top of pages, with very little to no clutter in terms of other text located in the same general area. The performance of the models in terms of precision, recall and F_1 was the best during this period.

4.3.1 Precision

Both classifiers have very high precision for the majority of classes. KNN again has slightly higher precision when it comes to `non-section` labels, whereas XGboost generally bests KNN when it comes to the section titles. Xgboost performs close to perfect on this time period.

	Xgboost		**KNN**		
Section	Mean	SE	Mean	SE	**n**
non-section	0.9995	0.0002	0.9999	0.0001	49835
kultur	0.9858	0.0094	0.9746	0.0134	184
korsord	0.9742	0.0222	0.9612	0.0329	52
nyheter_inrikes	0.9920	0.0055	0.9734	0.0117	263
understrecket	0.9960	0.0138	0.8750	0.0607	36
idag	0.9853	0.0262	0.9753	0.0250	37
nyheter_utrikes	0.9951	0.0049	0.9668	0.0175	207
tvradio	0.9979	0.0065	0.9571	0.0213	108
nyheter	0.9992	0.0056	0.9452	0.0528	41
bioprogram	0.9979	0.0149	0.9140	0.0599	33
sport	0.9551	0.0475	0.8836	0.0626	34
familj	1.0000	0.0000	0.9515	0.0454	38
vinmat	0.9975	0.0125	0.9553	0.0506	23
ledare	0.9971	0.0088	0.9657	0.0283	61
debatt	0.9966	0.0119	0.9926	0.0175	42

Table 4.7: Precision.

The uncertainty in the precision estimates was visibly lower during the `2017-` period compared to previous periods. The other periods generally also had some categories with fewer labeled examples. Here, the category with the fewest labeled examples was `vinmat` with 23.

4.3.2 Recall

KNN again detects a higher proportion of the annotated true section labels. Largely because it makes more section title predictions than Xgboost. Both models however perform very well for this particular design of the newspaper.

	Xgboost		KNN		
Section	Mean	SE	Mean	SE	**n**
non-section	0.9999	0.0001	0.9991	0.0001	49835
kultur	0.9879	0.0116	0.9760	0.0153	184
korsord	0.9664	0.0358	0.9932	0.0141	52
nyheter_inrikes	0.9893	0.0079	0.9928	0.0081	263
understrecket	0.9338	0.0601	0.9978	0.0091	36
idag	0.9381	0.0534	1.0000	0.0000	37
nyheter_utrikes	0.9939	0.0060	0.9929	0.0063	207
tvradio	0.9956	0.0077	1.0000	0.0000	108
nyheter	0.9782	0.0347	1.0000	0.0000	41
bioprogram	0.8519	0.1096	0.9780	0.0270	33
sport	0.9058	0.0813	0.9943	0.0180	34
familj	0.9768	0.0476	1.0000	0.0000	38
vinmat	0.9726	0.0488	1.0000	0.0000	23
ledare	0.9652	0.0313	0.9834	0.0203	61
debatt	0.9854	0.0281	1.0000	0.0000	42

Table 4.8: Recall.

4.3.3 Macro F_1 score

Out of all the different typographic designs and time periods, the models perform the best on the most recent period. Xgboost has a slight edge again in macro F_1 score – with a score of 0.98 versus the 0.97 of KNN.

Model	**Mean**	**SE**
Xgboost	0.9803	0.0076
KNN	0.9700	0.0053

Table 4.9: Macro F_1 score

4.3.4 Feature Importance

The vertical pixel position has consistently been shown to be the most important feature for all the different time periods. The `width` of the textbox also appears to be an important predictor during the typographic design implemented from 2017 and onwards.

