A Pipeline for Digitizing Dutch TV Guides Using Mask R-CNN and OCR

Ying-Kai Dang
Information and Computing Sciences
Utrecht University
Utrecht, Netherlands
y.p.dang@students.uu.nl

***Abstract*—This research focuses on the development of an end-to-end pipeline for digitizing Dutch TV guides from archived collections, using a combination of Mask R-CNN for article segmentation and Optical Character Recognition (OCR) for text extraction. The project was conducted in collaboration with Beeld en Geluid, the Dutch cultural archive, aiming to digitally transform their physical TV guide collections into searchable and analyzable digital formats. The pipeline includes three main stages: article segmentation using Mask R-CNN, OCR using Tesseract to convert visual text to machine-encoded text, and post-processing to structure and clean the OCR outputs. This method allows for the preservation and efficient analysis of historical TV programming data.**

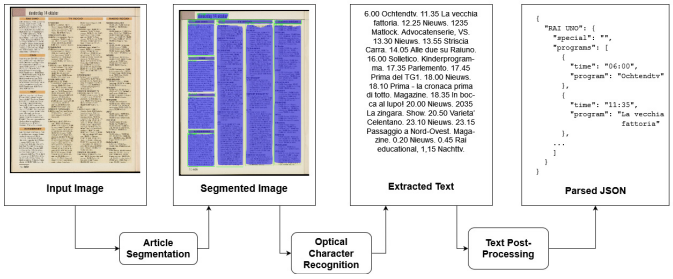


Fig. 1. End-to-end pipeline

I. Introduction

Beeld en Geluid, the Dutch cultural archive and museum in Hilversum, has been collecting and digitizing a lot of physical media. Among these documents, television guides (TV guides) offer a cultural snapshot of their times and allow for researching the evolution of television programming, as they provide a comprehensive listing of what was broadcast. However, the task of digitizing these archives, particularly through Optical Character Recognition (OCR), poses some challenges due to the specific layouts of the TV guides.

Each TV guide page may contain the title, date, broadcast channel, TV program, and other relevant components. While the contents of an entire page can be retrieved in a rather straightforward manner, such information retrieval does not allow for capturing the relation of individual components of the page together in a structured form, such as tying a TV program to the broadcast channel it was broadcast on. It is therefore important to first extract each component separately (e.g. the article body and the article title) to be able to find the correct structure, which can be converted into a format that can be queried digitally.

For this report, Beeld en Geluid has provided a selection of their archived TV guides and television broadcast recordings to create a mapping between them. This allows them to perform various forms of data examination, such as verifying the consistency between the scheduled programs outlined in the TV guides and the actual broadcasts, or to study the evolution of the content.

To achieve the aforementioned goal, the work of this report aims to create an end-to-end pipeline that takes a TV guide page as input and generates a structured JSON file. The pipeline consists of three parts: Article segmentation, OCR, and text post-processing.

Article segmentation involves dividing the TV guide page into distinct sections that contain different types of information. This segmentation is crucial for identifying and isolating individual components, namely the headings and the body. For this task, Mask R-CNN [1], which uses ResNet50 as backbone [2] that was pre-trained on the COCO dataset [3], was employed to segment the regions of interest (ROI). It segments the regions both with a mask and a bounding box. While Mask R-CNN is not a state-of-the-art (SOTA) model for newspaper segmentation [4], it was used here due to the narrow range of the data used in the project and the ease of use of the model. For this project, the segmentation mask predictions themselves are not used in the final output, as all relevant components to this project have a rectangular shape and therefore using only the predicted bounding boxes is sufficient.

Following segmentation, OCR is applied to each segmented part of the page to translate the visual information into machine-encoded text using Tesseract OCR [5] without any modifications or fine-tuning of the model.

The final step, text post-processing, involves cleaning and structuring the raw, machine-encoded text output from OCR. This step is essential for removing errors introduced during OCR, such as misinterpreted characters or words, and for structuring the text in a way that accurately reflects the original layout and content of the TV guide. Algorithms and heuristics are used to parse the OCR output into a structured JSON format, identifying and correctly labeling each piece

of information (program names, times, broadcast channels) according to its role and significance in the document.

The end-to-end pipeline, visualized in Figure 1, transforms the static images of TV guide pages into searchable, and analyzable digital records. This structured digital format opens up numerous possibilities for research and analysis, from tracking changes in television programming and popularity over time to understanding cultural shifts reflected in TV content. Moreover, by automating the digitization process as much as possible, the archive at Beeld en Geluid can efficiently process larger volumes of archival material.

In Section II we explore the existing tools and methodologies, mainly around newspaper segmentation. Section III examines the data that was provided by Beeld en Geluid, as well as additional data that was used for training the model. In Section IV we go through the pipeline that was developed during this project, elaborating on the techniques used for data annotation, preprocessing, article segmentation, OCR, and text post-processing. Methods that were tested but were eventually discarded are discussed in Section V. Subsequently, Section VI assesses the effectiveness of the methods used. Finally, Section VII proposes paths that further research could go down, especially how SOTA models would improve this pipeline.

II. Related Work

To annotate the data and create a ground truth, the *Visual Geometry Group Image Annotator* (VIA) annotation software [6] was used. The VIA software is a very lightweight, standalone, offline website that was developed by the Visual Geometry Group of the Department of Engineering Science at the University of Oxford and has been very popular in both industrial as well as academic settings [6]. We chose VIA as annotation tool, as our data is quite simple and requires no complex annotations. Additionally, since VIA is highly accessible due to its small size and it being readily distributable, an easy integration into the workflow was possible.

A. Datasets

Several datasets for benchmarking document segmentation exist, of which three widely used datasets are PubLayNet [7], DocLayNet [8], and M^{6}Doc [9].

1) PubLaynet: PubLayNet is a dataset containing 360,000 document images from PubMed. All images in PubMed come from PDF documents, which allows for the extraction of word-level OCR to automatically create annotations. PubLayNet distinguishes between 5 categories: Text, Title, List, Table, and Figure. However, due to the nature of scientific documents, the variance in their layouts is very limited, as the typeset templates are usually provided by the publishers.

2) DocLaynet: DocLayNet attempts to overcome PubLayNet's limitation by introducing diverse and complex layouts from different public sources and manually annotating them. It contains 80,863 manually annotated pages and includes six document categories: Financial Reports, Manuals, Scientific Articles, Laws and Regulations, Patents, and Tenders. DocLayNet distinguishes between 11 categories: Caption, Footnote, Formula, List-item, Page-footer, Page-header, Picture, Section-header, Table, Text, and Title. However, it is criticized that the document categories considered in DocLayNet are not commonly used and therefore not a good fit for training generalizable models [9].

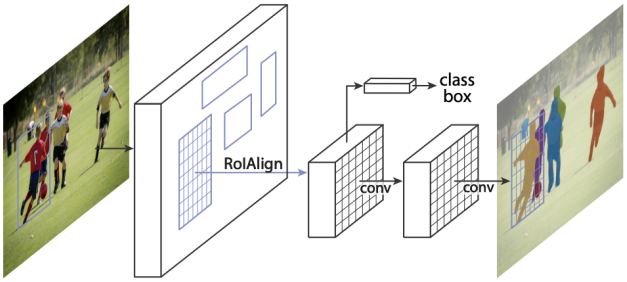


Fig. 2. Mask R-CNN framework

3) M^{6}Doc: M^{6}Doc introduces a diverse dataset that aims to provide a comprehensive dataset that can be used to train models that generalize well to real-world scenarios. It includes 9,080 document images from 7 document categories: Scientific articles, Textbooks, Books, Test papers, Magazines, Newspapers, and Notes. M^6 represents the six properties multi-format, multi-type, multi-layout, multi-language, multi-annotation, and modern documents. It is also the first dataset to consider both the general and unique aspects of documents by categorizing them into 74 distinct categories.

B. Document segmentation models

A wide variety of image segmentation models exist, of which many can be used in document segmentation as well [10].

1) Traditional methods: Newspaper segmentation has been under research for a long time already, with the First International Newspaper Segmentation contest [15], which was held at the ICDAR'2001 conference, kickstarting the evaluation of existing algorithms for document segmentation algorithms that can be applied to newspaper segmentation. The majority of methods used for page segmentation using traditional methods (i.e. not using neural networks) took a bottom-up approach, where the text lines and other distinctive lines were used to segment a page into its components. Another popular method was to use morphological operations, which enhance the segmentation process by identifying and structuring text and graphic elements on the page. [16]–[18]

2) Mask R-CNN: Mask R-CNN [11] builds on the Faster R-CNN [19] framework by adding a branch for predicting segmentation masks on each Region of Interest (RoI), effectively allowing for pixel-wise segmentation. This makes Mask R-CNN a powerful tool for instance segmentation tasks, where the goal is to classify each pixel in an image not just into categories but as distinct objects. The key innovation in

TABLE I
SEGMENTATION PERFORMANCE COMPARISONS ON DOCLAYNET, PUBLAYNET, AND M^{6}DOC

		DocLayNet			PubLayNet			M^{6}Doc			
Method	Model	Section-header	Text	mAP	Title	Text	mAP	AP50	AP75	Recall	mAP
Mask R-CNN [11]	R101	69.3	85.8	73.5	84.0	91.6	91.0	<u>58.4</u>	<u>46.2</u>	<u>50.8</u>	<u>40.1</u>
YOLOv5 [12]	v5x6	<u>74.6</u>	88.1	76.8	-	-	-	-	-	-	-
SelfDocSeg [13]	-	-	-	74.3	-	-	89.2	-	-	-	-
SwinDocSegmenter [14]	Swin	66.5	<u>88.2</u>	<u>76.9</u>	<u>87.2</u>	<u>94.6</u>	<u>93.7</u>	-	-	-	-
M2Doc [4]	R101	**90.7**	**93.9**	**89.0**	**89.7**	**95.6**	**95.5**	**86.7**	**79.4**	**82.5**	**69.9**

Mask R-CNN is the RoIAlign layer, which replaces the RoI pooling in Faster R-CNN. RoIAlign preserves spatial precision accurately by using bilinear interpolation to compute the exact values of the input features at four regularly sampled locations in each RoI bin and aggregating the results. This enhancement addresses the misalignment issue caused by RoI pooling's harsh quantization, allowing for more precise segmentation.