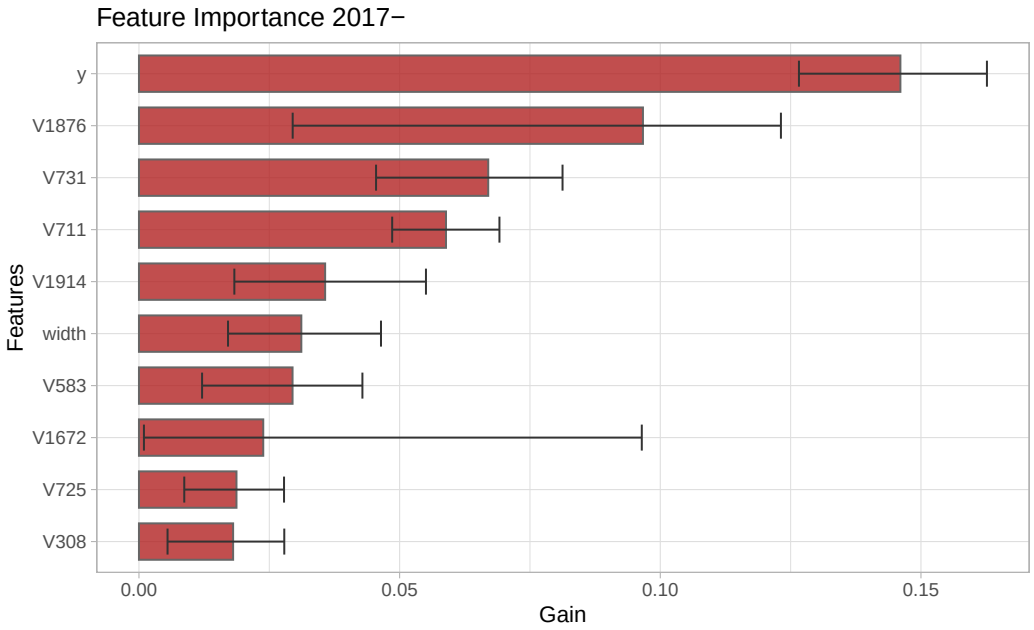


Figure 4.3: Feature importances with 10th and 90th percentile interval bars.

4.4 Rolling temporal classifier

Each textbox on a page was given a ranking with respect to their similarity to either a ground truth example or a weighted average embedding. The n most similar examples of the $rank = 1$ textboxes were predicted as the ground truth label. Below we can see how many of the section titles were correctly predicted depending on n_w: the number of most similar observations to weight together in in the starting window of $k = 500$ pages. When $n_w = 1$, only the ground truth was used (no other observations were weighted in to the average). Thus, $n_w = 1$ can be seen as the baseline upon we wish to improve.

Section	$\mathbf{n_w = 1}$	$\mathbf{n_w = 3}$	$\mathbf{n_w = 5}$	$\mathbf{n_w = 10}$
bioprogram	0.97	0.97	0.94	0.94
debatt	0.90	0.93	0.90	0.90
familj	1.00	1.00	1.00	1.00
idag	0.62	0.62	0.62	0.62
korsord	0.87	0.87	0.87	0.87
kultur	0.77	0.85	0.87	0.89
ledare	0.87	0.90	0.92	0.93
nyheter	1.00	1.00	1.00	1.00
nyheter_inrikes	0.89	0.90	0.89	0.88
nyheter_utrikes	0.61	0.60	0.57	0.58
other	0.98	0.98	0.98	0.98
sport	0.21	0.18	0.15	0.03
tvradio	0.87	0.89	0.90	0.90
understrecket	0.81	0.78	0.83	0.69
vinmat	1.00	1.00	1.00	0.00
Average	0.825	0.831	0.830	0.748

Table 4.10: Precision measures for each section title in the 2017- design period. n_w denotes the number of similar examples weighted together with the ground truth in the starting window of $k = 500$ pages.

The performance varies across different section titles. The proportion of correctly predicted labels increases for `tvradio`, `ledare` and `kultur`. Other sections show either no change, orvarying improvements with different number of unlabelled examples being incorporated in the average embedding.

Table 4.10 shows the precision measures. We can expect over 80% of labels to be of the same class as the ground truth after predictions. The labels whose scores increase are generally more homogenous in their appearance. This refers to the fact that several of the labels are combined together from section titles describing the same section but being different in appearance. For example, the `nyheter_inrikes` is a combination of section title's spelling out `Nyheter Sverige` and `Sverige Nyheter` on even and odd numbered pages respectively. The same is true for and `nyheter_utrikes`. The poor performance of `sport` can be explained by the ground truth example being an extremely large captitalized version of the section title.

5. Discussion

This thesis has outlined a general pipeline for moving from OCR metadata to an enriched feature set which includes image embeddings. While the classifiers which were tried performed satisfactorily, a degradation in performance was still observed on older newspaper data. Reliance upon OCR procedures and their segmentation is a limitation which ideally needs to be overcome if classifiers are to perform better on older newspapers. At the end of this chapter we propose improvements and extensions to our methods. First, however, we begin by discussing the methods in the context of our results.

Precision vs. Recall

An observation which was made multiple times in the results section regards the differences in precision and recall between our two classifiers. Xgboost tended towards higher precision (except for the `non-section` class), whereas KNN had higher recall. A possible explanation for the difference may lie in the objective function and the loss being optimized. Cross entropy loss punishes confidently predicting the wrong class. When class distributions are imbalanced, this generally pushes a learning algorithm towards predicting the safer option whenever there is uncertainty. The safe option for Xgboost in an uncertain situation would be to predict `non-section` because it by far is the most prevalent class.

In contrast, we have KNN, which technically cannot be described as a learning algorithm. Its "training" merely consists of storing matrices of distances to be used at prediction time. Thus, a possible explanation for why the algorithms predict differently may lie in that one is optimizing an objective whereas the other looks only at distances in feature space.

On the question of which algorithm one would recommend using, the answer should always be that it depends on how the results will be used. If the results are to be checked and corrected manually KNN would probably be preferable. K-nearest neighbours had a higher precision for the `non-section` class – meaning fewer section titles risk turning up among a sea of tens of thousands of predicted `non-section` labels. The price we pay for choosing KNN is lower precision for the section titles, as it tends to predict a section title label more often even when the textbox in fact is `non-section` title content.

Xgboost, on the other hand, may be the preferred choice if predictions are to directly be put in production and exposed to researchers at the National Library as a search and filtering tool. It makes fewer mistakes in its predictions. However, this comes at the price of lower recall, which means it does not always detect all true section titles among its section title predictions.

Feature importance and positional information

Feature importance plots showed that information regarding the vertical pixel position of a textbox was the most important feature in classification. While it may be partly expected due to the annotation focusing on the topmost parts of pages, a second way of

looking at the results involves recognizing that convolutional neural networks are quite bad at retaining global and relative positional information of objects within images.

We explained in the method how weight-sharing in CNNs lend the property of translational equivarance to networks. Goodfellow, Bengio, and Courville (2016) make the analogy of imposing infinitely strong priors on a dense fully connected network, effectively forcing the weights of neighboring units to be the same but "shifted in space". This hypothetical prior leads to the network only considering local interactions (nearby pixels) in the inputs. Such a prior is only valid insofar as the assumptions are reasonable and accurate. When only local interactions are considered this means any sort of information regarding the relative or absolute positions of distant objects within an image are essentially discarded.

Cropping the images and the loss of relative positional information may explain why the `y` variable was important to each model and had relatively low uncertainty compared to other variables. Other metadata features such as `width` and `font_size`, which also showed up in the top ten may be variables which can implicitly be captured by a CNN and be represented in the embeddings. A rather common pitfall when working with image data can for example be that a CNN learns to differentiate between different sources of images based on attributes such as aspect ratio. If the source of the images differ in their label distribution, then a network may learn to differentiate between which cameras were used to generate the images as opposed to learning from contents of the image itself.