Mask R-CNN also maintains the architecture of its predecessor, consisting of a backbone for feature extraction (such as ResNet), a region proposal network (RPN) that shares full-image convolutional features with the detection network, and two heads: one for bounding box regression/classification and another for generating binary masks. Each head operates in parallel over the proposed regions, predicting not only the class and bounding box but also a binary mask for each class among class labels. This segmentation is class-specific but not class-aware, meaning it does not differentiate between instance categories within its predictions, focusing only on the spatial distribution of objects. Non-Maximum Suppression (NMS) is applied post-prediction to refine the bounding boxes and masks, ensuring that the model predicts distinct objects.

3) YOLOv5: YOLOv5 [12] is an iteration of the You Only Look Once (YOLO) object detection family [20]. As the name suggests, YOLO only "looks at" the image once and, thanks to the regression-based detection algorithm, directly predicts bounding boxes and class probabilities without a separate region proposal network.

The YOLO algorithm differs from the conventional sliding window approach, splitting the input image into $S \times S$ grids. Each grid is responsible for predicting B bounding boxes, which are presumed to belong to the same category, along with the confidence level for C different categories. Every bounding box yields five parameters: (x, y, w, h, c), denoting the position, dimensions, and the confidence score of the bounding box, respectively. Thus, each grid generates $(B \times 5 + C)$ values. Following this, NMS is applied to eliminate redundant detections. There are three main components: the backbone, neck, and head.

The backbone focuses on extracting image features at various scales. The neck generates a feature pyramid to aggregate those image features from the backbone. Finally, the head processes these features to predict bounding boxes and object classes. Despite newer iterations and versions of YOLO existing, we included YOLOv5 in this section due to the availability of the model's performance on the previously mentioned datasets.

4) SelfDocSeg: SelfDocSeg [13] is a self-supervised vision-based model for document segmentation. It is a novel approach for document segmentation using self-supervised learning without relying on ground-truth labels or derivatives.

Unlike other self-supervised document segmentation methods that use text mining and textual labels, and are therefore language-specific, SelfDocSeg uses a vision-based strategy, generating pseudo-layouts from document images to pretrain an image encoder. This pretraining enables the encoder to learn document object representation and localization within a self-supervised framework. Finally, the encoder is fine-tuned with an object detection model. While this method does not outperform existing models, the performance is comparable [13]. However, this novel technique sets a good foundation for further research in self-supervised vision-based document segmentation methods.

5) SwinDocSegmenter: SwinDocSegmenger [14] is a transformative vision-based model designed for instance-level segmentation of document layouts. Its architecture is a Swin transformer [21] backbone for extracting multi-scale features from documents and integrates with a transformer encoder-decoder, which detects and segments the document components.

The segmentation technique begins with mask queries as anchors, using the outputs of the encoder to predict segmentation masks early. It employs contrastive training with a mixed query selection strategy, which ensures that the feature representation and localization are robust and can therefore adapt to various document layouts effectively.

6) M2Doc: M2Doc [4] is a document layout analysis tool that uses a multi-modal fusion approach, which at its core is a document segmentation model. It uses both visual and textual information to improve the segmentation of the document layout and therefore can segment different components more effectively than just using visual data, as Mask R-CNN is doing. M2Doc employs early-fusion and late-fusion modules.

The early-fusion module aligns and combines visual and textual features at the pixel level, while the late-fusion model operates at the block level. One major feature is that M2Doc is designed to be pluggable, meaning a variety of existing document layout analysis detectors can be plugged into the early-fusion and late-fusion modules.

Table I shows a comparison of the performances of the models. In bold highlighted are the best-performing metrics, and underlined are the second-best-performing metrics.

C. Optical Character Recognition

Tesseract [5], [22], EasyOCR [23], and PaddleOCR [24] are among the most popular open source OCR tools, according to the GitHub star counts.

1) Tesseract OCR: Tesseract OCR is one of the oldest and most widely adopted Optical Character Recognition (OCR) engines. Originally developed by Hewlett-Packard in the 1980s and later released as open-source in 2005, Tesseract has been sponsored by Google since 2006. It supports over 100 languages and is considered one of the most accurate free OCR engines available today. Tesseract was a purely rule-based engine until 2018 when LSTM networks were introduced in Tesseract 4.0.

2) EasyOCR: EasyOCR, developed by Jaided AI, is a more recently developed OCR tool. It is designed for simplicity and ease of use, supporting over 80 languages. EasyOCR combines Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) with connectionist text proposals to effectively handle natural scene text recognition. This tool is particularly useful for developers looking for a straightforward implementation that provides support for multiple languages, including complex scripts like Chinese, Japanese, and Korean.

3) PaddleOCR: PaddleOCR is developed by PaddlePaddle, an AI platform by Baidu. This OCR tool aims to provide easy deployments of OCR capabilities as well. It supports over 80 languages and includes algorithms for text detection, recognition, and layout analysis. PaddleOCR is distinguished by its high performance in both speed and accuracy, and it supports a wide range of applications from ID card recognition to invoice analysis.

III. Data

Part of the data are provided by Beeld en Geluid and are of the magazine *de Gids*[1], issue 41 of the year 1999. These data are not available publicly. This is also the magazine that Beeld en Geluid has archived and on which this tool is intended to run. For training the Mask R-CNN model, two additional magazine issues by *TV Guide Magazine* from the public internet were taken as well. The issues were published in August 9th 1980[2] and July 5th 1988[3].

A. TV Guide Description

Every TV guide analyzed in this project was part of a broader magazine, incorporating schedules and TV programs amidst journalistic content like interviews, editorials, and reviews. To clarify terminology, 'TV guide magazines' refers to the entire publication, while 'TV program guides' — termed simply 'TV guides' for the remainder of this report — specifically denote the sections listing TV schedules.

[1]https://www.de-gids.nl/

[2]https://archive.org/details/tv-guide-collection_202108/TV_Guide_Aug-09-1980_Small/

[3]https://archive.org/details/tv-guide-collection_202108/TV_Guide_Jul-05-11-1988_sm/

Fig. 3. TV program guide

Fig. 4. Ad

TABLE II
DATA SOURCE DISTRIBUTION. DATASET SIGNIFIES HOW MANY TV GUIDE MAGAZINE PAGES HAVE BEEN USED AS TRAIN/TEST PAGES AND TV GUIDE SIGNIFIES HOW MANY PAGES OF THE TRAIN/SET SET ARE PAGES OF TV PROGRAM GUIDES

Name	Total pages	Dataset	TV guide
de Gids, 1999, issue 41	84	74	49
TV Guide 1980-08-09	144	112	81
TV Guide 1988-07-05	212	173	120
Total	**440**	**359**	**250**

The dataset for this project includes only those pages primarily composed of text (Figure 3), excluding pages dominated by images or advertisements (Figure 4). Along with TV program guides, segments such as interviews and editorials are included due to their similar visual layout. Table II outlines how the data from these sources are distributed, focusing only on relevant pages from the TV guide magazines.

B. Data Description

Given the highly uniform nature of the dataset to which we intend to apply our pipeline, where all images bear a strong resemblance to one another with only minor deviations in the layout, we have the advantage of tailoring our model's training process with a high degree of specificity. Each image was manually segmented into its core components to set the ground truth:

1) *Title*: The main heading
2) *Text*: The text content/body
3) *Image*: Any images
4) *Ad*: Ads that have a clear border around them
5) *Subtitle*: Any other headings on the page if a main heading is present

Table III shows the distribution of all annotation types across the different magazines present in the dataset and Figure 5 shows an example of the annotated data.

Of all the data, 19% originates from *de Gids* with 11% being actual TV program guides from this source. This 11% represents the core focus of our interest, as these are the TV program guides. The remaining data, though not directly

TABLE III
DISTRIBUTION OF ANNOTATION TYPES

	Rectangle					Polygon					Total
Name	Title	Text	Image	Ad	Subtitle	Title	Text	Image	Ad	Subtitle	
de Gids, 1999, issue 41	415	531	59	6	5	4	16	0	0	0	**1036**
TV Guide 1980-08-09	96	232	11	2	1	1	5	0	0	0	**348**
TV Guide 1988-07-05	137	642	46	39	115	5	586	22	3	9	**1604**
Total	**648**	**1405**	**116**	**47**	**121**	**10**	**607**	**22**	**3**	**9**	**2988**

Fig. 5. Ground truth masks. Red are text and green are title classes.

containing TV program guides, is still useful due to its similar layout (e.g., columnar structure), which aids in identifying the relevant TV program guide pages.