In our case it is reasonable to suspect position has an influence on whether a textbox may be a certain section title or not. A CNN is also unlikely to capture such information due to its structure and design. Therefore any such metadata variables should ideally be concatenated to the embeddings vector at some stage of a classification process, whether it be in external classifiers as we have done or in the classification head of the CNN itself.

Representative sampling versus targeted annotation

Results indicated precision and recall benefited from an increased amount of labeled training examples. In this thesis a stratified sampling scheme was set up in order to obtain a representative sample of newspaper content on different weekdays of each sampled year. This approach, however, is not particularly effective in gathering a sufficient number of annotated examples of section titles which appear for example only on a specific weekday. The section title `Kryss_söndag` ("Crossword Sunday") is one such example which appears only on Sundays.

While a representative sample may be desirable when formally evaluating the performance of a method in an academic setting, a more time efficient approach could be to target certain section titles, setting out to specifically find and annotate more examples of them.

Annotation and semi-supervised classification

One idea to quickly generate section title candidates for labeling and annotation is to use a semi-supervised classifier such as the rolling window classifier. A list of unlabeled example images would be returned to the user after supplying only a single or a few ground truth images. The results obtained suggest over 80% of the top ranked images in such a list could be expected to be of the correct class. The major advantage of image predictions over predictions on purely tabular data is that one is often able to verrify the correctness of the former type of predictions even in the absence of true labels.

Quick labeling may involve the user being presented with a folder of predictions in image form. The user may then use image thumbnails to quickly distinguish the correct predictions from the incorrect ones. The misclassified observations could be deleted by the annotator, leaving only image predictions of the section title in the folder. The remaining images in the folder would then be assigned to the same class as the ground truth image in the dataset.

Manual annotation of section title textboxes in digitized newspapers consumes a lot of time. The annotation process in this thesis required approximately 25 hours of focused work. While the speed of annotation was helped by focusing on textboxes in the topmost parts of pages, the spreadsheets still contained tens of thousands of textboxes even after filtering (the vast majority being `non-section` content). The chosen set up required the annotator to identify section titles by clicking on columns in the spreadsheet containing hyperlinks to API-calls. These API-calls displayed the image of the relevant page and the bounding box for the corresponding textbox observation.

Time savings should be quite substantial when single training examples can yield a precision of 83%. A folder with tens to hundreds of image predictions is both easier and faster to correct compared to the process of labeling in spreadsheets with upwards of tens to hundreds of thousands of entries. An annotator would still need to use the spreadsheet to fill in the rest of the section titles – but this way most of the pages would already have labels pre-filled.

In this thesis, labels signifying the same section were combined in to one single category even if the textboxes differed somewhat in appearance. Large capitalized versions of a section title were treated the same as smaller versions of the same title in lower case. The domestic news section for 2017-2019 was comprised of `nyhetersverige` and `sverigenyheter`, yet in the classification they were treated as the same category `nyheter_inrikes`. For the classification algorithms, this did not generally constitute a problem since there were enough labeled training examples of each different appearance. However, if a semi-supervised classifier is to be used with only single examples, some care should be taken to treat each appearance as a unique problem.

Contrasting results and methods against previous research

In section 1.2 two different approaches for classifying newspaper content were mentioned. One focused on the textual contents whereas the other directed its attention towards the layout and appearance of newspaper pages. This thesis used the second

approach to retrieve and recover already present section title labels assigned by the newspaper editors. That approach can however only work in periods where newspapers use and display section titles. Their usage generally becomes less consistent the further back in time we go. The need is thus still present for approaches which focus on textual content.

The addition of image embeddings in the classification process appears to have been successful. Precision and recall in the high 80s and 90s for almost all section titles outperforms the comparable studies using text based methods on newspapers. There is not much research to compare to when it comes to classifying section titles based on metadata features and images. Hurtado Bodell et al. (n.d.) used metadata features to classify whether a textbox was a section title or not and achieved an F_1 score of 0.63. This thesis' addition of image embeddings to metadata features managed to achieve macro F_1 scores of 0.886, 0.936 and 0.980 in a multiclass setting.

Suggestions for future research

The methods used in this thesis do not constitute an end-to-end pipeline for classification. Currently they rely upon metadata features generated from OCR software. The reason we are able to crop a box of text is because it first has been correctly detected and segmented as such by the OCR software.

Future development of the methods presented here should ideally strive towards

1. making the method more robust against poor quality in segmentation.
2. finding a way to incorporate the segmentation step in an end-to-end modeling pipeline which functions independently of OCR metadata.

A first option in making models more robust against poor segmentation may be to train a CNN. The data to train such a network could be generated in part by using the proposed methods in this thesis. To simulate poor segmentation, one may randomly add or subtract pixels of `height` and `width` when cropping images. This way a large amount of augmented training images could be generated. After generating sufficient training data one begins training the CNN – fine-tuning its weights to be robust against segmentation that varies greatly in quality. This approach could potentially improve the quality of the image embeddings; especially for older newspapers.

A second option would be to learn an object recognition model to perform one's own segmentation of section titles. Training data may be generated the same way as described above. However, this time the goal is to gather as much training data as possible which has already been well segmented by OCR software. The `x`, `y`, `width` and `height` metadata attributes can be used to construct bounding boxes (i.e. marking the areas in an image which belong to a specific class) and train an object detection R-CNN model (Ren et al., 2015) to specifically detect and segment section titles. A white paper implementing such an approach on historical newspaper data was published by researchers at the Library of Congress (Lee et al., 2020) a couple of weeks before the finalization of this thesis.

6. Conclusion

Image embeddings were shown to be an effective addition to metadata features when classifying section titles. The highest performing Xgboost model achieved F_1 scores of 0.866, 0.936 and 0.980 for the time periods 1990-1997, 2004-2013 and 2017- respectively. It was found that KNN in general achieved a higher recall for section titles, whereas Xgboost attained a higher precision. Choosing one model over the other should depend on whether predictions are to be manually reviewed and corrected (KNN) or be directly put in to production (Xgboost).

The performance of models declined for older newspapers. The decrease in performance is likely related to a deterioration in the quality of the physical newspaper copies, but also in part because of the more complex design of older newspapers, where other text content is frequently found in the vicinity of section titles. The discussion chapter suggested ways to remedy the deterioration in performance. Image embeddings can be made more robust to variations in segmentation quality by fine-tuning the weights of the CNN – training it to classify section titles while using augmented images which were purposefully cropped poorly. A second avenue for improvement comes from implementing and training a custom segmentation model as opposed to relying on the OCR software's segmentation for cropping images.

Lastly, classification based on single labeled examples of section titles was found to have some potential in aiding the annotation and labeling of data. Annotation was time consuming mostly due to the large fraction of `non-section` textboxes present in the newspapers and the time needed to identify the correct textboxes in a spreadsheet. Among the top ranked predictions resulting from using a single ground truth example, over 80% proved to be of the correct class. Since image predictions can be verified by simply looking at images, these results could be used to cut down annotation time by several hours.

Appendices

A. Annotation

For the year 1990-1997 all of the labels containing "`tv`" were combined in to one category called `tv`. All observations of `namn_familj`, `namn` and `familj` were combined in to one category called `namn_familj`. The lone instance of `väder` was changed to `vädret`. The section categories chosen for modeling were bridge, brännpunkt, inrikes, kultur, marginalen, namn_familj, politik, samtider, sidanfem, stockholm, stockholmsguiden, tv, utrikes and vädret. All others were assigned to the non-section category.