IV. METHODS

A. Data Annotation

We utilized VIA software for data annotation and ground truth creation. We defined five classes for annotation: title, text, image, ad, and subtitle. Each relevant component on an image was enclosed by a rectangle. For instances with intricate text shapes, polygons were employed. To capture finer details, particularly in text regions, we used a combined approach for the 1988 TV Guide issue. Each object that issue was annotated with both a polygon and its corresponding bounding box. The increased time investment for polygon annotations limited their rigorous application to the 1988 issue.

B. Data Preprocessing

The annotations from the VIA software are used to create grayscale masks from the images. Each mask of a class within an image has a unique color value such that the masks can be recognized as separate instances. The color value distributions are as follows: ads from 1 to 10, image from 11 to 40, intro from 41 to 70, subtitle from 71 to 100, text from 101 to 200, title from 201 to 255. 0 denotes the background. This also means that on each image there is a maximum amount of instances for each class, for example there can be only 10 ads or 100 texts on one image. For our use case this is plenty, but one might want to remove this limit by using a different approach. Figure 6 shows an example of the grayscale masks.

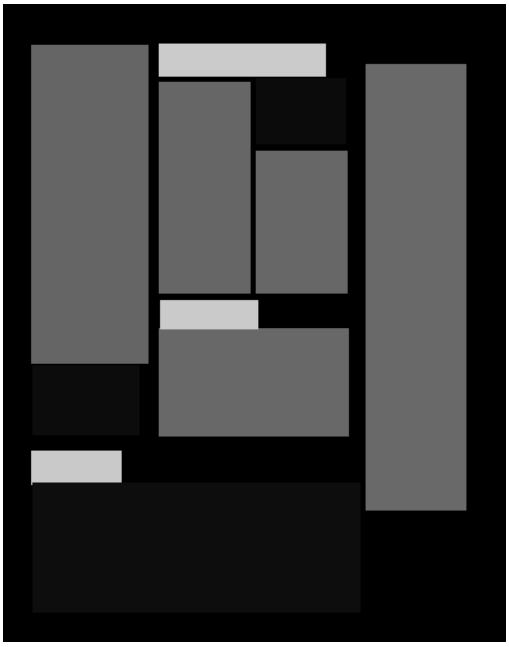


Fig. 6. TV guide page masks, where the color value of the mask is in a range based on the class the mask belongs to.

To optimize training efficiency, we compared training the Mask R-CNN model on a subset of the original-sized images versus images scaled down to a fixed height of 500 pixels. While both approaches achieved comparable performance (mAP with 50% IoU threshold at 76.9% and 76.3%, respectively), scaling offered a significant computational advantage. Consequently, all training images were downsized to a height of 500 pixels.

Furthermore, basic data augmentation techniques were employed, including horizontal, vertical, and combined flips. This process effectively quadrupled the training dataset size.

The dataset is divided into training and test sets. The test set comprises 426 images, with 60 (14%) from de Gids and 41 TV Guides from de Gids (9.5% of all test images). The

training set contains 1702 images, including 236 (13.8%) from de Gids.

C. Article Segmentation

In Section II we introduced a few different document segmentation models. Table I compares the performance of each model. Mask R-CNN ranks as one of the least effective models across all datasets, although it is only a few percentage points behind the others, except for M2Doc. Despite its relatively poor performance, we opted for Mask R-CNN in our pipeline due to its widespread availability and user-friendliness. The rationale is that the primary focus is on building the entire pipeline, so a segmentation model that performs adequately suffices for the time being, as it can be upgraded in future developments.

Following the paper by Almurairi and Almashan [1], we used Mask R-CNN to segment the TV guide instances. The base code was taken from a PyTorch tutorial [25] and adjusted accordingly to our use case.

1) Implementation Details: Firstly, a custom dataset class was created where the data were prepared for training. The images were converted to a tensor of type float and the images' pixel values were scaled to the range $[0, 1]$. This normalization step helps the model to converge more quickly and efficiently. Furthermore, target fields were also defined here, which were used for training and evaluation:

1) *boxes*: The "top left" and "bottom right" coordinates of the bounding boxes.
2) *labels*: The label as integer for each bouding box, where 0 represents the background.
3) *image_id*: An identifier for each image, unique across all images in the dataset. We use the index of the image in the dataset as the identifier.
4) *area*: The area of the bounding box, which is used for evaluation with the COCO metric to distinguish the metric scores between the differently sized boxes.
5) *masks*: The segmentation masks which were created with the VIA software.

We used a Mask R-CNN segmentation model with a ResNet-50-FPN [26] that was pre-trained on COCO [27]. The number of classes is 6, which includes the background. The pre-trained heads of the model were then replaced with new ones to adapt the model to the number of classes we were using (6), as well as allowing us to fine-tune the model on our dataset. The bounding box predictor head was replaced with a Fast R-CNN predictor using 6 classes and the mask predictor head was replaced with a Mask R-CNN predictor using 6 classes and 256 hidden layers. The batch size chosen for training was 2 and the chosen optimizer was stochastic gradient descent with learning rate = 0.005, momentum = 0.9, and weight_decay = 0.0005. The momentum helps to overcome some limitations of the basic gradient descent algorithm, such as oscillations or slow convergence in shallow or steep parts of the loss landscape. The weight decay is used as the coefficient for the L2 regularization term that is added to the loss function and controls how much to penalize larger weights in the model. A weight decay of 0.0005 means a relatively small regularization term but still contributes to preventing overfitting by encouraging the model to keep the weights small. Additionally, a learning rate scheduler is employed with step_size = 2 and gamma = 0.5, meaning the learning rate gets halved every two epochs. A batch size of 4 was used for training.

The model was trained for 10 epochs on an NVIDIA RTX 3080 GPU with 10GB VRAM.

2) Postprocessing: After training the model, all images from *de Gids* were put through the model to get the bounding boxes of the segmented instances. A confidence threshold of 85% filtered out erroneous bounding boxes, leaving us with only the bounding boxes that are truly the bounding box of an instance. Then, to determine the order of the components, we had to employ Algorithm 1 that determines the location and relation of the components between each other.

Algorithm 1: Bounding Box Order

```
Result: Sorted list of columns and bounding boxes in
        columns
MCIs = [];
for each box in boxes do
    box.topleft.y = 0;
end
for each combination of boxes (boxA, boxB, ...) do
    if intersection_exists(boxA, boxB,... then
        MCI = find_intersection(boxA, boxB, ...);
        if MCI is not subset of any other MCI then
            MCIs.append(MCI);
        end
    end
end
for each MCI in MCIs do
    MCI.corner = find_top_left_mci_corner(MCI);
    MCI.boxes.sort(box: box.topleft.y);
end
MCIs.sort(MCI: MCI.corner.x);
```

Algorithm 1 begins by normalizing the position of all bounding boxes along their y-axis, setting the top border of each box to zero. This step ensures that every box that belongs to a column is overlapping each other.

Next, the intersections among all boxes are identified, specifically the Maximum Common Intersection (MCI) for each combination of the boxes. An MCI is defined as the overlapping area shared by the maximum number of boxes. This step is important for assigning boxes to a column and assumes that there are no overlaps between two columns. This outputs a list of MCIs, each representing a group of overlapping bounding boxes, forming a column. The length of this list corresponds to the number of distinct columns present on the page.

Following the identification of columns, the algorithm calculates the top-left corner of the intersection area for each

MCI. The columns are ordered based on the x-values of these corners, creating a left-to-right sequence.

Within each sorted column, the individual articles are then further ordered based on the y-values of their top-left corners. This step ensures that within each column, the articles are arranged from top to bottom.

The result of these steps is an ordering of all bounding boxes. They are arranged first by their placement given by the corresponding column from left to right and then within each column from top to bottom. Each bounding box was saved as an image and contains the relevant metadata of its relative order on a page and the class of the bounding box.

D. Optical Character Recognition

1) Preprocessing: To prepare the segmented instance images for text extraction, a series of preprocessing steps were implemented. Initially, each image was converted to greyscale and then binarized using Otsu's thresholding method [28], which selects the optimal threshold value for pixel intensity to differentiate text from the background, resulting in an image with white text against a black backdrop. Subsequently, a 10-pixel black border was added around the image. This additional space allows for morphological operations aimed at mitigating border noise, potentially originating from partially segmented text. The applied preprocessing operations are as follows:

1) Opening, which is erosion followed by dilation, with a $(2, 2)$ kernel to remove noise, if the image is of class text. If the image is of class title, the kernel is $(3, 3)$, and the operation is repeated an additional time.
2) Closing, which is dilation followed by erosion, with a $(2, 2)$ kernel to close small holes in the text that could come from the scan of the physical media.
3) Gaussian blur is applied to smooth out the text, with a $(5, 5)$ kernel for text-class images and a kernel of $(7, 7)$ for title-class images.

The differentiation between text and title classes in applying opening and Gaussian blurring is due to titles being generally larger and bolder and can therefore withstand rougher operations.

The final step involved inverting the image to present black text on a white backdrop. We found that Tesseract [5] performs better under these conditions for our specific needs, as opposed to scenarios where the color scheme is reversed.

2) Tesseract: The resulting image is then passed to Tesseract [5] for optical character recognition. To each word that Tesseract identified, a confidence score ranging from 0% (least confident) to 100% (completely confident) was assigned, indicating the certainty of its recognition accuracy for that word. The confidence level threshold is set to 65% and any recognized word under that threshold was discarded.

The recognized words above the confidence level are joined together with white space and saved in a plaintext file.

E. Text Post-Processing

The input for the pipeline includes various pages from the TV guide magazine, not all of which contain TV program listings. To identify the relevant pages, we filtered out those that lacked program details by counting the occurrences of times listed on each page. A page is considered to contain TV program guides if it has time mentions at least 13 times. This threshold helps distinguish between actual program guide pages and other types of content within the magazine.

To structure the OCR-extracted text into our preferred format, we processed the text, addressing certain edge cases first. This discussion covers only one edge case; additional cases are detailed in Section VII. We have chosen JSON as the file format for storing the parsed data. For each TV guide page, a JSON file is created with a specified format that systematically organizes the content extracted from each page:

```
{
  "<broadcast channel>": {
    "special": "<special descriptions>"
      "programs": [
        {
          "time": "<time of program start
              >",
          "program": "<name or description
              of program>"
        },
        {
          "time": "<time of program start
              >",
          "program": "<name or description
              of program>"
        }
      ]
    }
}
```

Listing 1. JSON file format of parsed TV guide

In the magazine *de Gids* specifically, some programs that are broadcast regularly are not listed explicitly but are listed as something similar to "Elk heel uur t/m 18.00, 21.00, 22.00 en 0.00." This means that at each full hour until and including 18:00, the news program is being broadcast. After 18:00, the news broadcast will occur at 21:00, 22:00, and at midnight 00:00. To handle this special case, we used the following regex:

```
^(.*?)\.\s(\d+\.\d+.*?$)
```

This regex is designed to capture two main parts of a string: the text before the first period followed by a space, and then a sequence starting with a time format. Here is a detailed breakdown:

1) First Capturing Group `^(.+?)`:
   - `^` Asserts the start of the string.
   - `(.+?)` Captures everything in a non-greedy manner up to the first occurrence of the pattern that follows (in this case, the period followed by a whitespace). This part is enclosed in parentheses to create a

capturing group, allowing us to extract or refer to this segment separately from the whole match.

2) Literal Characters and Whitespace `\.\s`:
   - `\.` Matches the literal period character.
   - `\s` Matches any whitespace character.
3) Second Capturing Group `(\d+\.\d+.*?$)`:
   - `\d+\.\d+` Matches one or more digits (`\d+`), followed by a literal period (`\.`), and then one or more digits again. This part captures a time format, such as "12.30".
   - `.*?` Matches any character (except for line terminators) as few times as possible, until the next part of the pattern or the end of the string.
   - `$` Asserts the end of the string.

After applying that regex, we checked if the first part contains the text "Elk heel uur" or "Elk uur". Should the part contain one of those strings, we put that part into the *special* field in the JSON file. Should the part not contain any of those strings, we continue without doing anything.

Next, we parsed the remaining text into *time*s and *program*s. For that, we used the following regex, which has been split into three lines for better readability:

```
(\b\d{1,2}\.\d{2}(?:-\d{1,2}\.\d{2})?\b)
(.*?)
(?=\b\d{1,2}\.\d{2}(?:-\d{1,2}\.\d{2})?\b|$)
```

This regex is designed to capture the times and program texts by using a positive lookahead to determine when to stop the program text capture.

1) First Capturing Group: This capturing group looks for times of the format "0.00", "00.00", "0.00-0.00", or a combination of those.
2) Second Capturing Group: This capturing group looks for any text that comes after the time.
3) Positive Lookahead: The positive lookahead behaves in the same way as the first capturing group, allowing the second capturing group to stop capturing once it has hit another time.

This gives us a list of matches, where each match has two elements (the first and second capturing group). By splitting that, we were able to populate the JSON file accordingly.

V. Exploratory Methods

A. Article Segmentation

In the beginning, we tried to use traditional methods to segment the articles, as introduced in Section II-B1. We chose to go with an implementation that uses a combination of the researched methods: Relying on morphological operations to segment the articles and using distinctive lines to frame the page properly, as the page has some black borders that were introduced by scanning the TV guide magazine. This would serve us as a baseline.

However, after some deliberation, we chose to not pursue this path further, as the algorithm is very specific to the TV guide magazine *de Gids* of 1999, and any changes in the layout could break this algorithm. The general idea was to immensely dilate the text to make large "blobs", of which the contours essentially represent the article columns. As the scanned pages contain scanning artifacts such as black borders, since the scanner's scanning bed lid was not able to fully close due to the thickness of the scanned magazine, using this method was not reliable and we could not ensure that the method works when other magazines were scanned slightly different. Additionally, much better methods of article segmentation exist, as discussed in Section II-B, so time spent on using traditional methods was to be kept at a minimum.

B. Text Post-Processing

1) Large Language Model: Given the widespread availability of large language models (LLMs) today, we opted to evaluate a selection of them (ChatGPT 3.5 and 4 [29], Llama 2 [30], Mixtral 8x7b [31], and Mistral 7b [32]) to determine their effectiveness. The specific prompt used is detailed in Section IX-A of the appendix.

ChatGPT 4, Mixtral, and Llama 2 performed well, correctly segmenting all the text. However, Mistral and ChatGPT 3.5 failed to identify many of the programs in the input text. Due to the high resource demands of using LLMs, we have decided to reserve these models for secondary support, should other methods fail.

2) Named Entity Recognition: We also explored developing a Named Entity Recognition (NER) tagger to automatically classify each word by its corresponding category. We compiled a small dataset by annotating each word on two pages of a TV guide and storing this data in a TSV file. Each line in this file follows the format *TEXT*\t*CLASS*, where *TEXT* is the word and *CLASS* is the assigned category, separated by a tab. This dataset includes 1184 words categorized as *TIME*, *PROGRAM*, or *OTHER*, with a train-validation-test split of 0.7-0.15-0.15.

To create the NER tagger, we utilized the Python library spaCy [33]. The training used a whitespace tokenizer and spaCy's language processing pipeline. The tagger was trained over 100 epochs with a patience setting for early stopping at 5 epochs, using compounding minibatch sizes starting from 4 and increasing to 32 with a compound rate of 1.001, and a dropout rate of 0.5. Adam optimizer was employed, and the training ceased after 54 epochs due to early stopping, executed on an AMD Ryzen 3900X processor. Figure 7 displays the training and validation loss throughout the NER training process.

Unfortunately, the tagger's performance was underwhelming, with most words classified as *OTHER* being incorrectly labeled as *PROGRAM*. Table IV presents the precision, recall, and F1-score for each class. Given the infrequency of certain special phrases (approximately once every three pages) discussed in Section IV-E, employing a NER tagger effectively would necessitate a significant amount of training data.

VI. Evaluation

A. Article Segmentation

1) Metrics: To comprehensively evaluate the performance of object detection algorithms, several metrics are used: recall,

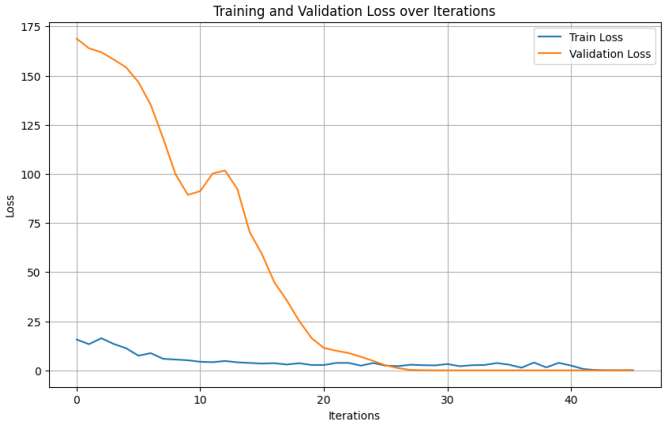


Fig. 7. Training and validation loss over epochs for NER tagger.

TABLE IV
PERFORMANCE OF THE NER TAGGER

	Precision	Recall	F1	Support
TIME	1.00	1.00	1.00	15
PROGRAM	0.77	0.96	0.86	123
OTHER	0.50	0.20	0.29	25
micro avg	0.78	0.85	0.81	163
macro avg	0.38	0.36	0.36	164

precision, Intersection over Union (IoU), average precision (AP), mean average precision (mAP), and maximum detections (maxDets). These metrics help quantify how well a model can detect objects while distinguishing between their different classes and accurately localizing them within an image [34].

Precision measures the accuracy of the positive predictions made by the model. In object detection, precision is the ratio of correctly predicted positive observations to the total predicted positives. It answers the question: "Of all the objects detected by the model, how many were actually correct?"

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \quad (1)$$

Recall measures the ability of a model to find all the relevant cases within a dataset. In object detection, this refers to the ratio of correctly predicted positive observations to all observations in actual class. It answers the question: "Of all the actual objects, how many did the model successfully detect?"

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \quad (2)$$

F1 Score is the harmonic mean of precision and recall and is a way to combine both precision and recall into a single measure that captures both properties. It is particularly useful when a balance of precision and recall is required, which is often the case in not clearly class-dominated datasets.

$$\text{F1 Score} = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

Intersection over Union is a metric used to measure the accuracy of an object detector on a particular dataset. IoU is defined as the ratio of the intersection of the predicted bounding box and the ground truth bounding box to their union.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad (4)$$

This metric evaluates the overlap between the predicted bounding box and the ground truth box. An IoU of 1 indicates perfect overlap, while an IoU of 0 indicates no overlap.

Average Precision is a measure of quality across recall levels, which is particularly useful when needing to compare the precision-recall balance of different algorithms or configurations. It is computed as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight:

$$\text{AP} = \int_0^1 p(r)dr \quad (5)$$

where $p(r)$ is the precision at recall r. Generally, AP is usually calculated for each class separately and then averaged over all classes. AP can also be computed at a particular IoU threshold, such as AP50 and AP75. For example, AP50 means the AP at an IoU of 50%. This means the bounding box predicted by the model needs to have an IoU of at least 0.5 with the ground truth box to be considered a correct detection.

Mean Average Precision is the average of the AP calculated for all classes and/or across different IoU thresholds. In datasets with multiple classes, mAP provides a way to combine the performance across all classes into a single metric:

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} \text{AP}_i \quad (6)$$

where N is the number of classes. The mAP can also be averaged over multiple IoU thresholds to provide a more robust metric that considers both lenient and strict evaluation criteria:

$$\text{mAP at IoU} = [.50 : .95 : .05] \quad (7)$$

This notation means calculating the AP at IoU thresholds from 0.50 to 0.95 with a step of 0.05 and then taking the average. This metric is used in benchmarking datasets to provide a comprehensive overview of model performance. Typically, when mAP is referenced without specifying thresholds, it is assumed to refer to mAP calculated at IoU thresholds from 0.50 to 0.95 with a step of 0.05.

maxDets is not a metric by itself but rather a parameter that is used in conjunction with other metrics. This parameter specifies the maximum number of object detections the model can produce per image which are considered in the computation of metrics like Average Precision (AP). For example, maxDets = 100 means that the AP will consider up to 100 detections per image. This limit is important in scenarios where there might be many potential detections per image, such as in crowds, and

it helps in assessing how well the model performs when it is restricted to producing a set maximum number of detections.

2) Evaluation: Table I displays a mAP of 73.5% on DocLayNet and 91.0% on PubLayNet, while our model achieves a mAP of 77.4%, as illustrated in Figure 9. Given that DocLayNet's document classes resemble our TV guides and PubLayNet features uniformly structured document layouts, we anticipated our model would outperform on our dataset relative to DocLayNet but not perform as well as on PubLayNet. The model trained by Almurairi and Almashan [1] achieved a mAP at 50% IoU of 81.6%, while our model achieved a mAP of 95.0%.

The Intersection over Union (IoU) metrics for segmentation and bounding boxes demonstrate similar performance levels, which aligns with expectations since most segmentation masks in our study were also used as bounding boxes. Notably, as shown in Figure 10, having approximately 20% of the segmentation masks in our training set be polygons is sufficient to effectively segment text blocks that incorporate text wrapping around objects.

Although the results were in line with our expectations, they were not sufficiently robust for broader application across the entire Beeld en Geluid dataset. This outcome was anticipated, as noted in Section IV-C, due to our decision not to employ any SOTA models. Figure 11 illustrates a case where the predicted segmentation masks and bounding boxes fail to cover the entire text body, resulting in lost information. Conversely, Figure 12 presents a scenario where the bounding boxes appropriately encompass the entire text body, though the segmentation masks do not fully extend across the text.

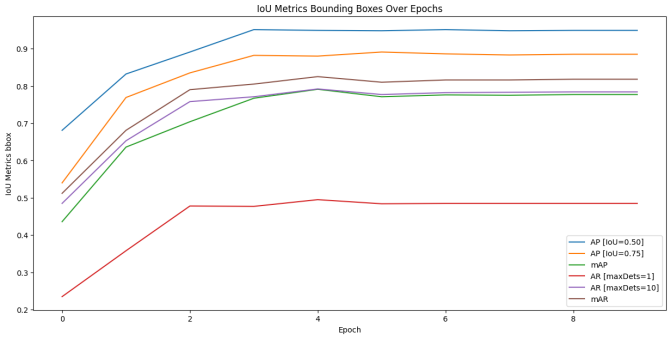


Fig. 8. Mask R-CNN IoU metrics for bounding boxes over training epochs

B. OCR

1) Metrics: Character Error Rate (CER) and Word Error Rate (WER) are commonly used metrics to evaluate the performance of OCR systems. An article published by Rose Holley [35], which evaluated OCR accuracy in historic newspaper digitization, found that "good" OCR accuracy was defined as 98-99% (or CER/WER of 1-2%), "average" as 90-98%, and "poor" as below 90%. Furthermore, they found that albeit the CER and WER are good performance measurements, it is important to note that the quality of the scan and the condition of the original document are as important as the

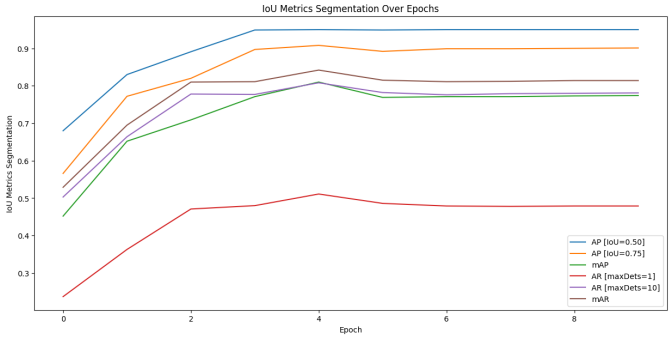


Fig. 9. Mask R-CNN IoU metrics for segmentation masks over training epochs

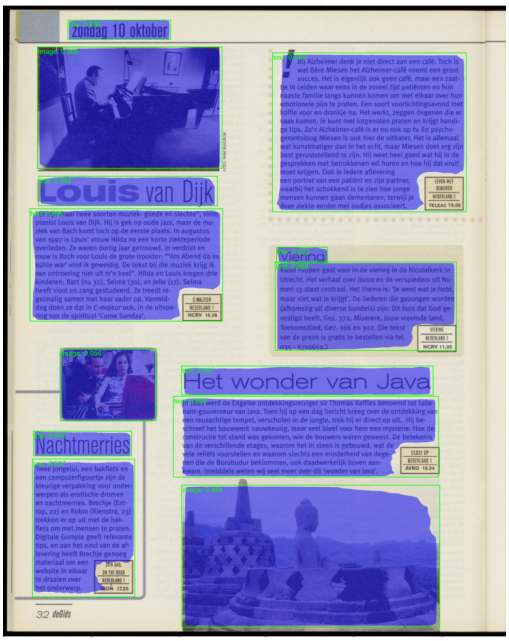


Fig. 10. Bounding box and segmentation mask prediction on non-TV guide page

OCR model. *Character Error Rate* measures the minimum number of insertions, deletions, or substitutions, also called Levenshtein or Edit Distance [36], required to change the transcription generated by the system into the correct version.

$$\text{CER} = \frac{\text{Levenshtein Distance}}{\text{Number of Characters in the Reference}} \tag{8}$$

Word Error Rate is similar to CER but works on the word level.

$$\text{CER} = \frac{\text{Levenshtein Distance}}{\text{Number of Words in the Reference}} \tag{9}$$

2) Evaluation: Due to the presence of unique characters such as a boxed letter B, which is not standardized in Unicode, Tesseract often misidentifies these symbols. For instance, it might interpret a boxed 'B' as "B]" or "[B". Additionally, characters like lowercase 'i' and 'l' are occasionally confused, especially if there are artifacts. Only a small ground truth dataset of about 300 words was created, devoid of special

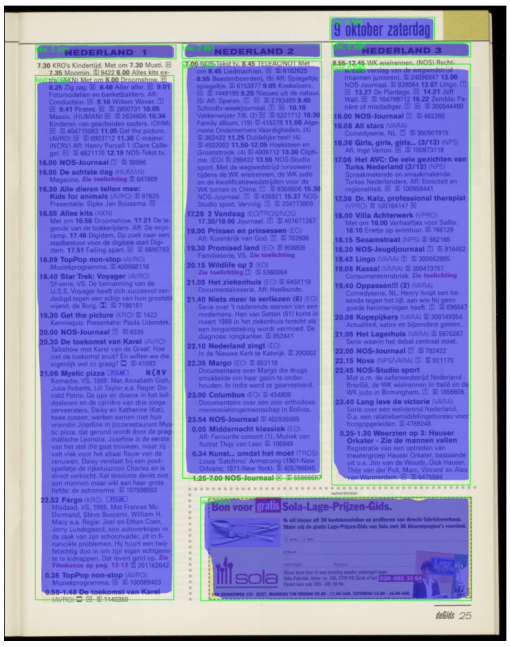


Fig. 11. Example of badly predicted segmentation masks and bounding boxes

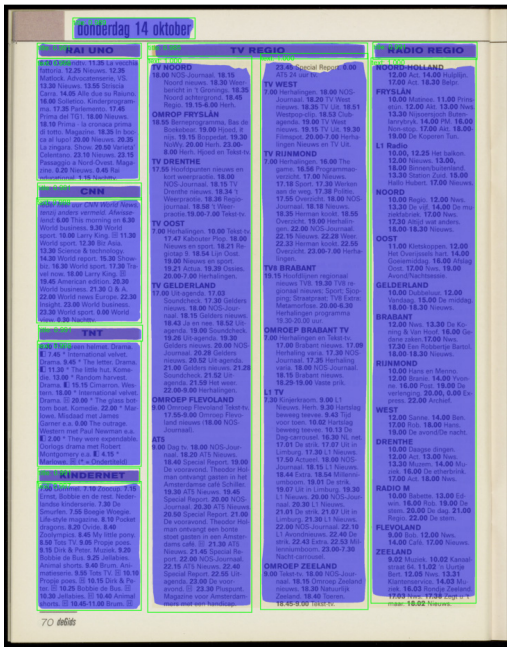


Fig. 12. Example of correctly predicted segmentation masks and bounding boxes

symbols, which limits the ability to conduct a thorough evaluation of OCR accuracy. Despite this, the CER achieved was 0.81% with 16 errors, and the WER was 2% with 6 errors.

C. Text Post-Processing

The rule-based approach to parsing text generally performed well, accurately extracting all times and programs from the OCR-processed text. Nonetheless, there were a few challenges. Sometimes, the section header, intended to identify the broadcast channel, would label it with a broader category like "TV Regio" rather than the specific channel name, which would instead appear within the text body. An example of this is seen in Figure 14. Moreover, the reliance on regular expressions to detect time formats presented additional complications. Times embedded within program names were mistakenly recognized as separate program start times.

VII. Future Work

A. Article Segmentation

Since no SOTA models were used for article segmentation, there is substantial room for improvement in this area. Implementing a SOTA model like M2Doc [4] could significantly enhance segmentation performance. M2Doc has shown superior results on the M^{6}Doc dataset, a collection featuring a wide array of document layouts, compared to the performance achieved by Mask R-CNN on the same dataset. This suggests a potential for large improvement in our segmentation accuracy and reliability by adopting more advanced models like M2Doc.

Currently, we utilize only the bounding boxes to extract text from images using OCR. However, incorporating a more advanced segmentation model could significantly enhance our capabilities. With a better model, we could extract text from more complex structures, not just the simple areas defined by bounding boxes. This would allow for more precise and comprehensive extraction of text, capturing nuances in document layouts that bounding boxes alone might miss or might take in too much.

B. Optical Character Recognition

Section VI-B2 highlights the presence of unique special characters in the TV guides, which are not accurately recognized by the Tesseract model in its default configuration. Enhancing the OCR accuracy could be achieved by fine-tuning Tesseract specifically on these special characters. This fine-tuning would allow for the correct recognition of these characters, reducing the need for extensive post-processing and potentially streamlining the entire text extraction workflow.

C. Text Post-Processing

The sequence of program times needs careful consideration to ensure continuity. For example, the listing "8.00 news, 9.00 sports, 11.00 1.00 euro lunch program, 13.00 news..." requires discerning that "1.00 euro" in a program title should not be confused with a time. Furthermore, times that reset at midnight (0.00) should be treated as continuing from the previous day, implying a logical transition from 23.00 to

0.00 as 24.00 to maintain sequence. Additionally, extracting times like "om ca. 22.03" necessitates additional processing to handle approximations indicated by "ca."

VIII. Conclusion

The developed pipeline for digitizing Dutch TV guides through Mask R-CNN and OCR effectively converts physical archives into accessible digital formats, enabling researchers to search and analyze historical TV programming. Despite its successes, the current system exhibits limitations, particularly in the accuracy of article segmentation. These issues sometimes result in errors or incomplete data capture, pointing out the need for improvements in the methodologies used. The integration of SOTA models in place of Mask R-CNN could potentially increase precision in segmenting page layouts enormously. Additionally, enhancing OCR capabilities by training on domain-specific examples could further reduce errors and improve text recognition. While the pipeline functions adequately for the current archive processing goals, achieving higher accuracy and efficiency remains a priority for ensuring that the digitized data is as informative and useful as possible.

References

[1] A. Almutairi and M. Almashan, "Instance Segmentation of Newspaper Elements Using Mask R-CNN," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, Dec. 2019, pp. 1371–1375. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8999273

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015, arXiv:1512.03385 [cs]. [Online]. Available: http://arxiv.org/abs/1512.03385

[3] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common Objects in Context," Feb. 2015, arXiv:1405.0312 [cs]. [Online]. Available: http://arxiv.org/abs/1405.0312

[4] N. Zhang, H. Cheng, J. Chen, Z. Jiang, J. Huang, Y. Xue, and L. Jin, "M2Doc: A Multi-Modal Fusion Approach for Document Layout Analysis," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 7, pp. 7233–7241, Mar. 2024, number: 7. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/28552

[5] R. Smith, "An Overview of the Tesseract OCR Engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2, Sep. 2007, pp. 629–633, iSSN: 2379-2140. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4376991?casa_token=cYF6LioJgs0AAAAA:uXzqt21uoI5kXaIAkpzdpBhHTMe9BuZhBhgjKuyoJhx4naWlDCC5xvgqjeJF28EY65Z6hy1ICusX026Q

[6] A. Dutta and A. Zisserman, "The VIA Annotation Software for Images, Audio and Video," in *Proceedings of the 27th ACM International Conference on Multimedia*. Nice France: ACM, Oct. 2019, pp. 2276–2279. [Online]. Available: https://dl.acm.org/doi/10.1145/3343031.3350535

[7] X. Zhong, J. Tang, and A. Jimeno Yepes, "PubLayNet: Largest Dataset Ever for Document Layout Analysis," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, Sep. 2019, pp. 1015–1022, iSSN: 2379-2140. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8977963

[8] B. Pfitzmann, C. Auer, M. Dolfi, A. S. Nassar, and P. Staar, "DocLayNet: A Large Human-Annotated Dataset for Document-Layout Segmentation," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '22. New York, NY, USA: Association for Computing Machinery, Aug. 2022, pp. 3743–3751. [Online]. Available: https://dl.acm.org/doi/10.1145/3534678.3539043

[9] H. Cheng, P. Zhang, S. Wu, J. Zhang, Q. Zhu, Z. Xie, J. Li, K. Ding, and L. Jin, "M6Doc: A Large-Scale Multi-Format, Multi-Type, Multi-Layout, Multi-Language, Multi-Annotation Category Dataset for Modern Document Layout Analysis," 2023, pp. 15 138–15 147. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2023/html/Cheng_M6Doc_A_Large-Scale_Multi-Format_Multi-Type_Multi-Layout_Multi-Language_Multi-Annotation_Category_Dataset_CVPR_2023_paper.html

[10] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image Segmentation Using Deep Learning: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3523–3542, Jul. 2022, conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9356353

[11] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," 2017, pp. 2961–2969. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2017/html/He_Mask_R-CNN_ICCV_2017_paper.html

[12] Ultralytics, "Comprehensive Guide to Ultralytics YOLOv5." [Online]. Available: https://docs.ultralytics.com/yolov5

[13] S. Maity, S. Biswas, S. Manna, A. Banerjee, J. Lladós, S. Bhattacharya, and U. Pal, "SelfDocSeg: A Self-supervised Vision-Based Approach Towards Document Segmentation," in *Document Analysis and Recognition - ICDAR 2023*, G. A. Fink, R. Jain, K. Kise, and R. Zanibbi, Eds. Cham: Springer Nature Switzerland, 2023, pp. 342–360.

[14] A. Banerjee, S. Biswas, J. Lladós, and U. Pal, "SwinDocSegmenter: An End-to-End Unified Domain Adaptive Transformer for Document Instance Segmentation," in *Document Analysis and Recognition - ICDAR 2023*, G. A. Fink, R. Jain, K. Kise, and R. Zanibbi, Eds. Cham: Springer Nature Switzerland, 2023, pp. 307–325.

[15] B. Gatos, S. Mantzaris, and A. Antonacopoulos, "First International Newspaper Segmentation contest," in *Proceedings of Sixth International Conference on Document Analysis and Recognition*, Sep. 2001, pp. 1190–1194. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/953973

[16] A. Antonacopoulos, B. Gatos, and D. Bridson, "ICDAR2005 page segmentation competition," in *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, Aug. 2005, pp. 75–79 Vol. 1, iSSN: 2379-2140. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1575513

[17] ——, "Page Segmentation Competition," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2, Sep. 2007, pp. 1279–1283, iSSN: 2379-2140. [Online]. Available: https://ieeexplore.ieee.org/document/4377121

[18] A. Antonacopoulos, S. Pletschacher, D. Bridson, and C. Papadopoulos, "ICDAR 2009 Page Segmentation Competition," in *2009 10th International Conference on Document Analysis and Recognition*, Jul. 2009, pp. 1370–1374, iSSN: 2379-2140. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5277763

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," May 2016, arXiv:1506.02640 [cs]. [Online]. Available: http://arxiv.org/abs/1506.02640

[21] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows," 2021, pp. 10 012–10 022. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2021/html/Liu_Swin_Transformer_Hierarchical_Vision_Transformer_Using_Shifted_Windows_ICCV_2021_paper

[22] "tesseract-ocr/tesseract," Apr. 2024, original-date: 2014-08-12T18:04:59Z. [Online]. Available: https://github.com/tesseract-ocr/tesseract

[23] "JaidedAI/EasyOCR," Apr. 2024, original-date: 2020-03-14T11:46:39Z. [Online]. Available: https://github.com/JaidedAI/EasyOCR

[24] "PaddlePaddle/PaddleOCR," Apr. 2024, original-date: 2020-05-08T10:38:16Z. [Online]. Available: https://github.com/PaddlePaddle/PaddleOCR

[25] "TorchVision Object Detection Finetuning Tutorial — PyTorch Tutorials 2.2.2+cu121 documentation." [Online]. Available: https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html

[26] "maskrcnn_resnet50_fpn_v2 — Torchvision main documentation." [Online]. Available: https://pytorch.org/vision/main/models/generated/torchvision.models.detection.maskrcnn_resnet50_fpn_v2.html

[27] Y. Li, S. Xie, X. Chen, P. Dollar, K. He, and R. Girshick, "Benchmarking Detection Transfer Learning with Vision Transformers," Nov. 2021, arXiv:2111.11429 [cs]. [Online]. Available: http://arxiv.org/abs/2111.11429

[28] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan. 1979, conference Name: IEEE Transactions on Systems, Man, and Cybernetics. [Online]. Available: https://ieeexplore.ieee.org/document/4310076

[29] "Introducing ChatGPT." [Online]. Available: https://openai.com/blog/chatgpt

[30] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open Foundation and Fine-Tuned Chat Models," Jul. 2023, arXiv:2307.09288 [cs]. [Online]. Available: http://arxiv.org/abs/2307.09288

[31] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M.-A. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mixtral of Experts," Jan. 2024, arXiv:2401.04088 [cs]. [Online]. Available: http://arxiv.org/abs/2401.04088

[32] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7B," Oct. 2023, arXiv:2310.06825 [cs]. [Online]. Available: http://arxiv.org/abs/2310.06825

[33] "spaCy · Industrial-strength Natural Language Processing in Python." [Online]. Available: https://spacy.io/

[34] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, Jul. 2020, pp. 237–242, iSSN: 2157-8702. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9145130

[35] R. Holley, "How Good Can It Get?: Analysing and Improving OCR Accuracy in Large Scale Historic Newspaper Digitisation Programs," *D-Lib Magazine*, vol. 15, no. 3/4, Mar. 2009. [Online]. Available: http://www.dlib.org/dlib/march09/holley/03holley.html

[36] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10. Soviet Union, 1966, pp. 707–710, issue: 8. [Online]. Available: https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf

IX. Appendix

A. LLM Prompt

Task: Convert the provided TV guide text into a structured JSON format that organizes the content by broadcast channel, with details about special descriptions and program listings. Do all of this without using code.

Input Text: This text includes information about various TV programs and their schedules, taken from a TV guide magazine. The text mentions different broadcast channels along with the times and descriptions of specific TV programs.

Output Format: The desired output is a JSON object structured as follows:

JSON from Listing 1

Instructions:

1) Identify and separate information related to different broadcast channels from the text.
2) For each channel, note any special announcements or descriptions under ”special”.
3) List all programs aired on that channel, capturing the start time and the program name or description for each.
4) Ensure all times are formatted correctly and consistently.
5) Arrange the extracted information into the JSON structure provided above, ensuring that each channel and its respective programs are correctly represented.

Example: If the input text of the broadcast channel CNN is ”leder heel uur CNN World News, tenzij anders vermeld. 6.00 This morning 6.30 World business.”, the JSON should look like:

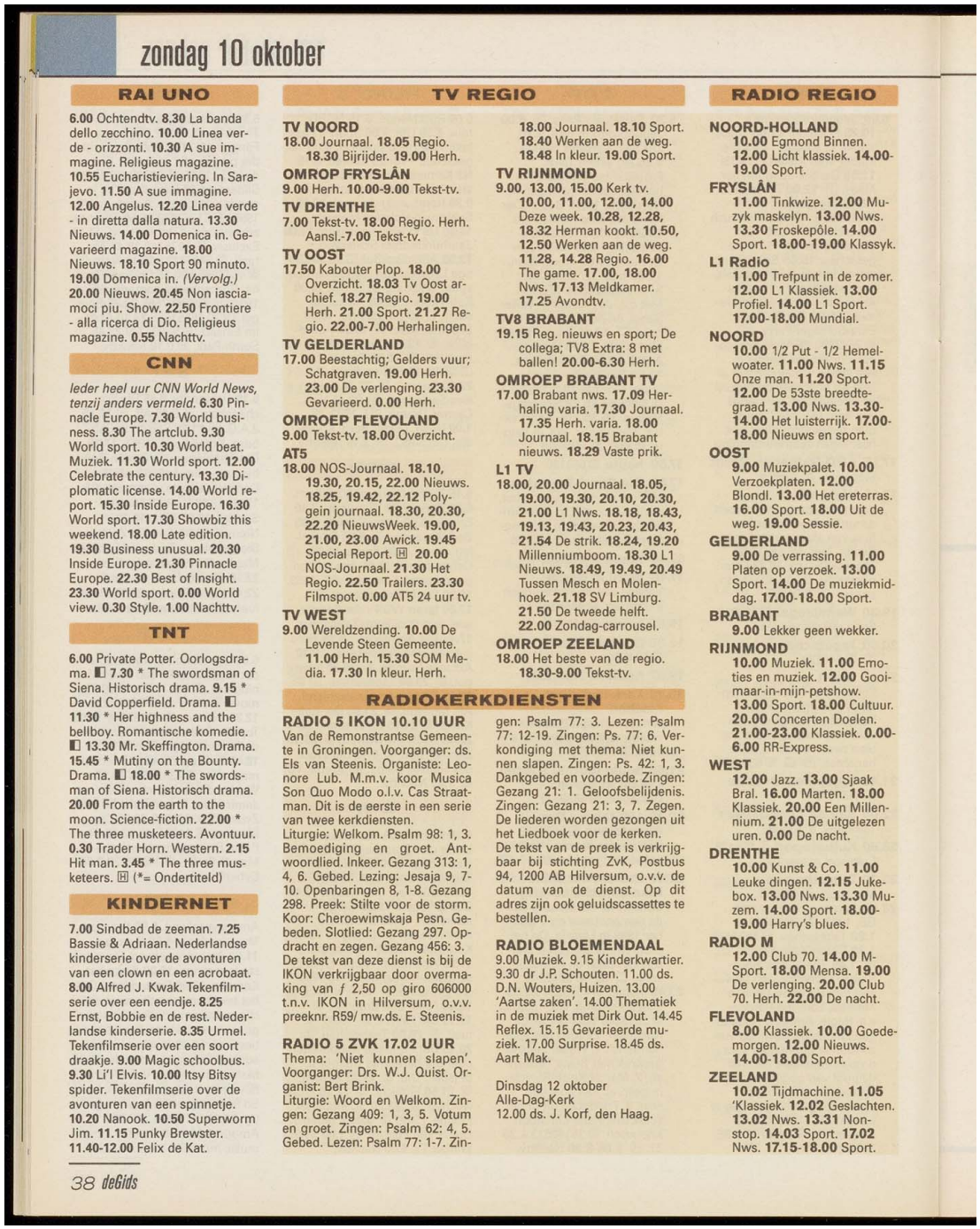
```
{
  "CNN": {
    "special": "leder heel uur CNN World News, tenzij anders vermeld.",
    "programs": [
      {
        "time": "06:00",
        "program": "This morning"
      },
      {
        "time": "06:30",
        "program": "World business."
      }
    ]
  }
}
```

Ensure the JSON is valid and accurately reflects the data provided in the text, with all necessary details included.

B. Mask R-CNN Training Loss

Fig. 13. Mask R-CNN training loss over training epochs

C. TV guide with broader category for section header



zondag 10 oktober

RAI UNO

6.00 Ochtendtv. **8.30** La banda dello zecchino. **10.00** Linea verde - orizzonti. **10.30** A sue immagine. Religieus magazine. **10.55** Eucharistieviering. In Sarajevo. **11.50** A sue immagine. **12.00** Angelus. **12.20** Linea verde - in diretta dalla natura. **13.30** Nieuws. **14.00** Domenica in. Gevarieerd magazine. **18.00** Nieuws. **18.10** Sport 90 minuto. **19.00** Domenica in. *(Vervolg.)* **20.00** Nieuws. **20.45** Non iasciamoci piu. Show. **22.50** Frontiere - alla ricerca di Dio. Religieus magazine. **0.55** Nachttv.

CNN

Ieder heel uur CNN World News, tenzij anders vermeld. **6.30** Pinnacle Europe. **7.30** World business. **8.30** The artclub. **9.30** World sport. **10.30** World beat. Muziek. **11.30** World sport. **12.00** Celebrate the century. **13.30** Diplomatic license. **14.00** World report. **15.30** Inside Europe. **16.30** World sport. **17.30** Showbiz this weekend. **18.00** Late edition. **19.30** Business unusual. **20.30** Inside Europe. **21.30** Pinnacle Europe. **22.30** Best of Insight. **23.30** World sport. **0.00** World view. **0.30** Style. **1.00** Nachttv.

TNT

6.00 Private Potter. Oorlogsdrama. ◧ **7.30** * The swordsman of Siena. Historisch drama. **9.15** * David Copperfield. Drama. ◧ **11.30** * Her highness and the bellboy. Romantische komedie. ◧ **13.30** Mr. Skeffington. Drama. **15.45** * Mutiny on the Bounty. Drama. ◧ **18.00** * The swordsman of Siena. Historisch drama. **20.00** From the earth to the moon. Science-fiction. **22.00** * The three musketeers. Avontuur. **0.30** Trader Horn. Western. **2.15** Hit man. **3.45** * The three musketeers. 🄷 (*= Ondertiteld)

KINDERNET

7.00 Sindbad de zeeman. **7.25** Bassie & Adriaan. Nederlandse kinderserie over de avonturen van een clown en een acrobaat. **8.00** Alfred J. Kwak. Tekenfilmserie over een eendje. **8.25** Ernst, Bobbie en de rest. Nederlandse kinderserie. **8.35** Urmel. Tekenfilmserie over een soort draakje. **9.00** Magic schoolbus. **9.30** Li'l Elvis. **10.00** Itsy Bitsy spider. Tekenfilmserie over de avonturen van een spinnetje. **10.20** Nanook. **10.50** Superworm Jim. **11.15** Punky Brewster. **11.40-12.00** Felix de Kat.

TV REGIO

TV NOORD
18.00 Journaal. **18.05** Regio. **18.30** Bijrijder. **19.00** Herh.

OMROP FRYSLÂN
9.00 Herh. **10.00-9.00** Tekst-tv.

TV DRENTHE
7.00 Tekst-tv. **18.00** Regio. Herh. Aansl.-**7.00** Tekst-tv.

TV OOST
17.50 Kabouter Plop. **18.00** Overzicht. **18.03** Tv Oost archief. **18.27** Regio. **19.00** Herh. **21.00** Sport. **21.27** Regio. **22.00-7.00** Herhalingen.

TV GELDERLAND
17.00 Beestachtig; Gelders vuur; Schatgraven. **19.00** Herh. **23.00** De verlenging. **23.30** Gevarieerd. **0.00** Herh.

OMROEP FLEVOLAND
9.00 Tekst-tv. **18.00** Overzicht.

AT5
18.00 NOS-Journaal. **18.10, 19.30, 20.15, 22.00** Nieuws. **18.25, 19.42, 22.12** Polygein journaal. **18.30, 20.30, 22.20** NieuwsWeek. **19.00, 21.00, 23.00** Awick. **19.45** Special Report. 🄷 **20.00** NOS-Journaal. **21.30** Het Regio. **22.50** Trailers. **23.30** Filmspot. **0.00** AT5 24 uur tv.

TV WEST
9.00 Wereldzending. **10.00** De Levende Steen Gemeente. **11.00** Herh. **15.30** SOM Media. **17.30** In kleur. Herh. **18.00** Journaal. **18.10** Sport. **18.40** Werken aan de weg. **18.48** In kleur. **19.00** Sport.

TV RIJNMOND
9.00, 13.00, 15.00 Kerk tv. **10.00, 11.00, 12.00, 14.00** Deze week. **10.28, 12.28, 18.32** Herman kookt. **10.50, 12.50** Werken aan de weg. **11.28, 14.28** Regio. **16.00** The game. **17.00, 18.00** Nws. **17.13** Meldkamer. **17.25** Avondtv.

TV8 BRABANT
19.15 Reg. nieuws en sport; De collega; TV8 Extra: 8 met ballen! **20.00-6.30** Herh.

OMROEP BRABANT TV
17.00 Brabant nws. **17.09** Herhaling varia. **17.30** Journaal. **17.35** Herh. varia. **18.00** Journaal. **18.15** Brabant nieuws. **18.29** Vaste prik.

L1 TV
18.00, 20.00 Journaal. **18.05, 19.00, 19.30, 20.10, 20.30, 21.00** L1 Nws. **18.18, 18.43, 19.13, 19.43, 20.23, 20.43, 21.54** De strik. **18.24, 19.20** Millenniumboom. **18.30** L1 Nieuws. **18.49, 19.49, 20.49** Tussen Mesch en Molenhoek. **21.18** SV Limburg. **21.50** De tweede helft. **22.00** Zondag-carrousel.

OMROEP ZEELAND
18.00 Het beste van de regio. **18.30-9.00** Tekst-tv.

RADIOKERKDIENSTEN

RADIO 5 IKON 10.10 UUR
Van de Remonstrantse Gemeente in Groningen. Voorganger: ds. Els van Steenis. Organiste: Leonore Lub. M.m.v. koor Musica Son Quo Modo o.l.v. Cas Straatman. Dit is de eerste in een serie van twee kerkdiensten.
Liturgie: Welkom. Psalm 98: 1, 3. Bemoediging en groet. Antwoordlied. Inkeer. Gezang 313: 1, 4, 6. Gebed. Lezing: Jesaja 9, 7-10. Openbaringen 8, 1-8. Gezang 298. Preek: Stilte voor de storm. Koor: Cheroewimskaja Pesn. Gebeden. Slotlied: Gezang 297. Opdracht en zegen. Gezang 456: 3.
De tekst van deze dienst is bij de IKON verkrijgbaar door overmaking van ƒ 2,50 op giro 606000 t.n.v. IKON in Hilversum, o.v.v. preeknr. R59/ mw.ds. E. Steenis.

RADIO 5 ZVK 17.02 UUR
Thema: 'Niet kunnen slapen'. Voorganger: Drs. W.J. Quist. Organist: Bert Brink.
Liturgie: Woord en Welkom. Zingen: Gezang 409: 1, 3, 5. Votum en groet. Zingen: Psalm 62: 4, 5. Gebed. Lezen: Psalm 77: 1-7. Zingen: Psalm 77: 3. Lezen: Psalm 77: 12-19. Zingen: Ps. 77: 6. Verkondiging met thema: Niet kunnen slapen. Zingen: Ps. 42: 1, 3. Dankgebed en voorbede. Zingen: Gezang 21: 1. Geloofsbelijdenis. Zingen: Gezang 21: 3, 7. Zegen.
De liederen worden gezongen uit het Liedboek voor de kerken.
De tekst van de preek is verkrijgbaar bij stichting ZvK, Postbus 94, 1200 AB Hilversum, o.v.v. de datum van de dienst. Op dit adres zijn ook geluidscassettes te bestellen.

RADIO BLOEMENDAAL
9.00 Muziek. 9.15 Kinderkwartier. 9.30 dr J.P. Schouten. 11.00 ds. D.N. Wouters, Huizen. 13.00 'Aartse zaken'. 14.00 Thematiek in de muziek met Dirk Out. 14.45 Reflex. 15.15 Gevarieerde muziek. 17.00 Surprise. 18.45 ds. Aart Mak.

Dinsdag 12 oktober
Alle-Dag-Kerk
12.00 ds. J. Korf, den Haag.

RADIO REGIO

NOORD-HOLLAND
10.00 Egmond Binnen. **12.00** Licht klassiek. **14.00-19.00** Sport.

FRYSLÂN
11.00 Tinkwize. **12.00** Muzyk maskelyn. **13.00** Nws. **13.30** Froskepôle. **14.00** Sport. **18.00-19.00** Klassyk.

L1 Radio
11.00 Trefpunt in de zomer. **12.00** L1 Klassiek. **13.00** Profiel. **14.00** L1 Sport. **17.00-18.00** Mundial.

NOORD
10.00 1/2 Put - 1/2 Hemelwoater. **11.00** Nws. **11.15** Onze man. **11.20** Sport. **12.00** De 53ste breedtegraad. **13.00** Nws. **13.30-14.00** Het luisterrijk. **17.00-18.00** Nieuws en sport.

OOST
9.00 Muziekpalet. **10.00** Verzoekplaten. **12.00** Blondl. **13.00** Het ereterras. **16.00** Sport. **18.00** Uit de weg. **19.00** Sessie.

GELDERLAND
9.00 De verrassing. **11.00** Platen op verzoek. **13.00** Sport. **14.00** De muziekmiddag. **17.00-18.00** Sport.

BRABANT
9.00 Lekker geen wekker.

RIJNMOND
10.00 Muziek. **11.00** Emoties en muziek. **12.00** Gooimaar-in-mijn-petshow. **13.00** Sport. **18.00** Cultuur. **20.00** Concerten Doelen. **21.00-23.00** Klassiek. **0.00-6.00** RR-Express.

WEST
12.00 Jazz. **13.00** Sjaak Bral. **16.00** Marten. **18.00** Klassiek. **20.00** Een Millennium. **21.00** De uitgelezen uren. **0.00** De nacht.

DRENTHE
10.00 Kunst & Co. **11.00** Leuke dingen. **12.15** Jukebox. **13.00** Nws. **13.30** Muzem. **14.00** Sport. **18.00-19.00** Harry's blues.

RADIO M
12.00 Club 70. **14.00** M-Sport. **18.00** Mensa. **19.00** De verlenging. **20.00** Club 70. Herh. **22.00** De nacht.

FLEVOLAND
8.00 Klassiek. **10.00** Goedemorgen. **12.00** Nieuws. **14.00-18.00** Sport.

ZEELAND
10.02 Tijdmachine. **11.05** 'Klassiek. **12.02** Geslachten. **13.02** Nws. **13.31** Nonstop. **14.03** Sport. **17.02** Nws. **17.15-18.00** Sport.

Fig. 14. TV guide with broader category for section header