label	n
bridge	10
brännpunkt	36
dagensnamn	2
debatt	1
dialog	2
europa	4
familj	11
festival	1
idag	1
inrikes	131
isjälhjärta	2
kultur	8
marginalen	20
namn	3
namn_familj	27
politik	64
radio	2
restaurangguiden	1
samtider	11
scenen	2
schack	4
seriekrypto	8
sidanfem	10
skola	1
sport	4
sportisiffror	1
sportlov	2
stockholm	46
stockholmsguiden	14
söndag	2
söndagsnamn	1
tv	4
tv_programmen	15
tvradio	1
tävlingar	2
utrikes	83
veckans_tv	1
vetenskap	5
väder	1
vädret	43
övriga_tv	15
non-section	64609

The years 2004-2013, `namn` and `familj` were combined to a single category called `namn_familj`. The categories brännpunk, namn_familj, helg, idag, kryss_söndag, ledare, nyheter, reportaget, sidan2, special, sport, svd_guiden, synpunkt and utrikes were chosen for modeling. The rest were labeled as non-section.

label	n
brännpunkt	53
familj	86
helg	16
idag	59
korstvärs	4
kryss_söndag	15
krysslösningar	4
ledare	57
namn	7
nyheter	412
reportaget	15
resultat_spelguide	1
sidan2	16
special	10
sport	345
svd_guiden	38
synpunkt	34
tävlingar	4
utrikes	230
vetenskap	2
non-section	64422

Table A.1: Raw annotations 2004-2013

In the National Library of Sweden's database, newspapers had been scanned and OCRd up until May of 2019. The sampled editions from time period `2017-` thus extend only until May 2019.
`idag` and `idagsidan` were combined to `idag`. Any label containing "`kultur`" was combined to the category `kultur`. `ledare` and `ledare_large` were coded as `ledare`, and `sport` and `sport_large` were coded as `sport`. Domestic news `sverigenyheter` and `nyhetersverige` were labeled as `nyheter_inrikes`, whereas international news `världennyheter` and `nyhetervärlden` were labeled as `nyheter_utrikes`. All other content was labeled as non-section.

label	n
banner_date	619
banner_text	141
header	17
large_text	86
medium_text	96
other	164
section_bioprogram	33
section_debatt	42
section_familj	38
section_idag	14
section_idagsidan	23
section_korsord	52
section_kultur	147
section_kultur_large	35
section_kulturspråk	2
section_ledare	10
section_ledare_large	51
section_nyheter	41
section_nyhetersverige	144
section_nyhetervärlden	126
section_sport	32
section_sport_large	2
section_språkkultur	3
section_svd	7
section_sverigenyheter	119
section_tvradio	108
section_understrecket	36
section_vinmat	23
section_världennyheter	81
text	91
non-section	48611

Table A.2: Raw annotations for 2017-

Bibliography

Allen, Robert B., Weizhong Zhu, and Robert Sieczkiewicz (2010). "What to Do With a Million Pages of Digitized Historical Newspapers?" Presented at iConference. URL: http://hdl.handle.net/2142/14932.

Bilgin, A. et al. (2018-10). "Utilizing a Transparency-Driven Environment Toward Trusted Automatic Genre Classification: A Case Study in Journalism History". In: *2018 IEEE 14th International Conference on e-Science (e-Science)*, pp. 486–496. DOI: 10.1109/eScience.2018.00137.

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag: Berlin, Heidelberg. ISBN: 0387310738.

Chen, Tianqi and Carlos Guestrin (2016). "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. Association for Computing Machinery: San Francisco, California, USA, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785.

Chen, Tianqi, Sameer Singh, et al. (2015-09–12 May). "Efficient Second-Order Gradient Boosting for Conditional Random Fields". In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Guy Lebanon and S. V. N. Vishwanathan. Vol. 38. Proceedings of Machine Learning Research. PMLR: San Diego, California, USA, pp. 147–155. URL: http://proceedings.mlr.press/v38/chen15b.html.

Czarniawska, Barbara (2014). *A Theory of Organizing: Second edition*. 2nd. Edward Elgar Publishing. ISBN: 978-1848444300.

Davison, A. C. and D. V. Hinkley (1997). *Bootstrap Methods and their Application*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press. DOI: 10.1017/CBO9780511802843.

di Lenardo, I., B. Seguin, and F. Kaplan (2016). "Visual Patterns Discovery in Large Databases of Paintings". In: *Digital Humanities 2016: Conference Abstracts*. Jagiellonian University Pedagogical University, pp. 169–172.

García-Mendoza, Consuelo-Varinia and Omar Gambino Juárez (2018). "News Article Classification of Mexican Newspapers". In: *Telematics and Computing*. Ed. by

Miguel Felix Mata-Rivera and Roberto Zagal-Flores. Springer International Publishing: Cham, pp. 101–109. ISBN: 978-3-030-03763-5.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Harbers, Frank and Juliette Lonij (2017). "Distinguishing Newspaper Genres. Exploring Automated Classification of Journalism's Modes of Expression". In: *Digital Humanities 2017*.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc.: New York, NY, USA.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2(5), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8.

Hurtado Bodell, Miriam et al. (n.d.). "Classifying and Curating the Historical Swedish Newspaper Corpus". unpublished.

Lee, Benjamin et al. (2020-05). "The Newspaper Navigator Dataset: Extracting And Analyzing Visual Content from 16 Million Historic Newspaper Pages in Chronicling America". In: URL: https://arxiv.org/pdf/2005.01583.pdf.

Lindén, J., S. Forsström, and T. Zhang (2018-09). "Evaluating Combinations of Classification Algorithms and Paragraph Vectors for News Article Classification". In: *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 489–495.

Lonij, Juliette and Melvin Wevers (2016). *SIAMESE*. KB Lab: The Hague. URL: http://lab.kb.nl/tool/siamese (visited on 2020-02-02).

Nerone, John and Kevin G. Barnhurst (1995-02). "Visual Mapping and Cultural Authority: Design Changes in U.S. Newspapers, 1920–1940". In: *Journal of Communication* 45(2), pp. 9–43. DOI: 10.1111/j.1460-2466.1995.tb00726.x.

Olah, Chris, Alexander Mordvintsev, and Ludwig Schubert (2017). "Feature Visualization". In: *Distill*. DOI: 10.23915/distill.00007.

Ren, Shaoqing et al. (2015). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Proceedings of the 28th International Conference on*

Neural Information Processing Systems - Volume 1. NIPS'15. MIT Press: Montreal, Canada, pp. 91–99.

Russakovsky, Olga et al. (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115(3), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

Snickars, Pelle (2018). *Datalabb på KB: en förstudie*. National Library of Sweden. URL: https://www.kb.se/dokument/Bibliotek/utredn_rapporter/2018/1.2.1-2017-752.pdf (visited on 2020-01-20).

Tan, Mingxing and Quoc V. Le (2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *ICML*.

Triguero, Isaac, Salvador Garcıa, and Francisco Herrera (2015-02). "Self-Labeled Techniques for Semi-Supervised Learning: Taxonomy, Software and Empirical Study". In: *Knowledge and Information Systems* 42(2), pp. 245–284. ISSN: 0219-1377. DOI: 10.1007/s10115-013-0706-y.

Wang, Yan et al. (2019). "SimpleShot: Revisiting Nearest-Neighbor Classification for Few-Shot Learning". In: *arXiv preprint arXiv:1911.04623*.

Wu, Hui-Yin and Pierre Kornprobst (2019-07). *Multilayered Analysis of Newspaper Structure and Design*. Research Report RR-9281. UCA, Inria. URL: https://hal.inria.fr/hal-02177784.

Yosinski, Jason et al. (2014). "How transferable are features in deep neural networks?" In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., pp. 3320–3328. URL: http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf.